The R Journal

Volume 10/1, July 2018

A peer-reviewed, open-access publication of the
R Foundation for Statistical Computing

Contents

News and Notes

Prospective authors will find detailed and up-to-date submission instructions on the Journal's homepage.

R Journal Homepage:
http://journal.r-project.org/

Email of editors and editorial board:
firstname.lastname@R-project.org

R Journal Homepage:
https://journal.r-project.org/

The R Journal is indexed/abstracted by EBSCO, DOAJ, and Thomson Reuters.

Editorial

by John Verzani

On behalf of the Editorial Board, I am pleased to present Volume 10, Issue 1 of the R Journal. This issue contains 36 contributed articles. The majority of which cover new or newly enhanced packages on CRAN.

As of writing, CRAN has over 12,500 contributed packages. Including BioConductor packages, there are over 14,000 packages readily installable for R users. Despite the large combined size of the repositories, there are still numerous new contributions made each year, as highlighted in the note "Changes on Cran." The majority of submissions to the R Journal cover add-on packages for R. In this issue over a dozen articles are focused on newly developed packages for applied statistical modeling. As examples, the article by Wurm and Rathouz describes their **gldrm** package for semiparametric generalized linear models; the article by Kim, Zhang, and Zhou describes their **MGLM** package for multivariate categorical data; and Garcia-Donato and Forte's article present their **BayesVarSel** package. For hypothesis tests there are articles on the **HHG** package for non-parametric independence tests; and an article describing the **onewaytests** package, which provides an interface to numerous oneway tests. We have several articles on imputation, including descriptions of the **ImputeRobust**, **FHDI**, and **imputeTestbench** packages.

Several contributions are related to managing complexity, either in data or computational time. The article by Happ, Harrar, and Bathke on their **HRM** package discusses challenges arising in high-dimensional longitudinal data; Kraemer, Reichstein, and Mahecha write about their packages **dimRed** and **coRanking** which enhance R's dimension reduction facilities; and Liu and Quertermous describe their **sinib** package for precise calculations of a useful probability distribution.

Though packages on CRAN do come and go, many are constantly improved and updated. A few articles discuss enhancements to existing packages, for example we have Burkner's article on improvements to his **brms** package. For years, spatial data has been analyzed by R users with the **sp** package. One of **sp**'s authors, Pebesma, describes the important new package **sf** in the article "Simple Features for R" for spatial data. This package is more tightly coupled to standard representations of such data. Spatial data users will also be interested in the article describing **rpostgis** about using a data base add on for spatial data within R.

The Editorial Board has encouraged submissions with comparisons and benchmarking of available packages on CRAN, as in some instances packages offer overlapping features. In this edition we have a few contributions providing overviews. As an example, the "Collections in R" article by Barry reviews several packages providing support, in addition to base R, for various type of collections.

Finally, as R continues to provide value to more and more disciplines, there are a large number of field-specific articles. Examples, among many, include the article on **PortfolioOptim** and its application to optimizing financial portfolios; the article on the **mopa** package and its application to climate data; and the article on the **rtip** package and its application to income data.

In addition the News and Notes section contains the usual updates on CRAN, the Bioconductor project, and several conferences that have highlighted R's usage.

I'd like to thank Roger Bivand for his excellent leadership as editor in chief for the past two issues, welcome Olivia Lau to the Editorial Board, and say farewell to Michael Lawrence from the Editorial Board. Finally, I'd like to thank the enormous number of reviewers who have helped significantly shape the articles contained herein. Their peer-review is invaluable and always most appreciated.

John Verzani
John.Verzani@r-project.org

A System for an Accountable Data Analysis Process in R

by Jonathan Gelfond, Martin Goros, Brian Hernandez, and Alex Bokov

Abstract Efficiently producing transparent analyses may be difficult for beginners or tedious for the experienced. This implies a need for computing systems and environments that can efficiently satisfy reproducibility and accountability standards. To this end, we have developed a system, R package, and R Shiny application called **adapr** (Accountable Data Analysis Process in R) that is built on the principle of accountable units. An accountable unit is a data file (statistic, table or graphic) that can be associated with a provenance, meaning how it was created, when it was created and who created it, and this is similar to the 'verifiable computational results' (VCR) concept proposed by Gavish and Donoho. Both accountable units and VCRs are version controlled, sharable, and can be incorporated into a collaborative project. However, accountable units use file hashes and do not involve watermarking or public repositories like VCRs. Reproducing collaborative work may be highly complex, requiring repeating computations on multiple systems from multiple authors; however, determining the provenance of each unit is simpler, requiring only a search using file hashes and version control systems.

Introduction

Reproducibility, accountability and transparency are increasingly accepted as core principles of science and statistics. While some of the impetus for the interest in reproducibility has come from high-profile incidents that reached the sphere of the general public (Carroll, 2017), there is a consensus that reproducibility is fundamental to scientific communication and to the acceleration of the scientific process (Wilkinson et al., 2016). Furthermore, transparency and reproducibility are deemed necessary by the American Statistical Association's Ethical Guidelines for Statistical Practice (American Statistical Association, 2016). There have been significant strides in improving the set of available tools for constructing reproducible analyses such as the concept and implementation of literate computing, version control systems, and mandates by journals and sponsors that create standards for submitting data, code, and systems for reproducibility. Also, interest in data science has expanded the number of researchers and students who are undertaking data analyses, which underlies the need for tools manageable for a broad range of user expertise. While there is commercial software such as SAS (SAS, Cary, NC) and Stata (StataCrop, College Station, TX, USA) and open-source tools that have systems for reproducible computations, these may have significant challenges. Both open-source and commercial tools could encounter a number of barriers such as 1) the tool is focused on one dimension of the reproducibility problem rather than offering a complete system, 2) the tool may be hard to identify, 3) they may have subtle differences in capabilities, making it hard for users to select among them, 4) documentation or support are in short supply, and 5) they require degrees of sophistication that beginners lack.

Before the introduction of R, Becker and Chambers (1988) proposed a framework for auditing data analyses that could replicate and verify calculations called the AUDIT S-plus (Becker et al., 1988) system. Their auditing system would not only extract the computing steps required to produce the analysis, but also would identify the patterns of the creation and use of particular objects within the directed acyclic graph of computations. Currently, many R packages facilitate reproducible analyses with literate programing or by tracking computational results. Some of these are listed within the CRAN Task View: *Reproducible Research* (Zeileis, 2005). The **knitr** (Xie, 2015) and **rmarkdown** (Allaire et al., 2017; Baumer et al., 2014) packages are widely used and enable literate programing by weaving R code and results into the same document in many formats. Tracking computational results with the **cacher** packgage (Peng, 2008) can be useful for efficient storage and reproducibility, and the **rctrack** package (Liu and Pounds, 2014) tracks computations including random seeds in a read-only format without modifications of existing R code. The package **RDataTracker** (Lerner and Boose, 2014) provides a very granular level of tracking R objects within a script with minimal overhead on the user, and the **recordr** (Slaughter et al., 2015) package tracks dependencies within an R script and does not require the modification of the code, but this facility is limited to specific read/write methods (e.g. `read.csv`,`write.csv`,`ggsave`,`readLines`,`writeLines`, etc.). The **remake** (Fitzjohn et al., 2015) package proposes a different model for managing the complexity of analyses. Rather than instrumenting arbitrary or minimally modified R code for tracking, **remake** provides a structured approach for projects in which dependencies are documented explicitly. An interesting addition is the **archivist** package (Biecek and Kosinski, 2017) that can store R code and R objects, handle multiple versions thereof, and has many features that enhance sharing and browsing results.

Besides R, other open source programming languages and computing environments are developing reproducibility tools such as Python's Sumatra (Davison et al., 2014) which stores computations for simulation or analysis as an electronic lab notebook. These notebooks represent a type of dynamic document (Becker, 2014) that combines data, code, and results within the same object. Dynamic documents enable a very high level of transparency, and while technically challenging to implement, are becoming an increasingly prevalent means of producing and sharing results. The tool Jupyter (Ragan-Kelley et al., 2014), originally developed to be polyglot including languages such as Python, is a web-based electronic notebook built for literate programming which is extendable to multiple languages, recently including R. The Jupyter system has many features such as interactivity, ability to share notebooks, graphical user interfaces, and the potential for high performance computing. Other systems such as Traverna (Wolstencroft et al., 2013) and Galaxy (Afgan et al., 2016) are geared toward workflow development and allow multiple programing languages, web-based sharing and integrate high performance computing. The development of *compendia* is also important to reproducibility, Gentleman and Temple Lang (2007) introduced the concept of a compendium that is comprised of data, code, and the computational results that also allows the distribution of the software tools. This goes beyond reproducibility by distributing a large set of general tools thereby enabling other researchers to use and assess these methodologies in different contexts.

We present the **adapr** (Accountable Data Analysis Process in R) package to facilitate reproducible and accountable computations. The objective is to use a structured, transparent, and self-contained system that would allow users to quickly and easily generate reports and manage the complexity of analyses while minimizing overhead. Unlike other tools such as Jupyter, Traverna and Galaxy, **adapr** is built in R for R and is geared toward simplicity for beginning R users with some minimal alterations to standard R coding. **adapr** is an R package that uses a R **shiny** (Chang et al., 2017) graphical user interface to interact with the directed acyclic graph (DAG) that represents the data analysis. This structured system would facilitate multiple data analysts in interpreting or collaborating on projects. Although **adapr** is capable of literate programming using R Markdown, **adapr** is designed for producing files (graphics, tables, etc.) that are distinct from the code that produced them and can be shared within a scientific collaboration. We call these files accountable units because the system can provide their providence, meaning the data, code dependencies, date, and authorship. This functionality is sustained by computing the cryptographic hashes (Rabin and others, 1981) of the result files and storing these alongside the R code within the version control system Git (Hamano and Torvalds, 2005). It is important to note that the primary data are not version tracked, but only the file hashes, this leads to a check of the identity of the data, but does not allow for reproducing data if modified so that primary data should be stored in another archive. The reasons for this are that 1) the analysis should not modify the primary data, 2) the primary datasets should be stored in a secured version system or database, and 3) tracking versions of the primary data is most efficiently done using a database rather that system such as Git.

The idea of accountable units is similar to that of verifiable computational results (VCRs) proposed by Gavish and Donoho (2011). These authors recognized the difficulties and of ensuring long-term re-execution of source code, and argued the only way of ensuring this is to virtualize and store the full computing platform. They further argued that the source code, the inputs, and output are most important to evaluating the reasoning behind published results rather than strict computational reproduction, which can be time-consuming and technically challenging. There is a bright distinction between collecting a history of computations compared to retaining and sharing the ability to reproduce the final results. We have emphasized elsewhere (Gelfond et al., 2014) that, beyond reproducibility, the analytical history is essential to accountable data analyses based upon formal theories of accountability (Jagadeesan et al., 2009). These theories imply that an evaluator (e.g. peer reviewer) cannot make a valid assessment of a communication (e.g. scientific analysis) unless the evaluator is aware of a verifiable context and history. Furthermore, the computational time required to identify a history is that of text search, which most often can be accomplished much more simply and quickly than attempting to recompute the analysis. We see **adapr** as fulfilling and assisting in creating some of the necessary components of VCRs. While **adapr** is not intended as a tool for instrumenting arbitrarily structured R code to track computation input/output, this package is aimed at providing a transparent organizational structure while establishing provenance and reproducibility assuming the stability of the computing platform.

Setting up adapr and defining a project

The **adapr** package is available from CRAN (Comprehensive R Archive network, cran.r-project.org) with the latest package version available on GitHub (github.com) that can be installed with `adaprUpdate()`. In order to initially set up **adapr**, there is a helper function `defaultAdaprSetup()` that will ask a short sequence of questions (default project directories, preferred username, ver-

sion control preference, etc.) and will create the first project 'adaprHome' within the project directory (e.g. by default the computer's 'Document' directory). This setup function must be executed in ordered for **adapr** to function properly. If RStudio (RStudio Team, 2016) is used, this setup function can utilize the RStudio environment to locate system resources such as *pandoc* automatically. The setup function creates a directory at the top level of the user directories called 'ProjectPaths' with two files 'projectid_2_directory_adapr.csv' and 'adapr_options.csv' that are required to locate projects and configure **adapr**, respectively. The location of the `ProjectPaths` options directory can be changed from the default by setting an option within the R Profile using the statement `options(adaprHomeDir='myPathToOptions')`.

An **adapr** project is a set of analyses based on a given collection of data that is structured as a directed acyclic graph. The project is contained within a folder (a file directory with the same name as the project) with all data, R scripts, and results. A new project can be created with the function `initProject` that can be invoked as `initProject('myProject')`. When a new project is initialized, the project is created as a single folder within a file system (OS X, Linux or Windows) with the following structure:

```
/
├── Data
├── Programs
│   ├── Set of R script files operating on /Data
│   ├── Support
│   │   ├── Library Files (list of R packages used by project)
│   │   └── R shiny Apps
│   ├── Markdown (set of R markdown files)
│   └── Dependencies (set of tab-delimited files storing dependency information)
└── Results (Main)
    └── Set of Results Directories (named corresponding to each R script)
```

The fundamental unit of the project is an R script that is linked to three corresponding elements. The first element is a 'Results' folder with the same name as the R Script that resides within the project's main 'Results' directory. The second is an R markdown file with the same name (with file extension '.Rmd') within the Markdown directory. The third element is a dependency file with the same name as the script (with the file extension `.txt`) within the 'Dependency' directory. The dependency file contains the read/write history of the R script. When a project is initialized by default, a 'read_data.R' file is created and executed automatically. This, in turn, creates a results directory 'Results/read_data.R' and an R markdown file 'Markdown/read_data.Rmd' and a corresponding dependency file 'Dependency/read_data.txt'.

In addition to the main project directory, each project is associated with a 'Publish' directory that can be used to share selected results with collaborators. The Publish directory could be a Dropbox, file server or web server directory.

Initializing projects and adapr R scripts

The function call `projectInit('myProject')` in the R console will create a new project that by default contains a single R script 'read_data.R'. A side effect of this function is to automatically set current project to the new project. Lastly, the project's settings can also be changed with the function `relocateProject` by specifying the project and the new settings. It is also possible to import projects that were already built by a collaborator. For example,

```
relocateProject('importedProjectID',project.path='parentDirectory')
```

imports a new project called 'importedProjectID' that is within a subdirectory of 'parentDirectory'.

The **adapr** package contains a sample project called 'adaprTest' that can be loaded into the users file system with the command `loadAdaprTest()`. To create another R script within that project, one can execute the following console commands:

```
>loadAdaprTest()
>setProject('adaprTest')
>makeScript(r='postProcess.R',description='mixed model analysis',
           project.id=getProject())
```

The setProject function specifies the project and the getProject function retrieves the project ID in other functions by default so that the user does not have to respecify the project. The makeScript function will, by default, automatically open the corresponding script with the default R file browser. Each script (e.g. 'postProcess.R') is given a text string description (e.g. "mixed model analysis"), and also, every file that is read or written within a project has an associated text string description. Timely descriptions are an important part of tracking because they facilitate the *gradual accumulation* of metadata that promotes efficient transparency rather than an automated code instrumentation, which may not easily generate precise, correct, or human-interpretable metadata tags. When a program is created with makeScript, the program is executed, creating a script-specific directory within the parent 'Results' directory, R markdown file, and the dependency file. The **adapr** R scripts have a structured header that identifies the project and the script, and a footer that writes the read/write dependencies into a file corresponding to the R script within the 'Program/Dependency' directory. The header and footer that are automatically created by makeScript and are shown below.

```
rm(list=ls()) #Clears R workspace
set.seed( 2011 ) # Sets random seed

library('adapr')

source.file <-'read_data.R'  #Name of file matching filename /Programs directory name
project.id <- 'project_name'  # Name of project
source_info <- create_source_file_dir(source.description='reads data')
# Source description will be linked to R script

# Library statements here (e.g.\, Library('ggplot2'))

# Begin program body here



# End Program Body
dependency.out <- finalize_dependency()
```

Within the **adapr** script, the key step within the header is the create_source_file_dir function that creates the source_info object of type 'list'. The source_info object is stored within the R options and can be retrieved using getSourceInfo(). This is used to collect the read/write information within the body of the R script by wrapping the read/write commands within three R commands. The first is the Read function:

```
Read(file.name = 'data.csv', description = 'Data file',
          read.fcn = guess.read.fcn(file.name), ...)
```

This command returns the file object read by the function read.fcn and tracks the description and the file hash with "..." passed as additional arguments to the read function read.fcn. Note that in the usage example below, it is transparent and parsimonious, especially given that the Read function defaults to reading from the project's data directory:

```
Dataset <- Read('datafile.csv','my first dataset')
```

The argument read.fcn could be any function that given a file, returns an R object, and so, Read is not limited to flat files. To facilitate the identification of a project's data files, the listDatafiles() function call produces a list of files within the data directory.

The second command is the Write function:

```
Write(obj = NULL, file.name = 'data.csv', description = 'Result file',
        write.fcn = guess.write.fcn(file.name), date = FALSE, ...)
```

This example writes object 'cars' into the Results directory as a comma separated value (CSV) file

```
Write(cars,'cars.csv','speed and distance')
```

and this example writes object 'cars' into the Results directory as a R data object.

```
Write(cars,'cars.rda','speed and distance')
```

This command writes the R object obj using the function write.fcn and tracks the description and the file hash with "..." passed as additional arguments to write.fcn. If the file.name argument has

a `.rda` extension then the R object is written to the results directory as an R data object. The third function used by **adapr** creates graphics files:

```
Graph(file.name = 'ouput.png', description = 'Result file',
        write.fcn = guess.write.fcn(file.name), date = FALSE, ...)
```

This command opens a graphics device for plotting using the function `write.fcn` and tracks the description and the file hash with "..." passed as additional arguments to `write.fcn`. The function `guess.write.fcn` uses the file suffix to determine the type of file to write (e.g. pdf, png, etc). The date variable is a logical that determines whether a date is added to the filename. This function is followed by graphics command set and closed by a `dev.off()` or `graphics.off()` statement. Below is an example.

```
Graph('histogram.png','hist of random normals')
        hist(rnorm(100))
graphics.off()
```

This code snippet creates a file in the results directory of the R script called 'histogram.png' with a histogram of 100 standard normal variables.

Another key function is `loadFlex` invoked as the following:

```
LoadedObject <- loadFlex('process_data.R/intermediate_result.rda')
```

The `loadFlex` function reads and returns an R data object with the `.rda` file extension written by another R script within the project. The `loadFlex` function could also read any intermediate result file type if the argument `read.fcn` is specified. This function is critical because it allows the R scripts within a project to build on the results of other R scripts in a transparent and structured manner. This function searches through all results for a file name that matches the argument string to identify the R object to load, and returns the corresponding R object (assigned to `LoadedObject` in the example). This function must follow the call `Write(intermediate,'intermediate_result.rda','normalized data')` within another R script (`process_data.R` in the example) to pass preprocessed data from one R script to another or to collect key results from multiple R scripts. It is important to note that `loadFlex` requires prior use of a `Write` function because **adapr** does not, by default, collect the entire workspace of each R script. The convenience function call `listBranches()` returns a data frame of all intermediate results available for loading along with their descriptions.

By transferring and tracking objects from one script to the next, **adapr** promotes the idea of *modularity*. In this manner, the scripts can have a specific function rather than one script being burdened with all analytical tasks. Modularity promotes transparency by allowing a collaborator or reviewer to quickly identify which scripts are responsible for which analytic tasks, and allowing a collaborator to isolate their contribution to a particular task within a script that is linked to scripts created by others. Lastly, modularizing scripts to perform specific tasks facilitates transforming scripts into functions themselves. Modularization makes clear what the inputs (arguments to `loadFlex`) and outputs (arguments to `Write`) of a particular script are. R functions that are specific to a project are stored within the 'Programs/support_functions' directory. Any R script within this directory will be sourced and loaded within the `create_source_file_dir` function within the **adapr** script header. A function can be created within this directory by using the function call `makeFunction('mySpecialPlot')` in the R console that will create a file 'Programs/support_funcionts/mySpecialPlot.R' and by default will open the function's file for editing.

The **adapr** footer contains the critical function call `finalize_dependency()` that transforms the R script's file input/output (I/O) stored within the object `getSourceInfo()` and writes this data to the project's 'Dependency' directory in the form of a tab-delimited file with the same prefix as the R script. The dependency data within the file includes the R script name, R script description, R script hash, R script file modification time, R script run time, the project ID, the target file that is read/written, the direction of dependency (read or write), the target file description, target file hash, and the target file modification time. The script header function `create_source_file_dir` and footer function `finalize_dependency` execute many side effects that are necessary for tracking and these are listed in Table 1.

For some critical analyses or data objects, it may be useful to store a version history. This means storing more than one version of the object itself, not just the code to reproduce the object. This can be done using the **archivist** package that has been incorporated into **adapr** by creating an **archivist** archive for each **adapr** script in the R script's 'Result' directory. Using the function call `arcWrite(myRobj,"my object description")` in the R script file stores the object within an **archivist** archive. If the R script that wrote the object was 'myRscript.R', this may be loaded into another R Script by using the statement `myObject <-arcRead('myRscript.R','my object description')` that assigns the value of that object to `myObject`.

Table 1: Side effects of header and footer functions

Function	Side effects
`create_source_file_dir`	Create directories for Project and Results Set project id for R session Initialize the file tracking object `source_info` Gather file information on all project files Initialize Git for project (if using Git) Add dependencies files to Git Initialize **archivist** repo for script Run all R scripts in 'support_function' directory Run all R scripts in script specific 'support_function/myRscript.R' directory Create R markdown file with same file prefix (if not already done) Create publication file (if not already done)
`finalize_dependency`	Pass workspace to R markdown file and render Store file modification time and hash for all dependencies Write dependency information to 'Dependency' directory Add dependency file and R session information to Git

Literate programing using R Markdown

An R Markdown file is created automatically within the 'Programs/Markdown' directory when an **adapr** R script is first created and executed. This file has the '.Rmd' suffix and is rendered when the R script is executed as a side-effect of the footer `finalize_dependency()` function within the corresponding R script. Additional R markdown files can be created using the `createMarkdown` function and rendered within the 'Results' directory with the 'renderRmd' function. Executing the R script will pass its workspace to the markdown file R commands and render these markdown files. By pairing each R script with a markdown file, **adapr** allows the R script to perform transformations and analytical tasks that can be more easily summarized, formatted, and displayed within the R markdown file.

The '.Rmd' file can also be executed by using the "knit" button in RStudio if the file header has the header line `scriptLoader(project.id,R Script)` executed (this is commented out by default). Additionally, file input/output can be tracked using the line `if(checkRmdMode()){dependency.out <-finalize_dependency() }` as the final line in the markdown file. These lines are automatically included in the R markdown files, but are commented out. The RStudio "knit" button can facilitate exploring and debugging; however, it does not direct the rendered markdown into the results directory, and the rendered markdown file would go to the 'Markdown' directory.

Managing R packages

By default, projects share a common R library that is specified at setup. Sometimes users may prefer that project has a project-specific library, instead of the default R library. This may be useful for improving consistency in computations, although the larger libraries can use substantial drive storage. The user may specify library preferences at initialization by setting the following argument within the `projectInit` function: `initProject('myProject',project.libraryTF=libOption)`. The `project.libraryTF` parameter `libOption` may be set to one of 3 values (`"TRUE"`,`"FALSE"`, or `"packrat"`). The default value `FALSE` implies the use of the default library. The value `TRUE` will use the project specific library located in the `Programs/Support/Library` directory, and the value `"packrat"` will imply the **packrat** (Ushey et al., 2016) to manage R packages. The **packrat** package allows the isolation of the package dependencies for R projects and has many features including isolation of the computing environment, cross-platform compatibility and cleaning out unused packages. **adapr** does not set up or operate **packrat**, and this responsibility is left to the user. We have found that the RStudio IDE is helpful for the initialization of **packrat** rather than R console commands.

adapr facilitates loading and tracking R packages using the `Library` function and by tracking the loaded libraries within the R session information. R packages needed for each R script can be loaded with the `Library` function within the R script header. This function will install and/or load the package if it is not installed or loaded. A use case would be `Library('ggplot2','cran')` or `Library('limma','bioc')`. At the end of the program body, the `finalize_dependency` function

writes the R session information to a file. The `getLibrary()` statement will read all of these session information files for a project and collect all of the packages and their versions. This information can be useful for reproducibility, and these packages can be installed using the `installLibrary()` function call. The `installLibrary` function has an argument to install the specific versions, and the packages are installed from CRAN, MRAN (Microsoft R Application Network, `https://mran.microsoft.com`), Bioconductor (Huber et al., 2015), or GitHub. Currently, only the latest compatible version of Bioconductor packages will be installed. If the **packrat** package is used for a project, then it is recommended to use **packrat** functions and the RStudio IDE for managing package versions. An example of how to use these functions is shown below: The following series of console commands loads the example project stored within the **adapr** R package.

```
>loadAdaprTest()
```

This sets the project.

```
>setProject('adaprTest')
```

This returns the directory containing the project library.

```
>getProjectLibrary()
```

This returns the project library's package information.

```
>getLibrary()
```

This installs project library.

```
>installLibrary()
```

There are many technical intricacies of tracking R package use within a project, and there has been significant effort build stable solutions for package reproducibility such as **packrat** and **switchr** (Becker et al., 2017). These difficulties are not currently completely handled by **adapr** and is an area of active development.

Summarizing and visualizing projects

The `reportProject` function and the app can summarize all programs and results within a single HTML document with descriptions, dependency information, and links to programs and results. The app can be launched with within the R console with the function `adaprApp()`. The report information can be obtained within the Report tab of the app in Figure 1. After clicking the "Report" button, the HTML project summary is written to the 'Results/Tree_controller'.R/ directory. The Report tab shown in figure also is capable of launching project R Shiny apps that access project data or Results.

Synchronizing and version control

Two critical parts of accountability and reproducibility are addressed by the `syncProject` and `commitProject` functions as well as within the Synchronize tab of the app. The `syncProject` function can be called in the R console with `syncProject()` after a `setProject('myProject')` call. It will synchronize data, scripts, and results for a project. The project is synchronized if two criteria are met: 1) the file modification times stored within the R scripts dependency files are the same as the modification times within the file system, and 2) if the cryptographic hash (data fingerprint) of the file hash of each file within the project is consistent with all of the dependency files. The `syncProject` function called by `syncProject()` detects lack of synchrony and can execute the corresponding R scripts that are needed to achieve synchrony. It is possible that the DAG has structures ("forks") that allow the R scripts to be run in parallel because they are not dependent on one another. We have added experimental functions `parallelSync` and `monitorParallelSync` that allow parallel execution of the DAG by different cores. Note that `monitorParallelSync` should be executed in a separate R session and will indicate which cores are working on which part of the project DAG. For larger projects, an alternative to synchronizing the whole project is to only synchronize the results of a particular R script. This involves only synchronizing the parts of the DAG that that R script depends upon. This can be done with the `syncTrunk('myscript.R','myProject')` function call. This is illustrated in the code snippet below, and Figure 2 shows the results of the last `graphProject()` statement.

```
>loadAdaprTest() # Loads the project stored within the adapr R package
>setProject('adaprTest') # Sets the project
>syncProject() # Synchronizes the project
>openScript('read_data.R') # Open R script
```

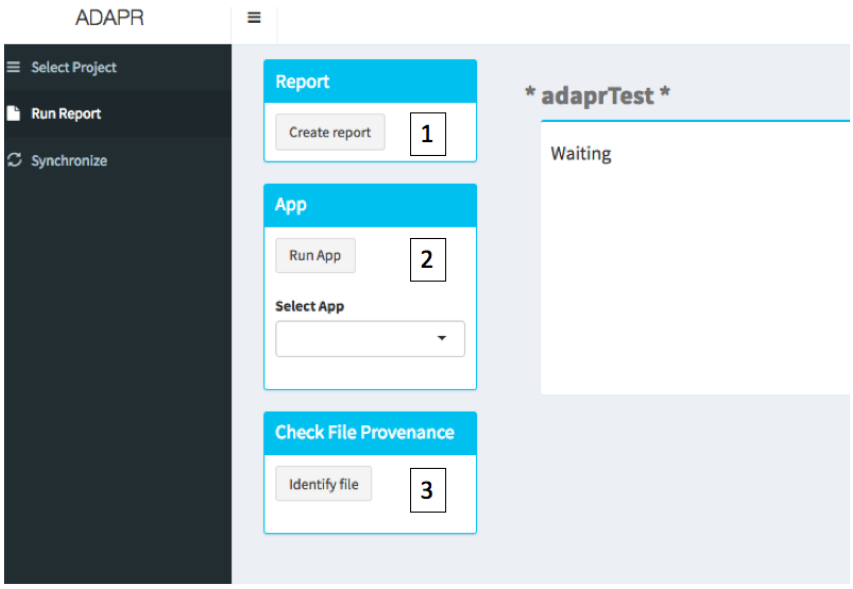


Figure 1: Run report tab. 1. The Create Report button will create an html report listing all R script input/outputs. 2. RunApp selects and launches App within the project App directory. 3. The Identify file button initiates a file select dialogue, and the version control history based upon the file hash is given.

We could make some minor change in 'read_data.R' and then execute the following to synchronize the project.

```
>graphProject() # Shows project DAG with synchronization status
>runScript('read_data.R') # Executes the script in a clean R environment
>syncTrunk('graph_cardata.R') # Executes scripts needed to synchronize
                                          results from 'graph_cardata.R'
>graphProject() # Shows project DAG with synchronization status
```

adapr optionally uses the Git version control system, which can be operated with the `commitProject()` function and the Synchonize tab. The Synchronize tab is shown in Figure 3. Git must be installed for this feature to work. There are platform independent and free Git clients available for download (`https://git-scm.com/downloads/guis`). While **adapr** handles Git tracking and snapshots, it does not currently revert the current version of the project back to a previous snapshot. We recommend that reversions be performed using Git directly or through a Git client, which have more features and visualization capabilities. The function `commitProject()` and button within the app takes a project snapshot of all R scripts and adds the synchronization status (synchronized or not) to the commit message. Because the file names and *file hashes/fingerprints* are stored within dependency files that are tracked by the version control system, any data, R script, or result file can be matched by its hash/fingerprint within the version control history to determine its provenance. The **adapr** function `Digest` is a wrapper for functions in the **digest** (Eddelbuettel et al., 2017) package and is used the compute the cryptographic hash (not serialized) of a file, and this file hash can be identified by a Git history search to determine its provenance within a project. **adapr** uses Git operating functions that were adapted from those within the **devtools** (Wickham and Chang, 2015) and **git2r** (Widgren and others, 2017) . If Git is used, **adapr** will automatically perform a commit after `syncProject`. This allows the user to capture all result file hashes and to identify which results were obtained in a synchronized state. It is important to note that **adapr** automatically adds all R script files into the project's Git repository, but it does not automatically use formal version control for data or results files because these files may be too large for the version control system to efficiently track the changes. If the user would like to store multiple versions of a result, then we recommend using the `arcWrite` function. However, by default, the file hash is computed and stored within the 'Dependency' directory for all files that are read or written within the project so that all relevant files and their provenances can be identified if a commit was made when the file was read/written. File histories can be identified within the "Run Report" tab that executes a file hash search and returns the Git commit associated with the file including the date, author, and file description. The Git history of a file is also obtained with the `gitProvenance` function with arguments `project.id` and `filepath` that opens a file selection window and runs a Git search for the hash of the file within the project.

There is an interactive project map feature within the "Synchronize" tab. The "Examine Program

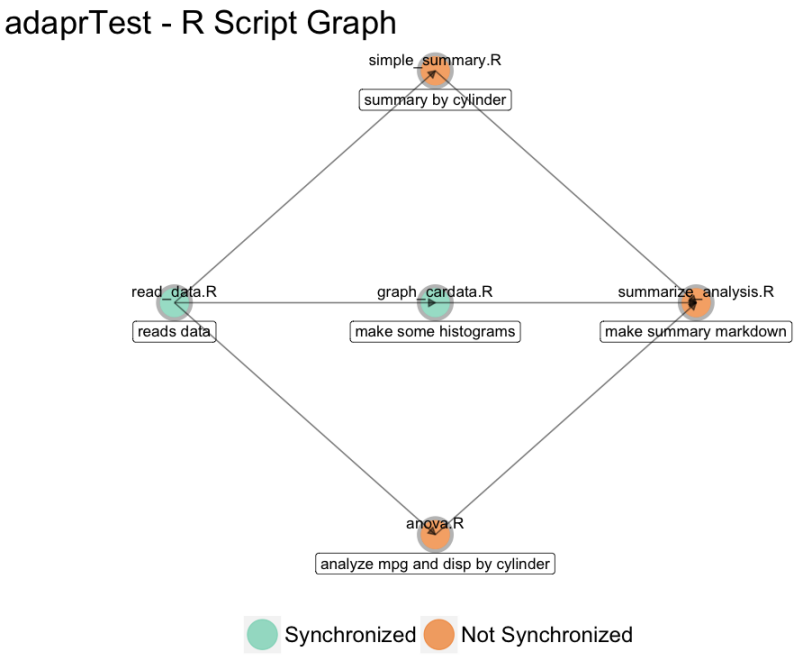


Figure 2: Resulting graph from `graphProject()` in code snippet. The R Script Graph shows dependencies of R scripts in the "adaprTest" project with their synchronization status after `syncTrunk('graph_cardata.R')`, which synchronizes results from the R script 'graph_cardata.R'.

Graph" button displays the DAG of all the project's R scripts colored by their synchronization states. These scripts can be executed by clicking on the colored point, and their file I/O can be examined by selecting or "brushing" over the colored point. This reveals the file I/O graphic in an adjacent panel. These files are labeled by their descriptions and can be opened by clicking on the file or their file directories may be browsed by selecting or "brushing" over the colored file labels. The graph may alternatively be displayed using the `graphProject()` function. A limitation is that very large DAGs or very large input/output lists for an R script can overwhelm the visual space of these graphics. Future versions will address this issue.

Sharing results

Specified result files from the project can be copied to a shared 'Publish' directory using the `publishProject('myProject')` function. This allows users to share or update a list a selected results with one command. The `publishProject` function reads a file within the ' Programs/Support_functions' directory that contains a list of files that are selected for publication (i.e. copied to the project publication directory, which may be a shared server folder, a web server, or cloud server such as Dropbox). Publishing can be automated by including the `publishProject` function within an **adapr** R script. The file that contains paths to the results that will be published can be opened for editing with the `browsePubFiles('myProject')` function.

Configuration

The **adapr** package can be updated and configured within the `setAdaprOptions` and `getAdaprOptions` functions. An option can be set with the syntax `setAdaprOptions("optionName","optionValue")`. The key options are summarized in Table 2. For example, the use of Git can be activated by `setAdaprOptions('git','TRUE')`. Git itself can be configured using Git commands or within R using the **adapr** function call `gitConfigure(username,email)`.

Convenience functions

We have found that certain project oriented functionality can save time and effort, and we have compiled a list of functions that we have found useful in Table 3. These can be used without specifying

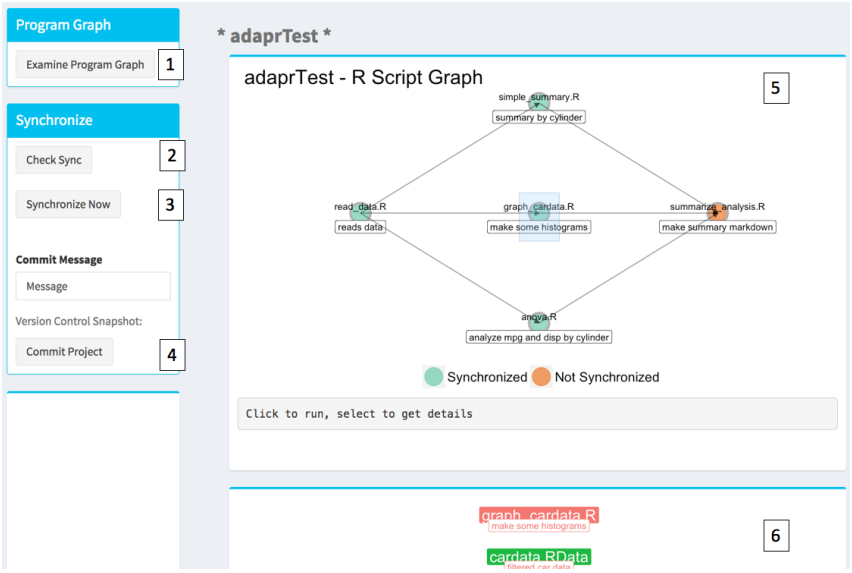


Figure 3: Synchronize Tab. 1. Examine program graph creates or updates two interactive plots. 2. Check the synchronization status and synchronize project by running necessary R scripts. 3. Syncronize project. A pop-up progress bar will open. 4. Commit message with Git version control snapshot of project. Commit Project button operates Git. 5. R Script Graph shows dependencies of R scripts. Clicking on button runs the R scripts. Selecting or brushing button opens panel 6. 6. Displays the inputs and output of the selected R script. Clicking on the file label opens the file or selecting opens the file folder.

Table 2: List of adapr options

Purpose	Option Name	Value
Path to system resources	PATH	Paths separated by system delimiter
Specify use of Git	git	TRUE or FALSE
Default project path	project.path	File path
Default publication path	publish.path	File path
Default R library	library	File path to default library
Specify use of adapr beta version	adaprBeta	TRUE or FALSE

Table 3: Convenience functions

Purpose	Function example
Browse main function cheat sheet for **adapr**	`adaprSheet()`
Show file usage information for all projects	`fileInfoProjects()`
Open project directory in file browser	`showProject()`
Open project or source results directory in file browser	`showResults()`
Open project's R script or R markdown file lists R scripts if not specified	`openScript()`
List intermediate result objects	`listBranches()`
Plot synchronization status of project scripts	`graphProject()`
Execute R script within a project optionally create a markdown log	`runScript('read_data.R',` `'adaprTest',logRmd=TRUE)`
Search R scripts for string	`searchScripts('lmer')`
Copy data file to project data directory	`importData()`
Return results directory within R script	`resultsDir()`
Return data directory within R script	`dataDir()`
Track file reads without using `Read`	`ReadTrack('arraydata.idat',` `'read by limma package')`
Track file writes without using `Write`	`WriteTrack('myFile.docx',` `'From ReporteRs package')`

the project argument after the `setProject('myProject')` function. The two pages of the function summary "cheat sheet" that is opened with the `adaprSheet()` function call is shown in Figures 4 and 5.

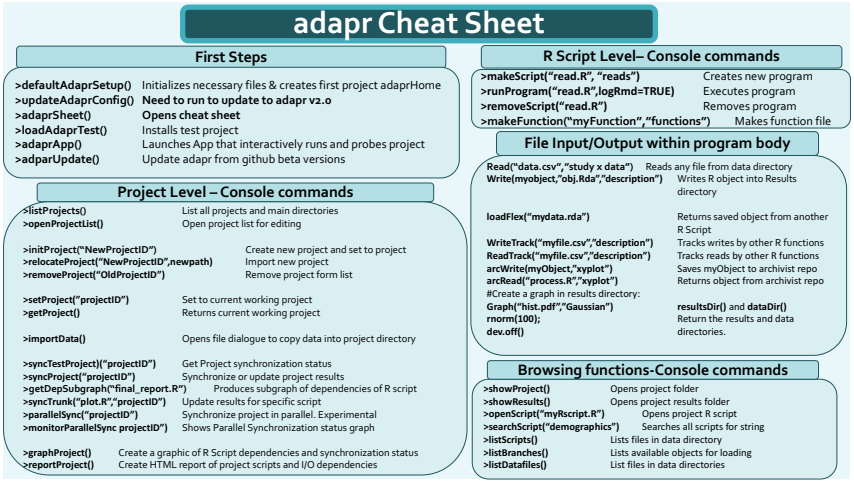


Figure 4: Cheat sheet snapshot. This figure shows first out of two pages of the function summary cheat sheet. This is opened from the R console using the function call `adaprSheet()`.

Conclusion

Becker and Chambers (1988) have written that "the complexity of the structure that represents a realistically large analysis makes a thorough interactive investigation very challenging. New statistical, computational, and graphical techniques will be needed." Since then, there have been many strides towards that goal. The **adapr** package is a demonstration of how scientific reproducibility and accountability can be accelerated by automatically structuring, tracking, and implementing analyses. Beginning R programmers may benefit from this approach by not having to create their own system, and while more experienced analysts have likely created their own methods for structuring analyses, it is possible that there are certain features of **adapr** that they could integrate into their systems. These features include adding a standard structure to project directories, checking the synchronization status of the project, selectively synchronizing part of the DAG, linking short text descriptions to data and results files, creating a HTML summary of the full project, interactive project maps, and version control integration with file hash search capability for results and programs. This integrated version control

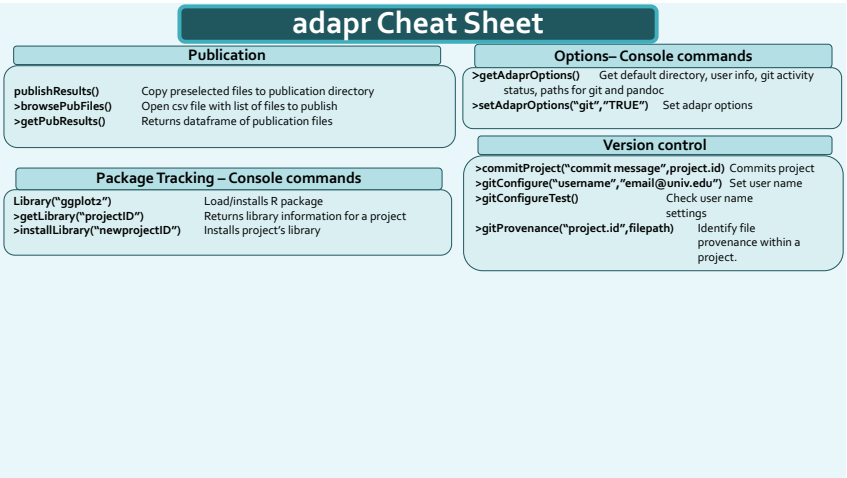


Figure 5: Cheat sheet snapshot. This figure shows second out of two pages of the function summary cheat sheet. This is opened from the R console using the function call `adaprSheet()`.

feature tracks file hashes of data, programs, and results and can capture the evolution of the analysis, not just the final results. This points to an important challenge: the analyses in many scientific articles are extremely intricate and necessarily become more complex as the data are inevitably explored. This makes version control system indispensable for unraveling the history of analysis.

Limitations

The limitations of **adapr** include the inability to directly track changes in external databases, file hash computation can be slow for files that are multiple gigabytes, the requirements for the user to use **adapr** read/write functions for tracking, and the package management system is less capable than packages such as **packrat** and **switchr**. Currently, **adapr** can be used with external database input functions, but these would not be tracked as inputs with the `Read` function because they are not within the file system. We are considering options for tracking external database input for future **adapr** versions. One workaround for this is to have an R script within the project that downloads the relevant database queries and writes these objects to the file system with a `Write` function, and these query results can be read into another R script and tracked as a dependency using the `loadFlex` function. The **adapr** package is also limited by its dependency on other R packages and by the fact that running the Shiny app while programming would require two R sessions. We have found that using project oriented functions (`makeScript`, `syncProject`, `commitProject`, `initProject`, `setProject`, etc.) within the console makes for a more fluid experience, but this requires that the user have an idea of which functions to utilize, which is why we created the `adaprSheet()` function to give a quick overview. Also, beginning R users may be just learning the art of programming, which could make using a structured programming system like **adapr** difficult if that structure is violated. There are some safeguards built into **adapr**, but violating the header/footer structure can result in errors. We have used **adapr** in a class with first-time programmers, and this has greatly informed our development. For example, the function `makeScript` originally only created the script within the file system, but now, it also opens the script with the default R program so that the students do not have to search for it. The students use a single function `Library` function instead of two functions `install.packages` and `library`. Also, we have found that the `runScript` function is very useful for teaching reproducibility as it lets the students run the script in a clean environment. Students can also be introduced to R through the R scripts, and after a few classes, can be introduced to R markdown. This allows time to get familiar with R, before adding the additional complexity of R markdown syntax. Because each **adapr** script is paired with and passes its environment to an R markdown file, the students can see how R markdown can nicely format the objects that they have previously created. While a limitation of this work is that the **adapr** user base is small, one of the key aspects of the system is that it has been tested on a variety of projects. The **adapr** package and earlier versions have been tested in over 160 projects. These projects were conducted by five statisticians in an academic health center (AHC). The projects included clinical trials, genomic analyses, experimental data from basic sciences, as well as projects created specifically to test **adapr**. Figure 6 represents the directed acyclic graphs with structures related to the complexity of the projects.

Other potential benefits

Additional goals of **adapr** include efficiency and transparency. The AHC environment creates a high demand for statistical analyses as well as significant pressure to reduce the time-to-completion. We have studied process of statistical consulting in some detail (Schmidt et al., 2016), and the use of the system has not led to any significant increase in lead times, but it has subjectively improved the ease of collaboration among analysts. The use of any structured system like **adapr** can be highly beneficial when a long dormant project becomes active again or a project is passed from one analyst to another. Another advantage of the system is that reports and interactive maps allow users who are not familiar with R the opportunity to browse the analysis, which promotes transparency. These maps also help the users, regardless of experience level, to orient themselves to large or complex analyses. Structuring projects in directories that are systematically scanned for activity (e.g. every 30 minutes) allows automated tracking of the work patterns within projects (e.g. file modification times of R scripts or markdown files within the projects directory). This facilitates the objective estimation of work-hours that are useful for reporting, billing, or resource allocation.

Lastly, given that accountability has been recently added as a requirement for authorship by the International Committee of Medical Journal Editors (Resnik et al., 2015; International Committee of Medical Journal Editors, 2016), scientific reviewers or readers may benefit from knowing that an **adapr** data analysis had a high degree of accountability or reproducibility. Hence, we suggest adding the statement "The data analysis was conducted using an accountable process." to the 'Statistical methods' or 'Data analysis' section of an article that used **adapr**. This designates that: 1) a documented system was used for the version-controlled analytic history, 2) the provenance of results can be checked, and 3) that the analyses and results have descriptions, a documented structure, and a means of reproduction. Note that version control systems such as Git store the author's name and email associated with commits so that these naturally document contributions.

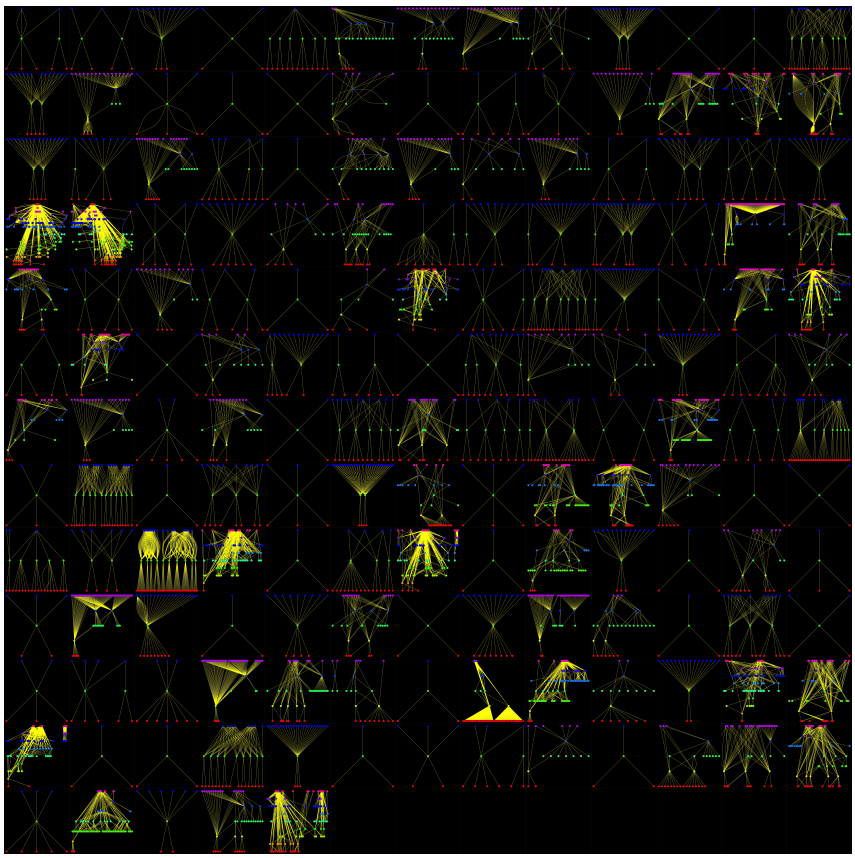


Figure 6: The directed acyclic graphs of 160 projects that used the **adapr** package. Each panel is a project, and points are files (data files, R scripts, and output files) organized by their dependencies (flowing from top to bottom) within that project.

Acknowledgement

The authors thank Emily Burnett for her contribution to early versions of the user interface. The authors would like to thank the editor and two anonymous reviewers for substantial improvement of this work. The project described was supported by the National Cancer Institute and the National Center for Advancing Translational Sciences, National Institutes of Health, through the grants GM070335, NIA Shock Center P30AG013319, NIA Pepper Center P30AG044271, CTRC P30 Cancer Center Support Grant CA054174 and the Clinical and Translational Science Award (CTSA) UL1 TR001120. The content is solely the responsibility of the authors and does not necessarily represent the official views of the NIH.

Bibliography

E. Afgan, D. Baker, M. van den Beek, D. Blankenberg, D. Bouvier, M. Čech, J. Chilton, D. Clements, N. Coraor, C. Eberhard, and others. The galaxy platform for accessible, reproducible and collaborative biomedical analyses: 2016 update. *Nucleic Acids Research*, page 1, 2016. URL https://doi.org/10.1093/nar/gkw343. [p7]

J. Allaire, J. Cheng, Y. Xie, J. McPherson, W. Chang, J. Allen, H. Wickham, A. Atkins, R. Hyndman, and R. Arslan. *Rmarkdown: Dynamic Documents for R*, 2017. URL https://CRAN.R-project.org/package=rmarkdown. R package version 1.6. [p6]

A. American Statistical Association. ASA Ethical Guidelines ethical guidelines for statistical practice. http://www.amstat.org/asa/files/pdfs/EthicalGuidelines.pdf, 2016. Accessed: 2017-11-20. [p6]

B. Baumer, M. Cetinkaya-Rundel, A. Bray, L. Loi, and N. J. Horton. R markdown: Integrating a reproducible analysis tool into introductory statistics. *arXiv preprint arXiv:1402.1894*, 2014. URL https://doi.org/10.1063/pt.5.028530. [p6]

G. Becker. *Dynamic Documents for Data Analytic Science*. University of California, Davis, 2014. [p7]

G. Becker, C. Barr, R. Gentleman, and M. Lawrence. Enhancing reproducibility and collaboration via management of R package cohorts. *Journal of Statistical Software*, 82(1):1–17, 2017. URL https://doi.org/10.18637/jss.v082.i01. [p12]

R. A. Becker and J. M. Chambers. Auditing of data analyses. *SIAM Journal on Scientific and Statistical Computing*, 9(4):747–760, 1988. URL https://doi.org/10.1137/0909049. [p6, 16]

R. A. Becker, J. M. Chambers, and A. R. Wilks. The new s language. *Pacific Grove, Ca.: Wadsworth & Brooks, 1988*, 1988. URL https://doi.org/10.2307/2531523. [p6]

P. Biecek and M. Kosinski. archivist: An R package for managing, recording and restoring data analysis results. *Journal of Statistical Software*, 82(11):1–28, 2017. URL https://doi.org/10.18637/jss.v082.i11. [p6]

A. Carroll. Science needs a solution for the temptation of positive results. the upshot. *New York Times, May*, 29, 2017. URL https://doi.org/10.3886/icpsr09104. [p6]

W. Chang, J. Cheng, J. Allaire, Y. Xie, and J. McPherson. *Shiny: Web Application Framework for R*, 2017. URL https://CRAN.R-project.org/package=shiny. R package version 1.0.5. [p7]

A. Davison, M. Mattioni, D. Samarkanov, and B. Teleczuk. Sumatra: a toolkit for reproducible research. *Implementing reproducible research*, pages 57–79, 2014. URL https://doi.org/10.1201/b16868. [p7]

D. Eddelbuettel, A. Lucas, J. Tuszynski, H. Bengtsson, S. Urbanek, M. Frasca, B. Lewis, M. Stokely, H. Muehleisen, D. Murdoch, J. Hester, W. Wu, Q. Kou, T. Onkelinx, M. Lang, and V. Simko. *Digest: Create Compact Hash Digests of R Objects*, 2017. URL https://CRAN.R-project.org/package=digest. R package version 0.6.12. [p13]

R. Fitzjohn, D. Falster, and Cornwell. Remake: Make-like declarative workflows in r. *URL https://github.com/richfitz/remake*, 2015. [p6]

M. Gavish and D. Donoho. A universal identifier for computational results. *Procedia Computer Science*, 4:637–647, 2011. URL https://doi.org/10.1016/j.procs.2011.04.067. [p7]

J. A. Gelfond, C. M. Klugman, L. J. Welty, E. Heitman, C. Louden, and B. H. Pollock. How to tell the truth with statistics: The case for accountable data analyses in team-based science. *Journal of translational medicine & epidemiology*, 2(2), 2014. URL https://doi.org/10.1002/sim.4282. [p7]

R. Gentleman and D. Temple Lang. Statistical analyses and reproducible research. *Journal of Computational and Graphical Statistics*, 16(1):1–23, 2007. URL https://doi.org/10.1198/106186007x178663. [p7]

J. Hamano and L. Torvalds. Git-fast version control system, 2005. [p7]

Huber, W., Carey, V. J., Gentleman, R., Anders, S., Carlson, M., Carvalho, B. S., Bravo, H. C., Davis, S., Gatto, L., Girke, T., Gottardo, R., Hahne, F., Hansen, K. D., Irizarry, R. A., Lawrence, M., Love, M. I., MacDonald, J., Obenchain, V., Ole's, A. K., Pag'es, H., Reyes, A., Shannon, P., Smyth, G. K., Tenenbaum, D., Waldron, L., Morgan, and M. Orchestrating high-throughput genomic analysis with Bioconductor. *Nature Methods*, 12(2):115–121, 2015. URL https://doi.org/10.1038/nmeth.3252. [p12]

I. International Committee of Medical Journal Editors. Defining the role of authors and contributors. http://www.icmje.org/recommendations/browse/roles-and-responsibilities/defining-the-role-of-authors-and-contributors.html, 2016. Accessed: 2017-12-30. [p18]

R. Jagadeesan, A. Jeffrey, C. Pitcher, and J. Riely. Towards a theory of accountability and audit. In *Computer Security–ESORICS 2009*, pages 152–167. Springer-Verlag, 2009. [p7]

B. Lerner and E. Boose. *RDataTracker: Collecting Provenance in an Interactive Scripting Environment*, 2014. URL https://www.usenix.org/system/files/conference/tapp2014/tapp14_paper_lerner.pdf. [p6]

Z. Liu and S. Pounds. An r package that automatically collects and archives details for reproducible computing. *BMC bioinformatics*, 15(1):1, 2014. URL https://doi.org/10.1186/1471-2105-15-138. [p6]

R. D. Peng. Caching and distributing statistical analyses in r. *Journal of Statistical Software*, 2008. URL https://doi.org/10.18637/jss.v026.i07. [p6]

M. O. Rabin and others. *Fingerprinting by Random Polynomials*. Center for Research in Computing Techn., Aiken Computation Laboratory, Univ., 1981. [p7]

M. Ragan-Kelley, F. Perez, B. Granger, T. Kluyver, P. Ivanov, J. Frederic, and M. Bussonier. The jupyter/ipython architecture: a unified view of computational research, from interactive exploration to communication and publication. In *AGU Fall Meeting Abstracts*, volume 1, page 07, 2014. [p7]

D. B. Resnik, A. M. Tyler, J. R. Black, and G. Kissling. Authorship policies of scientific journals. *Journal of medical ethics*, pages medethics–2015, 2015. URL https://doi.org/10.1136/medethics-2015-103171. [p18]

RStudio Team. *RStudio: Integrated Development Environment for R*. RStudio, Inc., Boston, MA, 2016. URL http://www.rstudio.com/. [p8]

S. Schmidt, M. Goros, H. M. Parsons, C. Saygin, H.-D. Wan, P. K. Shireman, and J. A. Gelfond. Improving initiation and tracking of research projects at an academic health center a case study. *Evaluation & the Health Professions*, page 0163278716669793, 2016. URL https://doi.org/10.1177/0163278716669793. [p18]

P. Slaughter, C. Jones, M. Jones, and L. Palmer. Recordr: Provenance tracking for r. *URL https://github.com/NCEAS/recordr*, 2015. [p6]

K. Ushey, J. McPherson, J. Cheng, A. Atkins, and J. Allaire. *Packrat: A Dependency Management System for Projects and Their R Package Dependencies*, 2016. URL https://CRAN.R-project.org/package=packrat. R package version 0.4.8-1. [p11]

H. Wickham and W. Chang. *Devtools: Tools to Make Developing R Packages Easier*, 2015. URL http://CRAN.R-project.org/package=devtools. R package version 1.8.0. [p13]

S. Widgren and others. *git2r: Provides Access to Git Repositories*, 2017. URL https://CRAN.R-project.org/package=git2r. R package version 0.19.0. [p13]

M. D. Wilkinson, M. Dumontier, I. J. Aalbersberg, G. Appleton, M. Axton, A. Baak, N. Blomberg, J.-W. Boiten, L. B. da Silva Santos, P. E. Bourne, and others. The fair guiding principles for scientific data management and stewardship. *Scientific data*, 3:160018, 2016. URL https://doi.org/10.1038/sdata.2016.18. [p6]

K. Wolstencroft, R. Haines, D. Fellows, A. Williams, D. Withers, S. Owen, S. Soiland-Reyes, I. Dunlop, A. Nenadic, P. Fisher, and others. The taverna workflow suite: Designing and executing workflows of web services on the desktop, web or in the cloud. *Nucleic acids research*, page gkt328, 2013. URL https://doi.org/10.1093/nar/gkt328. [p7]

Y. Xie. *Dynamic Documents with R and Knitr*, volume 29. CRC Press, 2015. [p6]

A. Zeileis. Cran task views. *R News*, 5(1):39–40, 2005. [p6]

Jonathan Gelfond
Department of Epidemiology and Biostatistics
UT Health San Antonio, TX, USA
gelfondjal@uthscsa.edu

Martin Goros
Department of Epidemiology and Biostatistics
UT Health San Antonio, TX, USA
goros@uthscsa.edu

Brian Hernandez
Department of Epidemiology and Biostatistics
UT Health San Antonio, TX, USA
hernandezbs@uthscsa.edu

Alex Bokov
Department of Epidemiology and Biostatistics
UT Health San Antonio, TX, USA
bokov@uthscsa.edu

RealVAMS: An R Package for Fitting a Multivariate Value-added Model (VAM)

by Jennifer Broatch, Jennifer Green, and Andrew Karl

Abstract We present **RealVAMS**, an R package for fitting a generalized linear mixed model to multimembership data with partially crossed and partially nested random effects. **RealVAMS** utilizes a multivariate generalized linear mixed model with pseudo-likelihood approximation for fitting normally distributed continuous response(s) jointly with a binary outcome. In an educational context, the model is referred to as a multidimensional value-added model, which extends previous theory to estimate the relationships between potential teacher contributions toward different student outcomes and to allow the consideration of a binary, real-world outcome such as graduation. The simultaneous joint modeling of continuous and binary outcomes was not available prior to **RealVAMS** due to computational difficulties. In this paper, we discuss the multidimensional model, describe **RealVAMS**, and demonstrate the use of this package and its modeling options with an educational data set.

Introduction

Originally developed in Broatch and Lohr (2012), the RealVAMS model is a multivariate generalized linear mixed model (GLMM) that extends previous theory by allowing the simultaneous joint modeling of continuous and binary outcomes within a value-added framework. Although the model may be used in a variety of contexts (e.g., sports, medicine, etc.), it is particularly useful in education for studying relationships between teachers and students. For example, the RealVAMS model can estimate potential teacher contributions toward a variety of student outcomes, including quantitative scores from different subjects and different assessments, as well as categorical success outcomes such as graduation status. In addition, the RealVAMS model estimates the relationships between the predicted teacher effects for these various measures of student success.

The **RealVAMS** package employs the RealVAMS model to analyze multimembership data. Multimembership data (Browne et al., 2001) poses unique modeling and estimation issues. Bates et al. (2015) describes the random effects that result from this type of data structure as "partially crossed" and "partially nested" because subjects are not perfectly nested under one higher level effect and may be nested under multiple, different higher level effects. The newly released **lme4** (Bates et al., 2015) adequately analyzes this type of effect, but cannot model a continuous outcome and a binary outcome at the same time. Specifically, `glmer` cannot utilize both a normal and binary link in a single function call.

Modeling both types of responses is essential for capturing a broader picture of the latent effects. If a relationship between the latent effects on the outcome variables exists, this simultaneous estimation improves prediction of outcomes. In an application of the RealVAMS model to the context of sports, Broatch and Karl (2018) found that the joint model benefits from significantly improved median log-loss and absolute residuals of cross-validation predictions if the estimated "team-level" random effects are correlated. However if the team-level random effects are not correlated, the joint model does not show improved predictive power. These "team effects" are analogous to the "teacher effects" in an education setting such that if different educational outcomes are somewhat correlated, the joint model will likely lead to improved predictions of academic achievement and more accurate estimates of teacher effectiveness. However, if the effects are not correlated, the joint model will likely not be an improvement over separate models that can be run in **lme4**.

Similar to other value-added models (VAMs) used within an educational context, the RealVAMS model aims to estimate the effects of educational factors such as teachers, schools, and districts on student learning while controlling for prior student achievement and other covariates, if available. These statistical models typically measure the correlation between different educational factors and student achievement, and do not directly measure causation (American Statistical Association, 2014). Consequently, "teacher effect" estimates represent unexplained classroom-level heterogeneity (Lockwood et al., 2007) and should be "viewed within the context of quality improvement" (American Statistical Association, 2014, p. 2) rather than used for high-stakes evaluation purposes. Although most VAMs are based on standardized test scores alone (American Statistical Association, 2014), the RealVAMS model enables users to explore the potential contributions of educational factors and programs to other types of student outcomes, providing a more nuanced characterization of student success to help enhance the quality and fidelity of educational programs and instruction.

The combination of different types of responses allows measurement of educational effectiveness using additional indicators of student success, not just standard test scores. **RealVAMS** also estimates

the relationships among the multidimensional random effects on all continuous and binary outcomes. Therefore, in addition to obtaining multidimensional real-world estimates of value added by teachers or programs, education researchers can simultaneously explore the relationships among the effects on typical continuous measures of student achievement and real-world outcomes, which are frequently binary. In this paper, we present the RealVAMS model and discuss the technical challenges overcome by **RealVAMS** when using multimembership data. We also demonstrate the use of **RealVAMS** (Karl et al., 2015) with an educational data set.

RealVAMS model

This section discusses the unique features of the multidimensional RealVAMS model and how the model extends current theory.

Normal multimembership model

For simplicity, we begin by introducing the RealVAMS model for the case where all responses for subject i are continuous, then extend this model to include a binary outcome. To begin, the RealVAMS model presented in a traditional mixed model framework is

$$\mathbf{y}_i = \mathbf{X}_i\boldsymbol{\beta} + \mathbf{S}_i\boldsymbol{\gamma} + \boldsymbol{\varepsilon}_i. \tag{1}$$

The vector of responses for subject i consists of t potentially different continuous responses that do not require time ordering or scaling. The $[t \times (p+1)]$ matrix $\mathbf{X}_i$ gives the p covariates for subject i. In an educational context, these covariates may include time-invariant covariates such as gender and ethnicity, as well as time-varying covariates such as participation in free lunch programs. The latent effect $j = 1, \ldots, m$ for responses $k = 1, \ldots, t$ is represented by the vector $\boldsymbol{\gamma}$. For latent effect j, $\boldsymbol{\gamma}_j = (\gamma_{j1}, \ldots, \gamma_{jt})'$ is the t-vector of effects, where γ_{jk} represents the latent effect of j on response k. Thus, the vector $\boldsymbol{\gamma} = [\boldsymbol{\gamma}_1^{'}, \ldots, \boldsymbol{\gamma}_m^{'}]^{'}$ is the concatenation of the individual m vectors. The $(t \times tm)$ matrix $\mathbf{S}_i$ is the multimembership design matrix. In an educational context, this matrix indicates which teacher instructs student i for each of the t responses since the continuous response is often measured in multiple years. In the model, $\mathbf{S}$ can be expanded to allow for fractional membership. The error term $\boldsymbol{\varepsilon}_i$ is assumed to follow a normal distribution with mean 0 and variance $\mathbf{R}_i$, where $\mathbf{R}_i$ is unrestricted and allows the subject responses to be correlated over the t responses. Additionally, all $\boldsymbol{\gamma}_j$ and $\boldsymbol{\varepsilon}_i$ are assumed to be uncorrelated. The model also assumes that $\boldsymbol{\gamma}$ is normally distributed with mean 0 and variance $\mathbf{G} = \text{blockdiag}(\mathbf{G}_1, \ldots, \mathbf{G}_m)$ where

$$\mathbf{G}_j = \begin{pmatrix} g_{11} & g_{12} & \cdots & g_{1t} \\ \vdots & & & \vdots \\ g_{1t} & g_{2t} & \cdots & g_{tt} \end{pmatrix}$$

and all $\mathbf{G}_j$ are initially assumed to be equal.

The multimembership structure precludes a factorization of the design and covariance matrices across subjects, which makes the non-sparse estimation routines inefficient and non-scalable for both linear and nonlinear mixed models. Thus, **RealVAMS** uses sparse matrix routines provided by **Matrix** (Bates and Maechler, 2015). Broatch and Lohr (2012) use the pseudo-likelihood approach to obtain approximations to the maximum likelihood estimators; they then adopt the penalized quasi-likelihood approach in SAS `PROC GLIMMIX` (SAS Institute Inc., 2008) to approximate maximum likelihood estimates (Breslow and Clayton, 1993; Wolfinger and O'Connell, 1993). However, the SAS estimation in presented in Broatch and Lohr (2012) is limited to less than 30 random effects, which is not feasible or efficient for a large scale model.

Normal-binary multimembership model

The model executed by **RealVAMS** simultaneously analyzes normal-binary responses with correlated random effects. To illustrate, model (1) can be extended to a GLMM that also includes a binary response. In this case, the continuous response model (1) is adapted by defining the binary response as an unobservable latent trait $\tilde{\mathbf{y}}$. The binary response is defined as $r_{ij} = 1$ if the latent variable $\tilde{y}_{ij} > 0$. For example, if college entry was the binary response of interest, then the response would equal 1 if the student entered college and 0 otherwise. A response $r_{ij} = 1$ is therefore equivalent to the subject's underlying latent trait exceeding some threshold, $\tilde{y}_{ij} > 0$. Thus,

$$\tilde{\mathbf{y}}_i = \mathbf{X}_i\boldsymbol{\beta} + \mathbf{S}_i\boldsymbol{\gamma} + \tilde{\boldsymbol{\varepsilon}}_i. \tag{2}$$

where $\gamma \sim N(0, \mathbf{G})$ and $\tilde{\varepsilon} \sim N(0, \mathbf{R})$. To maintain the identifiability in the parameter space, we take $\mathbf{R}_i$ to be a correlation matrix fixed at 1. This requirement is only necessary for the variance of the latent process for the binary response; the variance components for the continuous responses are unrestricted. The other terms in the model are defined as in model (1). The GLMM contains the linear mixed model inside the inverse link function:

$$E[\mathbf{y}_i \mid \gamma] = g^{-1}(\mathbf{X}_i\boldsymbol{\beta} + \mathbf{S}_i\gamma) \tag{3}$$

where $g(\cdot)$ is a multivariate probit link function for a binary response, following the recommendation of McCulloch (1994) and Rabe-Hesketh and Skrondal (2001).

The computational difficulty in estimating the random effects and correlations between those effects increases with the inclusion of different distributional outcomes. Simultaneously modeling a binary response requires the inclusion of a nonlinear link function in the integrand of the expression for the log-likelihood. No closed-form solutions are available for these integrals. Due to the multi-membership structure, the dimension of the intractable integral is equal to the number of random effects, with at least one random effect for each effect/response combination. Thus, **RealVAMS** uses the pseudo-likelihood (PL) method of Wolfinger and O'Connell (1993) to linearize the likelihood function. We use an EM algorithm to maximize the likelihood function, iteratively calculate the pseudo-response, and then re-run EM to maximize. PL is the estimation routine used to handle the probit link, and EM is used within each iteration for maximization instead of Newton-Raphson. This approach differs from the Bayesian estimation techniques used in Zhang et al. (2016). Specifically, we use a doubly-iterative pseudolikelihood routine: the outer pseudolikelihood procedure is given in Wolfinger and O'Connell (1993) and the inner loop follows Karl et al. (2013). For convenience, the comments within the **RealVAMS** source code identify specific equation numbers in Karl et al. (2013) and Wolfinger and O'Connell (1993) .

The random effects in the RealVAMS model are correlated such that the off-diagonal elements of G_j are not assumed to be 0, and there is a potential for high correlation among the random effects. The covariance matrix of the random teacher effects, $\mathbf{G}$, may be near (or on) the boundary of the parameter space. The Newton-Raphson (NR) methods are known to fail to converge in these situations (Demidenko, 2004). The Cholesky factorization is not an option for the RealVAMS model due to the structure imposed on $\mathbf{G}$. **RealVAMS** presents an efficient alternative: an EM algorithm to maximize the pseudo-likelihood function. Karl et al. (2013) document how the EM algorithm leads to a more stable maxima in the presence of near-singular covariance matrices. These highly efficient estimation routines have already been demonstrated to be effective on this type of multimembership model. Consequently, the resulting RealVAMS package is capable of handling models with tens of thousands of random effects.

Application of the RealVAMS package

Although the RealVAMS model may be used in a variety of contexts in which multimembership data occur (Karl, 2012), we choose to present it within an educational context. In this context, the model is generally referred to as a multidimensional value-added model (VAM). VAMs are popular accountability methods for estimating the potential effects of educational factors such as teachers and schools on student achievement. VAMs typically estimate the potential contributions of these various factors on test scores. Through unexplained classroom-level heterogeneity, existing VAMs measure potential teacher contributions toward the concepts, skills, and abilities directly measured by student exams (Koretz, 2008; McCaffrey et al., 2003) and provide a limited view of how classrooms may more broadly contribute to student success. The addition of a binary response in the RealVAMS model helps to address this issue.

The remainder of this section discusses the three main aspects to utilizing the **RealVAMS** package: data requirements, model specification, and output. To illustrate its features, we present an analysis of an educational data set.

Data requirements

The RealVAMS model extends previous theory to estimate the relationships between potential teacher contributions toward different student outcomes, often exams, and to allow the consideration of a real-world binary outcome such as graduation. Therefore, **RealVAMS** requires that data include at least one year of a continuous response (e.g., standardized test scores) and one binary outcome coded as 0 or 1 (e.g., college entry) to create the response vector. These required responses must be specified as separate data frames. At minimum, the continuous response data frame `score.data` must contain four different columns: "y" containing numeric student scores, "student" containing unique

student identification numbers, "teacher" containing unique teacher identification numbers linking students to their respective teachers, and "year" containing the numeric year or numeric time period indicator. Optionally, additional covariates may exist for modeling purposes and can be included in the `score.data`. Similarly, the binary response data frame `outcome.data` must contain: "`r`" containing the binary response coded as a 0 or 1, "student," "teacher," and "year" containing the numeric year or numeric time period indicator.

To demonstrate the use of the RealVAMS package, we apply the model to examine the effect of teacher effectiveness on college entry and standardized test scores. Student and classroom data were obtained for a single cohort of 11th grade students in a large public school district. Available outcome data included standardized test scores and whether or not each student entered college after graduation. For simplicity, students missing either test scores or college enrollment status were excluded from the analysis. The final data set included 803 students whose records were linked to 71 different teachers. Just as in traditional models, student background and baseline performance scores were included in the model specification. Student information regarding their gender (SEX), ethnicity (ETH), gifted classification (GIF), special needs classification (SPE), and/or English-Language learner classification (ESL) were included. The data set also contained students' math (PLM) and reading (PLR) sub-scale scores from the PLAN assessment, a national assessment administered to 10th grade students. The PLM and PLR scores were included as a baseline measure of student achievement.

The 11th grade state test scores (`y`) and college entry status (`r`) served as the continuous and binary outcome responses, respectively. The original data were separated into two different data frames for input to the `RealVAMS` function: "score" data for the state test scores and "outcome" data for college entry status. The two data frames are linked by teacher ID and student ID.

student	year	SEX	ETH	SPE	ESL	GIF	PLM	PLR	resp.type	teacher	y
148438	1	F	5	0	0	0	15	15	score	6	76
162376	1	F	1	0	0	0	15	17	score	15	36
162952	1	F	1	0	0	0	14	18	score	57	84
163593	1	F	3	0	0	0	16	19	score	69	96
164282	1	M	1	1	0	0	15	12	score	63	99
164460	1	M	1	0	0	0	27	24	score	64	115

Table 1: First six observations of score data. Note the required columns student, year, resp.type, and y.

student	year	SEX	ETH	SPE	ESL	GIF	PLM	PLR	resp.type	teacher	r
148438	1	F	5	0	0	0	15	15	outcome	6	0
162376	1	F	1	0	0	0	15	17	outcome	15	0
162952	1	F	1	0	0	0	14	18	outcome	57	1
163593	1	F	3	0	0	0	16	19	outcome	69	0
164282	1	M	1	1	0	0	15	12	outcome	63	1
164460	1	M	1	0	0	0	27	24	outcome	64	1

Table 2: First six observations of binary outcome data. Note the required columns student, year, resp.type, and r.

To further illustrate the package requirements for multi-year data, the **RealVAMS** help file includes a simple simulated example. The example score data frame `example.score.data` includes 1875 observations on 625 students over 3 years, with 25 different teachers in each year. The example score data frame includes the four required variables:

- **student** represents the unique students,
- **teacher** represents the unique teacher identification,
- **year** is a numeric vector containing the years 1, 2 and 3, and
- **y** represents the student score.

The data frame also contains an optional continuous covariate **cont_var**. Correspondingly, the example binary outcome data frame `example.outcome.data` includes 625 observations and the required two variables: `r` is composed of 0s and 1s representing a simulated binary outcome measured on students, and `student` represents the unique student identification that corresponds to the student identification used in `outcome.data`.

Model specification

The RealVAMS function accommodates different value-added modeling specifications for analyzing multimembership data, including the RealVAMS models (1) and (2). The following syntax illustrates how to analyze the joint continuous and binary outcomes using the RealVAMS function.

```
> res=RealVAMS(score.data,outcome.data,  persistence = "CP"
      score.fixed.effects = formula(~as.factor(SEX)+as.factor(ETH)+as.factor(SPE)
                                    +as.factor(ESL)+as.factor(GIF)+PLM+PLR),
      outcome.fixed.effects =formula(~as.factor(SEX)+as.factor(ETH)+as.factor(SPE)
                                    +as.factor(ESL)+as.factor(GIF)+PLM+PLR),
      school.effects = FALSE, REML = TRUE,
      max.iter.EM = 10,
      outcome.family = binomial(link = "probit"),
      tol1 = 1e-07, max.PQL.it = 30,
      pconv = .Machine$double.eps*1e9,
      var.parm.hessian = TRUE,
      verbose = TRUE)
```

Within the RealVAMS function, the persistence option designates the type of persistence coefficients that are modeled for teacher score effects. Users may choose between complete persistence "CP," as illustrated here, or variable persistence "VP" of teacher score effects. The CP model assumes that teacher effects persist undiminished into the future, whereas the VP model assumes that teacher effects in future years are multiples of the effect in the current year (Lockwood et al., 2007). The multipliers α in the VP model are called persistence parameters and are estimated in the model. In contrast, the CP model fixes the persistence parameters at 1 (Lockwood et al., 2007). The teacher binary outcome effects are modeled with complete persistence, regardless of the selection for the continuous score effects.

The RealVAMS function allows users to account for school effects using the school.effects option (not used in the example above). If school.effects=TRUE, correlated random school-level effects are fit in the continuous and binary response models with zero-persistence (e.g., a student's score in each year is associated with the current school attended, and their binary outcome is associated with the last school the student attended). To use the school effects option, the school ID should be included as a column labeled "schoolID" in both the score.data and the outcome.data data frames.

By default, the RealVAMS function fits the pseudo-response using restricted maximum likelihood (REML) estimation. If REML=FALSE, maximum likelihood estimation is used instead. In addition, users may specify the link function used for the binary outcome response by using the outcome.family option.

Fixed effects are included in the score.data and outcome.data data frames and specified in the model using score.fixed.effects and outcome.fixed.effects. Each of these options utilize the functionality of R's formula class such that categorical variables should be wrapped in an as.factor statement. For example, the RealVAMS syntax includes fixed effects for the factors (SEX, ETH, SPE, ESL, and GIF), continuous covariates (PLM, PLR), and separate intercepts for the continuous score outcome and the binary outcome. To fit a no-intercept model, +0 may be be specified in the formula. To fit an intercept-only model, formula(~ 1) may be used. In addition, an interaction between any two covariates may be specified using the * operator: as.factor(SEX) * PLM, or equivalently, as.factor(SEX) + PLM + as.factor(SEX):PLM.

Other options allow users to specify the efficiency of the estimation process. The model is linearized using a pseudo-likelihood approach (Wolfinger and O'Connell, 1993), and the resulting multiple membership linear mixed model is estimated via an EM algorithm (Karl et al., 2013). The argument max.iter.EM controls the maximum number of EM iterations during each pseudo-likelihood iteration, with tol1 representing the convergence tolerance for the EM algorithm during each interior pseudo-likelihood iteration. By default, convergence is declared for each interior iteration when $(l_k - l_{k-1})/l_k <$ tol1, where l_k is the log-likelihood at iteration k, max.PQL.it specifies the maximum number of outer pseudo-likelihood iterations, and pconv represents the convergence criterion for these outer iterations, similar to the PCOV option in SAS PROC GLIMMIX. By default, the Hessian is used to generate covariance matrices, providing standard errors for the estimated parameters. However, setting the var.parm.hessian to FALSE reduces the run time. When verbose is TRUE, model information is printed at each iteration.

Output

There are many output options available in **RealVAMS**. All parameters of the RealVAMS models (1) and (2) are available for output, including covariance and persistence estimates where appropriate.

Fixed effects

As with a standard mixed model, the function `RealVAMS` returns the fixed effect estimates $\hat{\boldsymbol{\beta}}$ for the given formulae. The fixed effect estimates and standard errors can be obtained by using `summary` or `parameters`. To return all parameter estimates including the fixed effects, use

```
> summary(res)
```

Covariate	State Test	SE	P-Value	College Entry	SE	P-Value
Intercept	-0.702	6.123	0.912	-0.791	0.3	0.008
Male	0.872	1.842	0.638	-0.134	0.102	0.187
Native American	-17.307	13.625	0.204	-1.161	0.728	0.11
African American	-3.496	3.859	0.363	-0.046	0.2	0.818
Asian American	11.05	4.602	0.016	0.566	0.304	0.063
Hispanic American	-9.849	3.789	0.009	-0.433	0.192	0.024
Two or More	-1.629	4.619	0.726	0.021	0.248	0.928
White (Reference)	—	—		—	—	
Special Needs Status	-15	4.103	<.0001	-0.303	0.204	0.139
ELL-YES	-19.513	13.084	0.136	-0.026	0.664	0.968
Gifted-YES	6.78	2.473	0.006	-0.188	0.139	0.177
PLAN-Math	4.972	0.279	<.0001	0.045	0.015	0.003
PLAN- Reading	0.777	0.26	0.003	0.03	0.014	0.035

Table 3: Fixed effects estimates from a RealVAMS model for state test scores (continuous outcome) and probability of college entry (binary outcome). Note: Fixed effects estimates for the probability of college entry are reported on the probit scale.

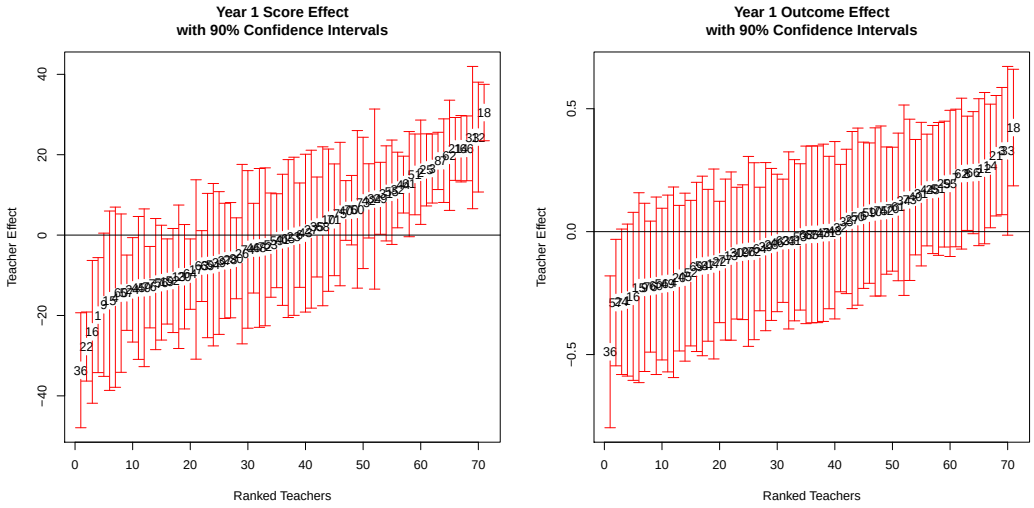
These effects are reported in a manner similar to a standard VAM or **lme4**. The RealVAMS model estimates teacher effects on multiple outcomes which results in a separate fixed effect for each of the two response variables: the continuous standardized state test scores and the binary indicator for college entry. For example, when all other variables are held constant, Hispanic American students are estimated to score on average 9.849 points lower on the state standardized test than white students (95% confidence interval $[-17.274, -2.423]$) and are 16.7% less likely than white students to enter college (95% confidence interval $[-29\%, -2\%]$). Note that the point estimate of -0.4333 presented in the table has been transformed from the probit scale.

These separate estimates for the fixed effects allow one to isolate the marginal effects of a covariate on each of the responses. For some covariates, the fixed effect estimates that are statistically significant for one response are also statistically significant for the other response. However, this need not be the case and may help to illuminate useful information about how a covariate's estimated conditional relationship with one response may differ from its estimated conditional relationship with another response. For example, when all other variables are held constant, students who are classified as gifted are estimated to score on average 6.78 points higher on the state standardized test than those who are not (95% confidence interval $[1.93, 11.627]$), but there is no statistically distinguishable difference in their probability for entering college (95% confidence interval $[-17\%, 3\%]$). In other words, holding everything else constant, a student classified as gifted is likely to perform better on the exam, but is not more likely to attend college than another student who is not classified as gifted.

Random effects

Although other value-added methods may consider the teacher effects as fixed effects, the goal of the RealVAMS model is to estimate the "teacher effects" using random effects $\hat{\boldsymbol{\gamma}}$ and the relationship between these effects $\hat{G}_j$. The presented example assumes that each teacher teaches only one year. If there is more than one year of data, the function appends "(year z)" to each teacher identification number to keep these effects separate where z is the year in which the teacher taught. Similarly, the program appends _outcome to indicate the random effect estimate for the binary outcome. The estimated random "teacher" effects (EBLUP) and their standard errors can be obtained using `teach.effects`:

```
> res$teach.effects[1:4,]
```



(a) Estimated Teacher Effect on Score - Standardized Test

(b) Estimated Teacher Effect on Outcome- Expected College Entry

Figure 1: Estimated effects on continuous score and binary outcome responses with 90% confidence intervals.

	EBLUP	std_error	year
1(year1)	-19.909	8.697	1
1(year1)_outcome	-0.278	0.188	1
10(year1)	3.710	10.760	1
10(year1)_outcome	0.0801	0.208	1

These estimates show that teacher 1 in year 1 had an estimated random effect on students' standardized test scores of -19.909 (standard error $= 8.697$), and an estimated effect of -0.278 on the probit scale (standard error $=$ 0.188) on the probability of students' college entry. These random effects are estimated relative to other teachers in the sample so that the teacher effects are centered on the distributional mean of zero. In this example, we assume complete persistence with `persistence = "CP"`, but changing this argument to `persistence = "VP"` will give the estimated persistence parameter $\hat{\alpha}$ in the `res$persistence_parameters` function output.

Figure 1a and 1b display the caterpillar plots for the estimated teacher effects for the continuous and the binary responses with 90% confidence intervals given by

```
> plot(res)
```

The `plot` function also displays the conditional residuals and predicted probability of a positive response by response level (omitted here).

Covariance matrices

A key feature of RealVAMS is that it estimates the relationship between the teacher effects for the two types of responses by calculating the covariance matrix for teacher j, $\hat{\mathbf{G}}_j$. In general, $\hat{\mathbf{G}}_j$ is a blockdiagonal matrix with blocks estimated for each year. The blockdiagonal components of $\mathbf{G}_j$ for the one year of data presented in the example are estimated as:

```
> res$teach.cov
```

$$\text{Teachers: } \hat{\mathbf{G}}_j = \begin{pmatrix} 264.75 & 3.3765 \\ 3.3765 & 0.0691 \end{pmatrix}.$$

In the example, the off-diagonal or $[1,2]$ component of each of the block matrices gives the covariance between the continuous response in $[1,1]$ and the binary response in $[2,2]$. The matrix above indicates that higher teacher effect estimates on the standardized assessment tend to associate with relatively higher teacher effect estimates on the probability of college entry ($r_G = 0.79$). The off-diagonal, $\hat{g}_{12} = 3.3765$, is significantly different from zero (p = 0.007), showing that there is a significant positive relationship between the teacher effects for the probability of college entry and performance on the state standardized test. The high correlation justifies the need for the multidimensional model as opposed to two separate models.

The RealVAMS function can also provide an estimate of the intra-student relationship between the continuous response and the binary response. Let $\hat{\mathbf{R}}_i$ denote the estimated error covariance matrix for student i for the continuous and binary responses.

```
> res$R_i
```

$$\text{Students: } \hat{\mathbf{R}}_i = \begin{pmatrix} 602.00 & 3.62 \\ 3.62 & 1.00 \end{pmatrix}.$$

The estimated relationship between the multidimensional teacher effects can also be compared to other variance component estimates, allowing investigation of the relative contributions of each random effect. For example, the estimated student covariance matrix, R_i, has larger variance component estimates than G_j. Fitting separate models for the responses assumes that they are independent and the correlation in both **G** and **R** is 0. One can compare the RealVAMS model to standard separate models in **lme4** or SAS PROC GLIMMIX since the RealVAMS function includes an independent.responses option. Setting independent.responses=TRUE fixes the correlations in both **R** and **G** to 0 and should lead to results that are identical to those from running separate models, allowing for comparison of the two estimation procedures.

Conclusion

RealVAMS is an R package for fitting a multivariate value-added model with a binary outcome and a normally distributed, continuous outcome. This package overcomes computational difficulties that arise when simultaneously modeling joint binary and continuous outcomes, and may be used in a variety of contexts. In particular, **RealVAMS** is useful in education to estimate the relationship between the potential teacher contributions and to allow the mixture of binary and continuous outcomes in the same model. By using models that provide a broader picture of student success, these results may help enhance the quality of educational programs.

Acknowledgments

This material is based upon work supported by the National Science Foundation under Grant # DRL-1336027 and # DRL-1336265. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the National Science Foundation. The authors would like to thank the Associate Editor for comments that led to an improved version of this manuscript.

Bibliography

American Statistical Association. *ASA Statement on Using Value-Added Models for Educational Assessment*, 2014. URL http://www.amstat.org/policy/pdfs/ASA_VAM_Statement.pdf. [p22]

D. Bates and M. Maechler. *Matrix: Sparse and Dense Matrix Classes and Methods*, 2015. URL http://CRAN.R-project.org/package=Matrix. R package version 1.2-2. [p23]

D. Bates, M. Mächler, B. M. Bolker, and S. C. Walker. Fitting linear mixed-effects models using lme4. *Journal of Statistical Software*, 67(1):1–48, 2015. URL https://doi.org/<10.18637/jss.v067.i01>. [p22]

N. E. Breslow and D. G. Clayton. Approximate inference in generalized linear mixed models. *Journal of the American Statistical Association*, 88:9–25, 1993. [p23]

J. Broatch and A. Karl. Multivariate generalized linear mixed models for joint estimation of sporting outcomes. *Italian Journal of Applied Statistics*, to appear:–, 2018. URL http://sa-ijas.stat.unipd.it. [p22]

J. E. Broatch and S. Lohr. Multidimensional assessment of value added by teachers to real-world outcomes. *Journal of Educational and Behavioral Statistics*, 37:256–277, 2012. [p22, 23]

W. J. Browne, H. Goldstein, and J. Rasbash. Multiple membership multiple classification (MMMC) models. *Statistical Modelling*, 1:103–124, 2001. [p22]

E. Demidenko. *Mixed Models: Theory and Applications*. John Wiley & Sons, Hoboken, NJ, 2004. [p24]

A. Karl, Y. Yang, and S. Lohr. Efficient maximum likelihood estimation of multiple memembership linear mixed models, with an application to educational value-added assessments. *Computational Statistics & Data Analysis*, 59:13–27, 2013. [p24, 26]

A. Karl, J. Broatch, and J. Green. *RealVAMS: Multivariate VAM Fitting*, 2015. URL http://CRAN.R-project.org/package=RealVAMS. R package version 0.3-2. [p23]

A. T. Karl. The sensitivity of college football rankings to several modeling choices. *Journal of Quantitative Analysis in Sports*, 8(3), 2012. URL https://doi.org/<10.1515/1559-0410.1471>. [p24]

D. M. Koretz. *Measuring Up: What Educational Testing Really Tells Us*. Harvard University Press, Cambridge, MA, 2008. [p24]

J. R. Lockwood, D. F. McCaffrey, L. T. Mariano, and C. Setodji. Bayesian methods for scalable multivariate value-added assessment. *Journal of Educational and Behavioral Statistics*, 32:125–150, 2007. [p22, 26]

D. F. McCaffrey, J. R. Lockwood, D. M. Koretz, and L. S. Hamilton. *Evaluating Value-Added Models for Teacher Accountability*. Rand Education, Santa Monica, CA, 2003. [p24]

C. E. McCulloch. Maximum likelihood variance components estimation for binary data. *Journal of the American Statistical Association*, 89:330–335, 1994. [p24]

S. Rabe-Hesketh and A. Skrondal. Parameterization of multivariate random effects models for categorical data. *Biometrics*, 57:1256–1264, 2001. [p24]

SAS Institute Inc. *SAS/STAT Software, Version 9.2*. Cary, NC, 2008. URL http://www.sas.com/. [p23]

R. Wolfinger and M. O'Connell. Generalized linear mixed models: A pseudo-likelihood approach. *Journal of Statistical Computation and Simulation*, 48:233–243, 1993. [p23, 24, 26]

Z. Zhang, R. M. A. Parkerand, C. M. J. Charlton, G. Leckie, and W. J. Browne. R2mlwin: A package to run MLwiN from within R. *Journal of Statistical Software*, 72(10):1–43, 2016. URL https://doi.org/<10.18637/jss.v072.i10>. [p24]

Jennifer Broatch
School of Mathematics and Natural Sciences
Arizona State University
United States
jennifer.broatch@asu.edu

Jennifer Green
Department of Mathematical Sciences
Montana State University
United States
jgreen@montana.edu

Andrew Karl
Adsurgo, LLC
United States
akarl@asu.edu

InfoTrad: An R package for estimating the probability of informed trading

by Duygu Çelik and Murat Tiniç

Abstract The purpose of this paper is to introduce the R package **InfoTrad** for estimating the probability of informed trading (PIN) initially proposed by Easley et al. (1996). PIN is a popular information asymmetry measure that proxies the proportion of informed traders in the market. This study provides a short survey on alternative estimation techniques for the PIN. There are many problems documented in the existing literature in estimating PIN. **InfoTrad** package aims to address two problems. First, the sequential trading structure proposed by Easley et al. (1996) and later extended by Easley et al. (2002) is prone to sample selection bias for stocks with large trading volumes, due to floating point exception. This problem is solved by different factorizations provided by Easley et al. (2010) (EHO factorization) and Lin and Ke (2011) (LK factorization). Second, the estimates are prone to bias due to boundary solutions. A grid-search algorithm (YZ algorithm) is proposed by Yan and Zhang (2012) to overcome the bias introduced due to boundary estimates. In recent years, clustering algorithms have become popular due to their flexibility in quickly handling large data sets. Gan et al. (2015) propose an algorithm (GAN algorithm) to estimate PIN using hierarchical agglomerative clustering which is later extended by Ersan and Alici (2016) (EA algorithm). The package **InfoTrad** offers LK and EHO factorizations given an input matrix and initial parameter vector. In addition, these factorizations can be used to estimate PIN through YZ algorithm, GAN algorithm and EA algorithm.

Introduction

The main aim of this paper is to present the **InfoTrad** package that estimates the probability of informed trading (PIN) initially proposed by Easley et al. (1996). PIN is one of the primary measures of proxy information asymmetry in the market. The structural model is driven from maximum likelihood estimation (MLE). Wide range of studies use PIN to answer questions in different fields of finance[1].

Although it is a heavily used measure in the finance literature, the development of applications that calculate PIN are quite slow. An initial attempt for R community is made by Zagaglia (2012). **FinAsym** package of Zagaglia (2012) and the **PIN** package of Zagaglia (2013) provide the trade classification algorithm of Lee and Ready (1991) which is an important tool for studies that use the TAQ database. Both packages also provide PIN estimates through `pin_likelihood()` functions. However, those estimates are prone to bias due to misspecification and other limitations. **InfoTrad** package aims to overcome such limitations and provide users with a wide range of options when estimating PIN.

Due to the popularity of the measure, problems in estimating PIN recently gained attention in the finance literature. Easley et al. (2010) indicate that for stocks with a large trading volume, it is not possible to estimate PIN due to floating-point-exception (FPE). Two different numerical factorizations are provided by Easley et al. (2010) and Lin and Ke (2011) to overcome the bias created due to FPE.

In addition, boundary solutions in estimating PIN are also shown to create bias in empirical studies. Yan and Zhang (2012) show that, independent of the type of factorization, the likelihood function can stuck at local optimum and provide biased PIN estimates. They propose an algorithm (YZ algorithm) that spans the parameter space by using 125 different initial values for the MLE problem and obtain the PIN estimate that gives the highest likelihood value with non-boundary solutions. Although YZ algorithm provides estimates with higher likelihood and guarantees obtain non-boundary solutions, the iterative structure makes this algorithm time-consuming especially for studies that use large datasets.

Considering the fact that recent studies that estimate PIN use large datasets, the effectiveness of the YZ algorithm is questioned. In recent years, clustering algorithms have become popular due to their efficiency in processing large sets of data. Gan et al. (2015) propose an algorithm that use hierarchical agglomerative clustering to estimate PIN. Ersan and Alici (2016) later extends this framework.

FPE and boundary solutions are *not* the only problems of PIN model. Duarte and Young (2009) indicate that the structural model of Easley et al. (1996) enforces a negative contemporaneous covariance between intraday buy and sell orders, which is contrary to the empirical evidence for symmetric order shocks. In addition,they show that the PIN model fails to capture the volatility of buy and sell orders,through simulations. Moreover, Duarte and Young (2009) adjust PIN to take into account the liquidity impact and show that liquidity is more prominent on stock returns compared to information

[1]For instance, analyst coverage (Easley et al., 1998), stock splits (Easley et al., 2001), initial public offerings (Ellul and Pagano, 2006), credit ratings (Odders-White and Ready, 2006), M&A announcements (Aktas et al., 2007) and asset returns [(Easley et al., 2002),(Easley et al., 2010)] among others.

asymmetry. Finally, it is important to note that PIN does not consider any strategic behaviour of investors such as order splitting. Order splitting can be more evident when a stock is jointly trading on multiple venues (Menkveld, 2008). Even for a stock that is traded on a single market, an informed investor may want to split her order in order avoid revealing her private information too quickly (Foucault et al., 2013). PIN model, by construction, fails to attach multiple small orders to a single informed investor.

This paper introduces and discusses the R (R Core Team, 2016) **InfoTrad** package for estimating PIN. **InfoTrad** provides users with the necessary methods to *solely* adress the problems of FPE and boundary solutions. The package contains the likelihood factorizations of EHO and LK as separate functions (`EHO()` and `LK()`, respectively) which provide likelihood specifications to avoid FPE. In addition, through `YZ()`, `GAN()` and `EA()` functions, PIN estimates can be obtained using the grid-search algorithm of Yan and Zhang (2012) and clustering algorithms of Gan et al. (2015) and Ersan and Alici (2016). For all of the algorithms, likelihood specification can be set to EHO or LK.

The paper is organized as follows; Section 8.2 provides a brief description of PIN. Specifically, section 8.2.1 discusses the problem of FPE and the alternative factorizations EHO and LK. Section 8.2.2 reviews the problem of boundary solutions and the YZ algorithm. Section 8.2.3 describes the clustering algorithms of Gan et al. (2015) and Ersan and Alici (2016). Section 8.3 introduces the package **InfoTrad** along with examples. Section 8.4 evaluates the performance of each method through simulations. Section 8.5 provides concluding remarks.

PIN Model

The structural model of Easley et al. (1996) and Easley et al. (2002) consists of three types of agents; informed traders, uninformed traders and market makers. On a trading day t, one risky asset is continuously traded. Market maker sets the price for a given stock by observing the buy orders (B_t) and sell orders (S_t). For that stock, an information event is assumed to follow a Bernoulli distribution with success probability α. This event reveals either a high or a low signal for the stock value. The event is assumed to provide a low signal with probability δ. When informed traders observe a high (low) signal, they are assumed to place buy (sell) orders at a rate of μ. Uninformed traders are assumed to place orders, independent of the information event and the signal. They arrive to market to place a buy (sell) order at a rate of ϵ_b (ϵ_s). Orders of both informed and uninformed investors are assumed to follow independent Poisson processes.

The joint probability distribution with respect to the parameter vector $\Theta \equiv \{\alpha, \delta, \mu, \epsilon_b, \epsilon_s\}$ and the number of buys and sells (B_t, S_t), is specified by

$$\begin{aligned} f(B_t, S_t|\Theta) \equiv\ & \alpha\delta exp(-\epsilon_b)\frac{\epsilon_b^{B_t}}{B_t!}exp[-(\epsilon_s+\mu)]\frac{(\epsilon_s+\mu)^{S_t}}{S_t!} \\ & +\alpha(1-\delta)exp[-(\epsilon_b+\mu)]\frac{(\epsilon_b+\mu)^{B_t}}{B_t!}exp(-\epsilon_s)\frac{\epsilon_s^{S_t}}{S_t!} \\ & +(1-\alpha)exp(-\epsilon_b)\frac{\epsilon_b^{B_t}}{B_t!}exp(-\epsilon_s)\frac{\epsilon_s^{S_t}}{S_t!} \end{aligned} \tag{1}$$

The estimates of arrival rates ($\hat{\mu}, \hat{\epsilon_s}$ and $\hat{\epsilon_b}$), along with estimates of the probabilities ($\hat{\alpha}$ and $\hat{\delta}$) can be obtained by maximizing the joint log-likelihood function given the order input matrix (B_t, S_t) over T trading days. The non-linear objective function of this problem can be written as;

$$L(\Theta|T) \equiv \sum_{t=1}^{T} L(\Theta|(B_t, S_t)) = \sum_{t=1}^{T} log[f(B_t, S_t|\Theta)] \tag{2}$$

The maximization problem is subject to the boundary constraints $\alpha, \delta \in [0,1]$ and $\mu, \epsilon_b, \epsilon_s \in [0, \infty)$[2]. The PIN estimate is then given by;

$$\widehat{PIN} = \frac{\hat{\alpha}\hat{\mu}}{\hat{\alpha}\hat{\mu} + \hat{\epsilon_b} + \hat{\epsilon_s}} \tag{3}$$

[2]Both **PIN** package of Zagaglia (2013) and **FinAsym** package of Zagaglia (2012) fail to acknowledge the boundary constraints on arrival rates $\mu, \epsilon_b, \epsilon_s$. Similar to event probabilities, they restrict these parameters to $[0,1]$ which forces the estimates for the arrival of informed and uninformed traders on a given day to take values at most one. This creates significant bias in PIN estimates.

Floating-Point Exception

PIN estimates are prone to selection bias, especially for stocks for which the number of buy and sell orders are large[3]. Lin and Ke (2011) show that the increase in the number of buy and sell orders for a given stock, significantly shrinks the feasible solution set for the maximization of the log likelihood function in equation (2). To maximize the non-linear function (1), the optimization software introduces initial values for the parameters in Θ. The numerical optimization method is applied after those initial parameters are introduced. Therefore, for large enough B_t and S_t whose factorials cannot be calculated by mainstream computers (i.e. FPE), the optimal value for equation (2) becomes undefined. The FPE problem is therefore, more pronounced in active stocks.

To avoid the bias created due to FPE, one factorization of the equation (2) is provided by Easley et al. (2010) as $L_{EHO}(\Theta|T) \equiv \sum_{t=1}^{T} L_{EHO}(\Theta|B_t, S_t)$ where

$$\begin{aligned} L_{EHO}(\Theta|B_t, S_t) = log[\alpha\delta exp(-\mu)x_b^{B_t-M_t}x_s^{-M_t} + \alpha(1-\delta)exp(-\mu)x_b^{-M_t}x_s^{S_t-M_t} + (1-\alpha)x_b^{B_t-M_t}x_s^{S_t-M_t}] \\ + B_t log(\epsilon_b+\mu) + S_t log(\epsilon_s+\mu) - (\epsilon_b+\epsilon_s) + M_t[log(x_b)+log(x_s)] - log(S_t!B_t!), \end{aligned} \tag{4}$$

where $M_t = min(B_t, S_t) + max(B_t, S_t)/2$, $x_b = \epsilon_b/(\mu+\epsilon_b)$ and $x_s = \epsilon_s/(\mu+\epsilon_s)$.

Lin and Ke (2011) introduce another algebraically equivalent factorization of the equation (2), $L_{LK}(\Theta|T) \equiv \sum_{t=1}^{T} L_{LK}(\Theta|B_t, S_t)$ where

$$\begin{aligned} L_{LK}(\Theta|B_t, S_t) = log[\alpha\delta exp(e_{1t}-e_{maxt}) + \alpha(1-\delta)exp(e_{2t}-e_{maxt}) + (1-\alpha)exp(e_{3t}-e_{maxt})] \\ + B_t log(\epsilon_b+\mu) + S_t log(\epsilon_s+\mu) - (\epsilon_b+\epsilon_s) + e_{maxt} - log(S_t!B_t!), \end{aligned} \tag{5}$$

where $e_{1t} = -\mu - B_t log(1+\mu/\epsilon_b)$, $e_{2t} = -\mu - S_t log(1+\mu/\epsilon_s)$, $e_{3t} = -B_t log(1+\mu/\epsilon_b) - S_t log(1+\mu/\epsilon_s)$ and $e_{maxt} = max(e_{1t}, e_{2t}, e_{3t})$. The last term $log(S_t!B_t!)$ is constant with respect to the parameter vector Θ, and is, therefore, dropped in the MLE for both factorizations.

Boundary Solutions

Another source of bias in estimating PIN arises from boundary solutions. Yan and Zhang (2012) indicate that in calculating PIN, parameter estimates $\hat{\alpha}$ and $\hat{\delta}$ usually fall onto the boundaries of the parameter space, that is, they are equal to zero or one. PIN estimate presented in equation (3) is directly related to the estimate of $\hat{\alpha}$. Letting $\hat{\alpha}$ equal to zero will make sure that PIN is zero as well. This can create a sample selection bias in portfolio formation, especially for quarterly estimations[4]. Yan and Zhang (2012) show that;

$$E(B) = \alpha(1-\delta)\mu + \epsilon_b \tag{6}$$

$$E(S) = \alpha\delta\mu + \epsilon_s \tag{7}$$

Then, they propose the following algorithm to overcome the bias created due to boundary solutions. Let $(\alpha^0, \delta^0, \epsilon_b^0, \epsilon_s^0, \mu^0)$ be the initial parameter function to be placed in the non-linear program presented in equation (4). In addition, let $\bar{B}$ and $\bar{S}$ be the average number of buy and sell orders.

$$\alpha^0 = \alpha_i, \quad \delta^0 = \delta_j, \quad \epsilon_b^0 = \gamma_k\bar{B}, \quad \mu^0 = \frac{\bar{B}-\epsilon_b^0}{\alpha^0(1-\delta^0)} \text{ and } \epsilon_s^0 = \bar{S} - \alpha^0\delta^0\mu^0 \tag{8}$$

where $\alpha_i, \delta_j, \gamma_k \in \{0.1, 0.3, 0.5, 0.7, 0.9\}$. This will yield 125 different PIN estimates along with their likelihood values. In line with Yan and Zhang (2012), we drop any initial parameter vector having negative values for ϵ_s^0. In addition, following Ersan and Alici (2016), we also drop any initial parameter vector with $\mu^0 > max(B_t, S_t)$. Yan and Zhang (2012) then select the estimate with non-boundary parameters yielding highest likelihood value. This method, by construction, spans the parameter space and tries to avoid local optima and provides non-boundary estimates for α.

[3]For example, Zagaglia (2012) provides a sample data to calculate PIN. In sample data the maximum trade number is 19. If you multiply each observation in the sample data by 10, the `pin_likelihood()` function of **FinAsym** package fails to provide results with the sample initial parameter vector.

[4]For quarterly estimations of PIN, one can be sure that there is at least one information event, earnings announcement. Therefore $\hat{\alpha}$ cannot be equal to zero.

Clustering Approach

In recent years, clustering algorithms are increasingly becoming popular in estimating the probability of informed trading due to efficiency concerns. Gan et al. (2015) and Ersan and Alici (2016) use clustering algorithms to estimate PIN. Gan et al. (2015) introduce a method that clusters the data into three groups (good news, bad news, no news) based on the mean absolute difference in order imbalance. Let $X_t = B_t - S_t$ be the order imbalance on day t computed as the difference between buy orders and sell orders. The clustering is then based on the distance function defined as $D(I,J) = |X_i - X_j|, \quad 1 \leq i,j \leq T$ where $i \neq j$. They use hierarchical agglomerative clustering (HAC) to group the data elements based on the distance matrix. Specifically, they use `hclust()` function of Müllner (2013) in R[5]. The algorithm sequentially clusters, in a bottom-up fashion, each observation into groups based on X_t and stops when it reaches three clusters. The theoretical framework of Easley et al. (1996) indicates that a stock has high (low) X_t on good (bad) days. Therefore, the cluster which has the highest (lowest) mean X_t is labelled as good (bad) news. The remaining cluster is then labelled as no news. Once each observation is grouped into their respective clusters (good news, bad news, no news), $c \in \{G,B,N\}$, the parameter estimates for $\Theta \equiv \{\alpha,\delta,\mu,\epsilon_b,\epsilon_s\}$ are calculated simply by counting. Let ω_c be the proportion of cluster c occupying the total number of days T, such that $\sum_{c=1}^{3}\omega_c = 1$. Similarly, let $\bar{B_c}$ and $\bar{S_c}$ be the average number of buys and sells on cluster c, respectively.

Then, the probability of an information event is given by $\hat{\alpha} = \omega_B + \omega_G$. Moreover, the estimate for the probability of information event releasing bad news is given by $\hat{\delta} = \omega_B/\hat{\alpha}$. The estimate for the arrival rate of buy orders of uninformed traders represented by $\hat{\epsilon_b} = \frac{\omega_B}{\omega_B+\omega_N}\bar{B_B} + \frac{\omega_N}{\omega_B+\omega_N}\bar{B_N}$. Similarly, the estimate for the arrival rate of sell orders of uninformed traders represented by $\hat{\epsilon_s} = \frac{\omega_G}{\omega_G+\omega_N}\bar{S_G} + \frac{\omega_N}{\omega_G+\omega_N}\bar{B_N}$. Finally, the arrival rate for the informed investors is calculated as $\hat{\mu} = \frac{\omega_G}{\omega_B+\omega_G}(\bar{B_G} - \hat{\epsilon_b}) + \frac{\omega_B}{\omega_B+\omega_G}(\bar{S_B} - \hat{\epsilon_s})$ where $(\bar{B_G} - \hat{\epsilon_b})$ corresponds to the buy rate of informed investors $\hat{\mu_b}$ and $(\bar{S_B} - \hat{\epsilon_s})$ corresponds to the sell rate of informed investors $\hat{\mu_s}$[6].

Through simulations, Gan et al. (2015) show that estimates calculated as above are proper candidates for the initial parameter values to be used in MLE process. Ersan and Alici (2016) argue that the estimates for the informed arrival rate, μ, contains a downward bias with GAN algorithm[7]. This is what we observe in this study as well. In addition, they state that GAN algorithm provides inaccurate estimates for δ. In order to overcome these issues, instead of using X_t, Ersan and Alici use absolute daily order imbalance, $|X_t|$, to cluster the data. They initially cluster, $|X_t|$ into two, again by using hclust(). The cluster with the lower mean daily absolute order imbalance is labelled as "no event" cluster and the remaining as "event" cluster. Then, the formation of "good" and "bad" event day clusters are obtained through separating the days in the "event" cluster into two with respect to the *sign* of the daily order imbalances. The parameter estimates are then computed with the same procedure presented above[8].

The InfoTrad Package

The R package **InfoTrad** provides five different functions `EHO()`,`LK()`,`YZ()`,`GAN()` and `EA()`. The first two functions provide likelihood specifications whereas the last three functions can be used to obtain parameter estimates for Θ to calculate PIN in equation (3). All five functions require a data frame that contains B_t in the first column, and S_t in the second column. We create B_t and S_t for ten hypothetical trading days[9]. `EHO()` and `LK()` read (B_t, S_t) and return the related functional form of the negative log likelihood. These objects can be used in any optimization procedure such as `optim()` to obtain the parameter estimates $\hat{\Theta} \equiv \{\hat{\alpha},\hat{\delta},\hat{\mu},\hat{\epsilon_b},\hat{\epsilon_s}\}$, the likelihood value and other specifications, in one iteration with a pre-specified initial value vector, Θ_0, for parameters. We define `EHO()` and `LK()` as simple likelihood specifications rather than functions that execute the MLE procedure. This is due to the fact that MLE estimators vary depending on the optimization procedure. Users who wish to develop alternative estimation techniques, based on the proposed likelihood factorization, can use `EHO()` and `LK()`. This is the underlying reason why those functions do not have built-in optimization procedures.

[5] `hclust()` function is used at its default setting in line with Gan et al. (2015).

[6] Both Gan et al. (2015) and Ersan and Alici (2016) do not mention the case where $\hat{\mu_b} < 0$ or $\hat{\mu_s} < 0$. It is fair to assume that in such cases, informed investors are not present on the buy (sell) side. Therefore, we set μ_b and μ_s equal to zero when we obtain a negative estimate.

[7] We also show that estimates for μ contains a significant downward bias due to poor choice of initial parameter value μ_0 when GAN algorithm is used.

[8] Ersan and Alici (2016) also provide an iterative process in which they systematically update the clusters. We plan to introduce this methodology in the future versions of our package.

[9] The numbers are randomly selected. We set numbers to be high enough so that the original likelihood framework presented in equation (1) cannot be used due to FPE. Easley et al. (1996) indicate that at least 60 days worth of data is required in order to obtain proper convergence for $\widehat{PIN}$. We use ten days for demonstration purposes.

By specifying `EHO()` and `LK()` as simple likelihood functions, we give developers the flexibility to select the most suitable optimization procedure for their application.

For researchers who want to calculate an estimate of PIN, `YZ()`, `GAN()` and `EA()` functions have built-in optimization procedures. Those functions read a likelihood specification value along with data. Likelihood specification can be set either to "LK" or to "EHO" with "LK" being the default. All estimation functions use `neldermead()` function of **nloptr** package to conduct MLE with the specified factorization. GAN and EA functions also use `hclust()` function of Müllner (2013) to conduct clustering. The output of these three functions is an object that provides $\{\hat{\alpha}, \hat{\delta}, \hat{\mu}, \hat{\epsilon_b}, \hat{\epsilon_s}, f(\hat{\Theta}), \widehat{PIN}\}$, where $f(\hat{\Theta})$ represents the optimal likelihood value given the parameter estimates $\hat{\Theta}$.

EHO() function

An example is provided below for `EHO()` with a sample data and initial parameter values. Notice that the first column of sample data is for B_t and second column is for S_t. Similarly, the initial parameter values are constructed as; $\Theta_0 = \{\alpha, \delta, \mu, \epsilon_b, \epsilon_s\}$. We use `optim()` with 'Nelder-Mead' method to execute MLE, however *developer* is flexible to use other methods as well.

```
library(InfoTrad)
# Sample Data
#   Buy Sell
#1  350  382
#2  250  500
#3  500  463
#4  552  550
#5  163  200
#6  345  323
#7  847  456
#8  923  342
#9  123  578
#10 349  455

Buy<-c(350,250,500,552,163,345,847,923,123,349)
Sell<-c(382,500,463,550,200,323,456,342,578,455)
data=cbind(Buy,Sell)

# Initial parameter values
# par0 = (alpha, delta, mu, epsilon_b, epsilon_s)
par0 = c(0.5,0.5,300,400,500)

# Call EHO function
EHO_out = EHO(data)
model = optim(par0, EHO_out, gr = NULL, method = c("Nelder-Mead"), hessian = FALSE)

## Parameter Estimates
model$par[1] # Estimate for alpha
# [1] 0.9111102
model$par[2] # Estimate for delta
#[1] 0.0001231429
model$par[3] # Estimate for mu
# [1] 417.1497
model$par[4] # Estimate for eb
# [1] 336.075
model$par[5] # Estimate for es
# [1] 466.2539

## Estimate for PIN
(model$par[1]*model$par[3])/((model$par[1]*model$par[3])+model$par[4]+model$par[5])
# [1] 0.3214394
####
```

In this example, B_t and S_t vectors are selected so that the likelihood function cannot be represented as in equation (1). We set the initial parameters to be Θ_0=(0.5,0.5,300,400,500). For the given B_t, S_t and Θ_0 vectors, PIN measure is calculated as 0.32 with EHO factorization.

LK() function

An example is provided below for `LK()` function with a sample data and initial parameter values. Notice that the first column of sample data is for B_t and second column is for S_t. Similarly, the initial parameter values are constructed as; $\Theta_0 = \{\alpha, \delta, \mu, \epsilon_b, \epsilon_s\}$. We use `optim()` with 'Nelder-Mead' method to execute MLE, however *developer* is flexible to use other methods as well.

```
library(InfoTrad)
# Sample Data
#   Buy Sell
#1  350  382
#2  250  500
#3  500  463
#4  552  550
#5  163  200
#6  345  323
#7  847  456
#8  923  342
#9  123  578
#10 349  455

Buy<-c(350,250,500,552,163,345,847,923,123,349)
Sell<-c(382,500,463,550,200,323,456,342,578,455)
data=cbind(Buy,Sell)

# Initial parameter values
# par0 = (alpha, delta, mu, epsilon_b, epsilon_s)
par0 = c(0.5,0.5,300,400,500)

# Call LK function
LK_out = LK(data)
model = optim(par0, LK_out, gr = NULL, method = c("Nelder-Mead"), hessian = FALSE)

## The structure of the model output ##
model

#$par
#[1]   0.480277   0.830850 315.259805 296.862318 400.490830

#$value
#[1] -44343.21

#$counts
#function gradient
#    502       NA

#$convergence
#[1] 1

#$message
#NULL

## Parameter Estimates
model$par[1] # Estimate for alpha
# [1] 0.480277
model$par[2] # Estimate for delta
# [1] 0.830850
model$par[3] # Estimate for mu
# [1] 315.259805
model$par[4] # Estimate for eb
# [1] 296.862318
model$par[5] # Estimate for es
# [1] 400.4908

## Estimate for PIN
```

```
(model$par[1]*model$par[3])/((model$par[1]*model$par[3])+model$par[4]+model$par[5])
# [1] 0.178391
####
```

For the given B_t, S_t and Θ_0 vectors, PIN measure is calculated as 0.18 with LK factorization.

YZ() function

An example is provided below for `YZ()` function with a sample data. Notice that the first column of sample data is for B_t and second column is for S_t. In addition, the first example is with default likelihood specification LK and the second one is with EHO. Notice that `YZ()` function do not require any initial parameter vector Θ_0.

```
library(InfoTrad)
# Sample Data
#   Buy Sell
#1  350  382
#2  250  500
#3  500  463
#4  552  550
#5  163  200
#6  345  323
#7  847  456
#8  923  342
#9  123  578
#10 349  455

Buy<-c(350,250,500,552,163,345,847,923,123,349)
Sell<-c(382,500,463,550,200,323,456,342,578,455)
data<-cbind(Buy,Sell)

# Parameter estimates using the LK factorization of Lin and Ke (2011)
# with the algorithm of Yan and Zhang (2012).
# Default factorization is set to be "LK"

result=YZ(data)
print(result)

# Alpha: 0.3999999
# Delta: 0
# Mu: 442.1667
# Epsilon_b: 263.3333
# Epsilon_s: 424.9
# Likelihood Value: 44371.84
# PIN: 0.2004457

# Parameter estimates using the EHO factorization of Easley et. al. (2010)
# with the algorithm of Yan and Zhang (2012).

result=YZ(data,likelihood="EHO")
print(result)

# Alpha: 0.9000001
# Delta: 0.9000001
# Mu: 489.1111
# Epsilon_b: 396.1803
# Epsilon_s: 28.72002
# Likelihood Value: Inf
# PIN: 0.3321033
```

For the given B_t and S_t vectors, PIN measure is calculated as 0.20 with YZ algorithm along with LK factorization. Moreover, PIN measure is calculated as 0.33 with YZ algorithm along with EHO factorization.

GAN() function

An example is provided below for `GAN()` function with a sample data. Notice that the first column of sample data is for B_t and second column is for S_t. In addition, the first example is with default likelihood specification LK and the second one is with EHO. Notice that `GAN()` function do not require any initial parameter vector Θ_0.

```
library(InfoTrad)
# Sample Data
#   Buy Sell
#1  350  382
#2  250  500
#3  500  463
#4  552  550
#5  163  200
#6  345  323
#7  847  456
#8  923  342
#9  123  578
#10 349  455

Buy<-c(350,250,500,552,163,345,847,923,123,349)
Sell<-c(382,500,463,550,200,323,456,342,578,455)
data<-cbind(Buy,Sell)

# Parameter estimates using the LK factorization of Lin and Ke (2011)
# with the algorithm of Gan et. al. (2015).
# Default factorization is set to be "LK"

result=GAN(data)
print(result)

# Alpha: 0.3999998
# Delta: 0
# Mu: 442.1667
# Epsilon_b: 263.3333
# Epsilon_s: 424.9
# Likelihood Value: 44371.84
# PIN: 0.2044464

# Parameter estimates using the EHO factorization of Easley et. al. (2010)
# with the algorithm of Gan et. al. (2015)

result=GAN(data, likelihood="EHO")
print(result)

# Alpha: 0.3230001
# Delta: 0.4780001
# Mu: 481.3526
# Epsilon_b: 356.6359
# Epsilon_s: 313.136
# Likelihood Value: Inf
# PIN: 0.1884001
```

For the given B_t and S_t vectors, PIN measure is calculated as 0.20 with GAN algorithm along with LK factorization. Moreover, PIN measure is calculated as 0.19 with GAN algorithm along with EHO factorization.

EA() function

An example is provided below for `EA()` function with a sample data. Notice that the first column of sample data is for B_t and second column is for S_t. In addition, the first example is with default likelihood specification LK and the second one is with EHO. Notice that `EA()` function do not require

any initial parameter vector Θ_0.

```
library(InfoTrad)
# Sample Data
#   Buy Sell
#1  350  382
#2  250  500
#3  500  463
#4  552  550
#5  163  200
#6  345  323
#7  847  456
#8  923  342
#9  123  578
#10 349  455

Buy=c(350,250,500,552,163,345,847,923,123,349)
Sell=c(382,500,463,550,200,323,456,342,578,455)
data=cbind(Buy,Sell)

# Parameter estimates using the LK factorization of Lin and Ke (2011)
# with the modified clustering algorithm of Ersan and Alici (2016).
# Default factorization is set to be "LK"

result=EA(data)
print(result)

# Alpha: 0.9511418
# Delta: 0.2694005
# Mu: 76.7224
# Epsilon_b: 493.7045
# Epsilon_s: 377.4877
# Likelihood Value: 43973.71
# PIN: 0.07728924


# Parameter estimates using the EHO factorization of Easley et. al. (2010)
# with the modified clustering algorithm of Ersan and Alici (2016).

result=EA(data,likelihood="EHO")
print(result)

# Alpha: 0.9511418
# Delta: 0.2694005
# Mu: 76.7224
# Epsilon_b: 493.7045
# Epsilon_s: 377.4877
# Likelihood Value: 43973.71
# PIN: 0.07728924
```

For the given B_t and S_t vectors, PIN measure is calculated as 0.08 with EA algorithm along with LK factorization. Moreover, PIN measure is calculated, again, as 0.08 with EA algorithm along with EHO factorization.

Simulations and Performance Evaluation

In this section, we investigate the performance of the estimates obtained for Θ and PIN using the existing methods. We evaluate the methods based on their accuracy proxied by mean absolute errors (MAE)[10]. We first examine how the estimates vary in different trade intensity levels. To this end, we follow the methodology in Gan et al. (2015). Let I be the the set of trade intensity levels ranging from 50 to 5000 at step size of 50, that is, I=$\{50, 100, 150, \ldots, 5000\}$. We first set our parameters as

[10] All estimations are conducted on a 2.6 Intel i7-6700HQ CPU.We do not consider speed as a performance measure since the average processing time for each method is less than 10 seconds.

$\Theta = \{\alpha = 0.5, \delta = 0.5, \mu = 0.2i, \epsilon_b = 0.4i, \epsilon_s = 0.4i\}$, where $i \in I$. For each trade intensity level, we generate N=50 random samples of $\tilde{\alpha}$ and $\tilde{\delta}$ that are binomially distributed with parameters α and δ respectively. $\tilde{\alpha}$ and $\tilde{\delta}$ proxy the content of the information event. For each pair of $\tilde{\alpha}$, $\tilde{\delta}$ values, we generate buy and sell values (B_t, S_t) for hypothetical T=60 days in the following manner;

- if $\tilde{\alpha}$ = 0, then there is no information event, therefore, generate $B_t \sim Pois(\epsilon_b)$ and $S_t \sim Pois(\epsilon_s)$.
- if $\tilde{\alpha}$ = 1, and $\tilde{\delta}$ =1, then there is bad news, therefore generate $B_t \sim Pois(\epsilon_b)$ and $S_t \sim Pois(\epsilon_s + \mu)$
- if $\tilde{\alpha}$ = 1, and $\tilde{\delta}$ =0, then there is good news, therefore generate $B_t \sim Pois(\epsilon_b + \mu)$ and $S_t \sim Pois(\epsilon_s)$

We then form the joint likelihood function represented by equation (4) in EHO form or by equation (5) in LK form and obtain the estimates using `YZ()`, `GAN()` or `EA()` methods.

The results are presented in Table 1 which indicates that `YZ()` method with `LK()` factorization provides the PIN estimates with lowest MAE. Although the clustering algorithms, especially `GAN()` method, provide powerful estimates of $\hat{\alpha}, \hat{\delta}, \hat{\epsilon_b}, \hat{\epsilon_s}$, they fail to estimate the arrival rate of informed investors $\hat{\mu}$,accurately. This is in line with Ersan and Alici (2016). On the contrary, `YZ()` method with `EHO()` factorization provides the best estimates for $\hat{\mu}$, but fails to provide good estimates for other parameters.

Method	Factorization	$\widehat{PIN}$	$\hat{\alpha}$	$\hat{\delta}$	$\hat{\mu}$	$\hat{\epsilon_b}$	$\hat{\epsilon_s}$
YZ	**LK**	0.075	0.199	0.059	415.2	104.3	109.0
YZ	**EHO**	0.134	0.428	0.310	154.6	288.3	247.4
GAN	**EHO**	0.101	0.087	0.083	479.4	124.1	117.3
GAN	**LK**	0.101	0.087	0.083	479.5	123.8	118.1
EA	**LK**	0.102	0.268	0.274	484.6	128.7	119.3
EA	**EHO**	0.102	0.270	0.275	483.1	128.5	107.8

Table 1: This table represents the mean absolute errors (MAE) of the parameter estimates obtained by a given method for a given factorization. Each row represents a different method with a different factorization. First two column represent the specification of method and factorization respectively. The last six columns represents the power of estimates of PIN along with the parameter space $\Theta \equiv \{\alpha, \delta, \mu, \epsilon_b, \epsilon_s\}$. MAE measures for the estimates calculated as $\sum_{i=1}^{N} \frac{|\widehat{\Theta}_i - \Theta_i^{TR}|}{N}$ where $\widehat{\Theta}$ represent the estimates and Θ^{TR} represents the true value.

A more general way of examining the accuracy of PIN estimates is proposed in several studies (e.g, Lin and Ke (2011), Gan et al. (2015), Ersan and Alici (2016)). In this setting, we fix the trade intensity, I=2500. The total trade intensity represents the overall presence of informed and uninformed traders, that is, I=(μ, ϵ_b, ϵ_s). We then generate three probability terms p_1, p_2, p_3 with N=5000 random observations that are distributed uniformly between 0 and 1. p_1 represents the fraction of informed investors in total trade intensity, that is, μ=$p_1 * I$. The rest of the trade intensity is distributed equally to buy and sell orders of uninformed investors, that is, $e_b = e_s = (1 - p_1) * I/2$. p_2 represents the true parameter for the probability of news arrival, α, and p_3 is the true parameter for the content of the news, δ. We generate observations for $\tilde{\alpha}$ and $\tilde{\delta}$, as described earlier. For each pair of $\tilde{\alpha}$ and $\tilde{\delta}$, we generate buy and sell values (B_t, S_t) for hypothetical T=60 days, again, in the manner presented above; form the likelihood and obtain the parameter estimates.

The results are presented in Table 2. Similar to first simulation, `GAN()` captures the true nature of $\hat{\alpha}$ and $\hat{\delta}$ better than any other method with both factorizations. `YZ()` method with `EHO()` factorization performs best when estimating the arrival of informed traders, $\hat{\mu}$. The importance of estimating $\hat{\mu}$ becomes quite evident in Table 2. Although other methods outperform `YZ()` method with `EHO()` factorization in estimating α, ϵ_b and ϵ_s, it provides the best estimate for PIN due to it's performance on estimating $\hat{\mu}$.

Summary

This paper provides a short survey on five most widely used estimation techniques for the probability of informed trading (PIN) measure. In this paper, we introduce the R package **InfoTrad**, covering estimation procedures for PIN using EHO, LK factorizations along with YZ, GAN and EA algorithms (`EHO()`,`LK()`, `YZ()`, `GAN()` `EA()`). The functions `EHO()` and `LK()` read a (Tx2) matrix where the rows of the first column contains total number of buy orders on a given trading day t, B_t, and the rows

Method	Factorization	$\widehat{PIN}$	$\hat{\alpha}$	$\hat{\delta}$	$\hat{\mu}$	$\hat{\epsilon_b}$	$\hat{\epsilon_s}$
YZ	**LK**	0.323	0.428	0.432	1,212.0	303.4	325.0
YZ	**EHO**	0.237	0.437	0.357	942.9	386.0	470.2
GAN	**LK**	0.348	0.380	0.410	1,218.7	314.5	323.3
GAN	**EHO**	0.347	0.357	0.397	1,216.2	328.5	339.5
EA	**LK**	0.348	0.437	0.421	1,224.0	325.1	336.3
EA	**EHO**	0.347	0.428	0.413	1,222.0	331.3	345.9

Table 2: This table represents the mean absolute errors (MAE) of the parameter estimates obtained by a given method for a given factorization. Each row represents a different method with a different factorization. First two column represent the specification of method and factorization respectively. The last six columns represents the power of estimates of PIN along with the parameter space $\Theta \equiv \{\alpha, \delta, \mu, \epsilon_b, \epsilon_s\}$. MAE measures for the estimates calculated as $\sum_{i=1}^{N} \frac{|\widehat{\Theta}_i - \Theta_i^{TR}|}{N}$ where $\widehat{\Theta}$ represent the estimates and Θ^{TR} represents the true value.

of the second column contains the total number of sell orders on a given trading day t, S_t, where t $\in \{1, 2, \ldots, T\}$. In addition, they also require an initial parameter vector in the form of, $\Theta_0 = \{\alpha, \delta, \mu, \epsilon_b, \epsilon_s\}$. Both functions produce the respective log-likelihood functions.

The functions `YZ()`, `GAN()` and `EA()` read (B_t, S_t) as an input along with a likelihood specification that is set to '`LK`' by default. These functions do not require initial parameter matrix to obtain the parameter estimates when calculating PIN. All three functions use `neldermead()` method of **nloptr** as built-in optimization procedure for MLE. `YZ()` `GAN()` and `EA()` produce an object that gives the parameter estimates $\hat{\Theta}$ along with likelihood value and $\widehat{PIN}$.

Acknowledgments

This research is supported by the Scientific and Technological Research Council of Turkey (TUBITAK), Grant Number: 116K335.

Bibliography

N. Aktas, E. De Bodt, F. Declerck, and H. Van Oppens. The PIN anomaly around *m&a* announcements. *Journal of Financial Markets*, 10(2):169–191, 2007. URL https://doi.org/10.1016/j.finmar.2006.09.003. [p31]

J. Duarte and L. Young. Why is PIN priced? *Journal of Financial Economics*, 91(2):119–138, 2009. URL https://doi.org/10.1016/j.jfineco.2007.10.008. [p31]

D. Easley, N. M. Kiefer, M. O'Hara, and J. B. Paperman. Liquidity, information, and infrequently traded stocks. *The Journal of Finance*, 51(4):1405–1436, 1996. URL https://doi.org/10.1111/j.1540-6261.1996.tb04074.x. [p31, 32, 34]

D. Easley, M. O'Hara, and J. Paperman. Financial analysts and information-based trade. *Journal of Financial Markets*, 1(2):175–201, 1998. URL https://doi.org/10.1016/s1386-4181(98)00002-0. [p31]

D. Easley, M. O'Hara, and G. Saar. How stock splits affect trading: A microstructure approach. *Journal of Financial and Quantitative Analysis*, 36(01):25–51, 2001. URL https://doi.org/10.2307/2676196. [p31]

D. Easley, S. Hvidkjaer, and M. O'Hara. Is information risk a determinant of asset returns? *The Journal of Finance*, 57(5):2185–2221, 2002. URL https://doi.org/10.1111/1540-6261.00493. [p31, 32]

D. Easley, S. Hvidkjaer, and M. O'Hara. Factoring information into returns. *Journal of Financial and Quantitative Analysis*, 2010. URL https://doi.org/10.1017/s0022109010000074. [p31, 33]

A. Ellul and M. Pagano. IPO underpricing and after-market liquidity. *Review of Financial Studies*, 19(2):381–421, 2006. URL https://doi.org/10.1093/rfs/hhj018. [p31]

O. Ersan and A. Alici. An unbiased computation methodology for estimating the probability of informed trading (PIN). *Journal of International Financial Markets, Institutions and Money*, 43:74–94, 2016. URL https://doi.org/10.1016/j.intfin.2016.04.001. [p31, 32, 33, 34, 40]

T. Foucault, M. Pagano, and A. Röell. *Market Liquidity: Theory, Evidence, and Policy*. Oxford University Press, 2013. URL https://doi.org/10.1093/acprof:oso/9780199936243.001.0001. [p32]

Q. Gan, W. C. Wei, and D. Johnstone. A faster estimation method for the probability of informed trading using hierarchical agglomerative clustering. *Quantitative Finance*, 15(11):1805–1821, 2015. URL https://doi.org/10.1080/14697688.2015.1023336. [p31, 32, 34, 39, 40]

C. Lee and M. J. Ready. Inferring trade direction from intraday data. *The Journal of Finance*, 46(2): 733–746, 1991. URL https://doi.org/10.1111/j.1540-6261.1991.tb02683.x. [p31]

H.-W. W. Lin and W.-C. Ke. A computing bias in estimating the probability of informed trading. *Journal of Financial Markets*, 14(4):625–640, 2011. URL https://doi.org/10.1016/j.finmar.2011.03.001. [p31, 33, 40]

A. J. Menkveld. Splitting orders in overlapping markets: A study of cross-listed stocks. *Journal of Financial Intermediation*, 17(2):145–174, 2008. URL https://doi.org/10.1016/j.jfi.2007.05.004. [p32]

D. Müllner. Fastcluster: Fast hierarchical, agglomerative clustering routines for r and python. *Journal of Statistical Software*, 53(9):1–18, 2013. URL https://doi.org/10.18637/jss.v053.i09. [p34, 35]

E. R. Odders-White and M. J. Ready. Credit ratings and stock liquidity. *Review of Financial Studies*, 19 (1):119–157, 2006. URL https://doi.org/10.1093/rfs/hhj004. [p31]

R Core Team. *R: A Language and Environment for Statistical Computing*. Vienna, Austria, 2016. [p32]

Y. Yan and S. Zhang. An improved estimation method and empirical properties of the probability of informed trading. *Journal of Banking & Finance*, 36(2):454–467, 2012. URL https://doi.org/10.1016/j.jbankfin.2011.08.003. [p31, 32, 33]

P. Zagaglia. *FinAsym*, 2012. URL https://CRAN.R-project.org/package=FinAsym. R package version 1.0. [p31, 32, 33]

P. Zagaglia. PIN: Measuring Asymmetric Information in Financial Markets with R. *The R Journal*, 5(1):80–86, 2013. URL https://journal.r-project.org/archive/2013/RJ-2013-008/index.html. [p31, 32]

Duygu Çelik
Bilkent University
Bilkent University Department of Management 06800 Bilkent Ankara
Turkey
duygu.celik@bilkent.edu.tr

Murat Tiniç
Bilkent University
Bilkent University Department of Management 06800 Bilkent Ankara
Turkey
tinic@bilkent.edu.tr

RatingScaleReduction package: stepwise rating scale item reduction without predictability loss

by Waldemar W. Koczkodaj, Feng Li, and Alicja Wolny–Dominiak

Abstract This study presents an innovative method for reducing the number of rating scale items without predictability loss. The "area under the receiver operator curve" method (AUC ROC) is used for the stepwise method of reducing items of a rating scale. **RatingScaleReduction** R package contains the presented implementation. Differential evolution (a metaheuristic for optimization) was applied to one of the analyzed datasets to illustrate that the presented stepwise method can be used with other classifiers to reduce the number of rating scale items (variables). The targeted areas of application are decision making, data mining, machine learning, and psychometrics.

Keywords: rating scale, receiver operator characteristic, ROC, AUC, scale reduction.

Introduction

Rating scales are designed to gather data and rate an entity (objects of concepts). Rating scales are also called assessment scales. In our study, we use "the scale" when no ambiguity takes place. Probably, the most significant and frequently used rating scales are exams or tests (e.g., Ontario Driver's test with 40 multiple-choice questions). Some rating scales use values "1 to 10" but five "stars" are gaining popularity for the online reviewing of goods or services. Yes/no answers may also be used for answers in rating scales. Number of questions (called items in the rating scale terminology) may also drastically differ from scale to scale. Numerous rating scales have over 100 items to rate but one item rating scale is also useful for rating the customer's satisfaction with goods or services. An example of a popular rating scale is the intelligence quotient (IQ). It is a total score derived from several standardized tests designed to assess human intelligence.

Sometimes, the scale is called a *survey* or a *questionnaire*. A questionnaire is a tool for data gathering and may not be used for a rating. A survey may not necessarily be conducted by questionnaires and usually does not rate anything. Its goal is to gather data. Some surveys may be conducted by interviews or extracted by Internet agents with or without our consent or knowledge. The important distinction of rating scales from questionnaires and surveys, is that the rating scales are used for assessments. It means that rating scales are expected to have an outcome making them classifiers (in the terminology of statistics and machine learning). The scale term in the rating scale has the meaning as in "the scale of disaster" hence this study assumes that:

$$[ratingscale] = [dataframe] + [assessment] = [classifier]$$

The "assessment" procedure must be in place for a questionnaire or survey to become a rating scale. The assessment procedure may be as complex as the imagination of their authors but most psychiatric rating scales use a simple summation. The simple summation has been used in the first examples. The simple summation can be replaced by, for example, the assignment of weights. In our example 2, we computed weights by the differential evolution (using R packagbe DEval). As expected, it has not influenced the reduction, but it has improved the predictability rate.

"A picture is worth a thousand words" hence Fig. 1 has been used to illustrate ratings scales which are used in many examinations and ratings of various products.

Figure 1: Rating scale example

The recent popularity of rating scales is due to various "Customer Reviews" on the Internet where five stars are often used instead of ordinal numbers. Rating scales are predominantly used to express our subjective assessments such as "on the scale 1 to 5, express your preference" by one selection of: "strongly agree to strongly disagree" with 3 as "neutral" preference. Using a slider implementation (as in Fig. 1) gives us the flexibility. It is known as a graphic or continuous rating scale.

Terminology issue

In our study, we decided to use the data mining (DM) terminology having in mind that regardless of the scientific discipline, the ultimate goal is to predict the outcome of future examples to make a decision. Statistical observations are called examples in DM terminology and items in the rating scale terminology. Statistics uses variables. R project uses attributes and the data frame with all attributes in columns and examples in rows creates an attribute matrix. The additional validation variable is called the decision vector or simply, the decision. In psychiatry, the decision may be "is the patient sick with depression or not." In case of Ontario Driver's test, it is: "do we give a passing grade or not?" The list of application is endless as decision making is of considerable importance for everyone. We also have introduced a new term: "gray examples" for two examples having identical values on all attributes but belonging to two different classes. Evidently, such situation should not occur in the ideal situation but uncertainty in data or imperfect data need to be handled in practice. The gray color is intermediate between black and white and our two examples belong to both classes. One example may be replicated *m* times. Replication of examples would deviate computations and should be detected and removed.

Rating scale reduction

Large rating scales discourage respondents from completing them. It is not unusual to have most answers as random numbers at the end of a long rating scale. It seems that the first successful rating scale reduction took place in Velden M. & Clark (1979) by the psychophysical model by means of signal detection theory (SDT). However, the last paragraph which served as the optional Conclusions at that time:

> It may be concluded that due to the psychophysical difference between the SOT discrimination and the common rating situation, signal detection type reduction of such rating data does not allow interpretation of resulting values as unbiased psychological distances. To avoid misinterpretations, it might be worthwhile not to use the SDT notation for indices derived from psychological rating data.

may be perceived as a risky proposition. For many rating scales (e.g., written academic exams), arbitrary (rarely, computed) weights are applied for some or all items (questions) to improve the overall rating. The proposed method respects it and takes into account only the final "total" (obtained by whatever method or procedure it is established). Formally speaking, the "total" is a metric. The rating scale with metric can be regarded as a classifier for classifying subjects by the rating scale (e.g., sick or not in case of medicine). In Koczkodaj et al. (2017), the stepwise algorithm is proposed for the reduction of the rating scale items by using a metric computed from the confusion matrix by AUC of ROC. The proposed algorithm is restricted to the dichotomous (binary) decision making by the supervised learning approach. Our heuristic algorithm is addressed in Section 10.2. In practice, it is the most needed type of decision (e.g., go/stop, left/right, alive/dead, passed/failed, etc.). There are many ways of transforming general data into dichotomous data to be useful.

A dichotomous (binary) rating scale groups observations (examples) into two categories based on the knowledge about the classified subject. The knowledge is the external assessment since it is the case of supervised learning. If, for example, the division into two groups is "sick/not sick", it is necessary to know if the patient was indeed sick or not by the opinion of an MD to be able to screen (classify) future patients, unnecessarily taking time of usually busy MDs.

Heuristic algorithm and rating scale stepwise reduction procedure

ROC method and corresponding AUC is a well known technique to asses the classifier performance. Both ROC and AUC concepts are well addressed by Fawcett (2004). They are implemented by many **R** packages including: **pROC** (Robin et al., 2011) and **ROCR** (Sing et al., 2005). There is also one interesting web application `easyROC` (Goksuluk et al., 2016) giving possibility to compute the confusion matrix and plot the curve on-line. The **RatingScaleReduction** package expands this analysis to carry out the procedure of rating scale reduction. The main function of the package is based on the procedure described in Koczkodaj et al. (2017).

In computer science and mathematical optimization, a heuristic is a technique (or method) designed for solving a problem by finding an approximate solution when classic methods fail to find the exact solution. Often, finding such a method is achieved by trading completeness, accuracy, or optimality, for the speed. However, most heuristics are designed to find an approximate solution of *NP-complete* problems (NP stands for "nondeterministic polynomial time"). In layman's terms, an

infinite computing time is needed to find an exact solution for an *NP-complete* problem hence a simpler (and usually approximated) solution needs to be accepted.

Evidently, heuristic algorithms produce "good enough"solutions. They are usually not the optimal solution but "good enough" is nearly always better than none. For example, the traveling salesman problem (TSP), often formulated as *find the shortest possible route to visit each city once only and return to the origin city*. It cannot be computed even for 50 cities by verifying all possible combinations since the total number of such permutations is estimated to $O(n^2 2^n)$ although a little bit more optimistic estimations are suspected to exist. By using heuristics, we can solve TSP for millions of cities with the accuracy of a small fraction of 1%.

Heuristic algorithm is a frequently used misnomer. If it is "heuristic", it is not an algorithm and if it is algorithm, it is not a heuristic. However, many heuristics are expressed (written) the same way as algorithms but such "algorithms" do not have well established scientific foundations. Instead, they are based on observations, experience, or even intuition. Some heuristic algorithms may become algorithms. As time passes, we gather more and more evidence and such evidence may lead to finding a theory.

In rating scale reduction problem, the number of possible combinations for a rating scale with 100 items is a "cosmic number" hence the complete search must be ruled out. Certainly, the results need to be verified and used only if the item reduction is substantial. Computing the AUC of ROC for all items is the basis for our heuristic. Common sense dictates that the contribution of the individual items to the overall value of AUC of ROC needs to be somehow utilized. In the **RatingScaleReduction**, the implemented algorithm (when reduced to its minimum) uses a loop for all attributes (with the class excluded) to compute AUC. Subsequently, attributes are sorted in the ascending order by AUC. The attribute with the largest AUC is added to a subset of all attributes (evidently, it cannot be empty since it is supposed to be the minimum subset S of all attributes with the maximum AUC). We continue adding the next in line (according to AUC) attribute to the subset S checking AUC. If it decreases, we stop the procedure. There is a lot of checking (e.g., if the dataset is not empty or full of replications) involved. In a more formal way, the `RSR` procedure implemented in **RatingScaleReduction** has the general steps:

1. input: $attM[i, j]$ - attribute matrix, $i = 1, ..., n, j = 1, ..., m$,
   where n - the number of examples, m - the number of columns,
   $D[i]$ - decision vector $i = 1, ..., n$
2. iterate in the loop:

```
for (j in 1:m){
    calculate AUC[j]
    }
```

3. sort the vector $AUC[1], ..., AUC[m]$ in the descending order receiving a new vector

$$AUCs[1], ..., AUCs[m]$$

4. create a new attribute matrix $attMs[i, j]$ with columns sorted in the descending order according to the vector:

$$AUCs[1], ..., AUCs[m]$$

5. iterate in the loop:

```
attMs[ ,1] = attM[ ,1]
for (j in 2:m){
    D.predict[j] = sum(attMs[ ,1],...,attMs[ ,j])
    calculate AUCtotal[j] = auc(D[j], D_predict[j])
    }
```

6. iterate in the loop:

```
k = 1
while (AUCtotal[k+1] > AUCtotal[k]){
    attMreduced = cbind(attMs[ ,1],...,attMs[ ,k])
    k = k+1
    }
```

7. output: the reduced attribute matrix `attMreduced` generating the reduced rating scale.

First part of `RSR` procedure is implemented by R functions **startAuc**, **totalAuc** while the second part is the main function of the package called **rsr**.

RatingScaleReduction: overview of the package functions

and shown by Fig.

The **RatingScaleReduction** package implements the above-stated stepwise procedure using two functions of the **pROC** package: `roc` and `roc.test`. The data can be the `matrix` or `data.frame`. Columns represent attributes and one column is the class with two categories: 0 or 1 (or any other two different integer or real values). The rows in `data.frame` represents examples. For our package, all attributes and the class must be numeric (preferably 0 or 1) hence some preprocessing may be needed.

There are two groups of functions available in the package. The first group is for implementing the core of the `RSR` algorithm:

1. `startAuc(attribute,D)` – compute the AUC values of every single attribute in the rating scale.
2. `totalAuc(attribute,D,plotT=FALSE)` – sort AUC values in the ascending order and compute AUCs of running total of first k attributes, $k = 1, ..., n$, where n is the number of attributes. Setting the argument `plotT` as `TRUE` the plot of new AUC values is created. The horizontal line marks the max new AUC.
3. `rsr(attribute,D,plotRSR=FALSE)` – the main function of the package reducing the rating scale. Setting the argument `plotRSR` as `TRUE` the plot of ROC curve of the sum of attributes in reduced rating scale is created.
4. `CheckAttr4Inclusion(attribute,D)` – subsequently, we check the next attribute for the possible inclusion in the reduced set of attribute. It is done by maximizing AUC of all already included attributes and the attribute we have just checked. In some cases, all attributes will be included in the new set of attributes. The reduced set of one attribute may be created if there is an identifying attribute. The function `CheckAttr4Inclusion` tests the inclusion. It carried out a statistical test for a difference in AUC of two correlated ROC curves: ROC1 of the sum of attributes from reduced rating scale and ROC2 of this sum plus the next ordered attribute. The function `roc.test` from the **pROC** is used and all implemented tests are available, in particular `delong` and `bootstrap`.

The package **RatingScaleReduction** also contains the second group of functions to support the reduction procedure. Before running the `PROC1`, the dataset should be analyzed to detect replicated examples (gray examples). This analysis of a dataset can be done by using functions: `diffExamples` and `grayExamples`.

1. `diffExamples(attribute)` – search replicated examples in the data and return the number of different examples and the number of duplicates.
2. `grayExamples(attribute,D)` – produce the list of pairs of examples having identical values on all attributes. The decision value and attributes are produced for every pair in the dataset, so the list clearly shows all gray examples.

In the examples presenting the capabilities of the **RatingScaleReduction** package, we have used the following two datasets:

1. Wine quality demonstrates how most of our package functions are used,
2. Somerville Happiness Survey for the use of differential evolution (DE) classifier.

Subsequently, we utilized our examples to demonstrate the capabilities of the **RatingScaleReduction** package. The full `R` code is available for download from https://github.com/woali/RatingScaleReduction/blob/master/example_Rj.r.

The first demonstration example: wine quality

Wine quality dataset taken from http://archive.ics.uci.edu/ml/index.php and available in the object `wineData` from the **RatingScaleReduction** package is used in this example. It has 6497 examples and 11 attributes. The reduction is achieved by three core functions of the package. The `data.frame` we work on contains 11 columns with attributes and one additional column as a decision (reality).

We begin the analysis by computing AUC for all 11 individual attributes by the use of function `totalAuc`. Setting the argument `plotT` as `TRUE` produced a plot.

```
> tauc.wine <-totalAuc(attribute, D, plotT=TRUE)

> tauc.wine$summary
                     AUC one variable AUC running total
```

```
alcohol                0.6098691        0.6098691
volatile.acidity       0.6047200        0.6397657
fixed.acidity          0.5656387        0.6415913
citric.acid            0.5550811        0.6294707
total.sulfur.dioxide   0.5536676        0.5971653
sulphates              0.5394024        0.5824898
density                0.5218842        0.5109257
chlorides              0.5215313        0.5227466
pH                     0.5134569        0.5229943
free.sulfur.dioxide    0.5052413        0.5237751
residual.sugar         0.4957781        0.5286474
```

The **R** output shows the `tauc.wine$summary` AUC of every single attribute in the second column, sorted in the ascending order. The running total of AUCs is in the third column. The initially selected variable (alcohol) for the first row is the attribute with the largest AUC. Subsequently, we add to it the variable with the largest AUC of the remaining attributes. The process continues while the last attribute of the scale is added.

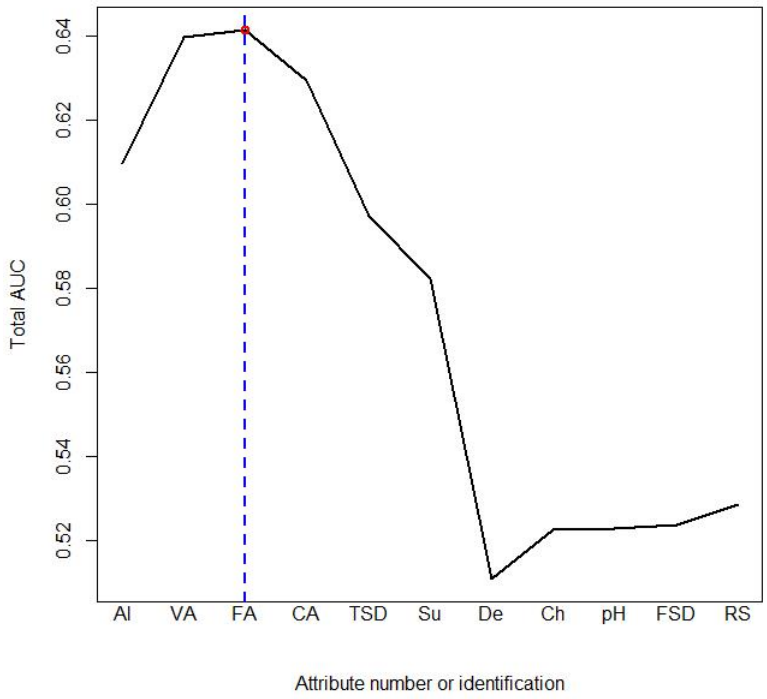


Figure 2: A stepwise AUC reduction method (example)

Printing the value `tauc.wine$item` we receive the attribute labels in an ascending order.

```
> tauc.wine$item

 [1] "alcohol"              "volatile.acidity"     "fixed.acidity"
[4] "citric.acid"          "total.sulfur.dioxide" "sulphates"
[7] "density"              "chlorides"            "pH"
[10] "free.sulfur.dioxide"  "residual.sugar"
```

As illustrated by Fig. 2, the value of AUC of the selected subset of attributes is increasing by adding the first three attributes labeled alcohol, volatile.acidity and fixed.acidity. For this reason, the reduction procedure is terminated after the first six attributes are added. The function `rsr` reduces the scale automatically assuming the truncation point as the attribute that first reaches the maximum AUC. AUC is a real value between 0 and 1. It is 0.5 for random data but hardly ever reaches 1 since , in reality, there are always "gray examples" in sizable data.

```
> rsr.wine <-rsr(attribute, D, plotRSR=TRUE)
The criteria: Stop first MAX AUC
> rsr.wine$rsr.auc

[1] 0.6098691 0.6397657 0.6415913

> rsr.wine$rsr.label
[1] "alcohol"          "volatile.acidity" "fixed.acidity"

> rsr.wine$summary
AUC one variable AUC running total
alcohol                0.6098691         0.6098691
volatile.acidity       0.6047200         0.6397657
fixed.acidity          0.5656387         0.6415913
```

Setting the `rsr` parameter `plotRSR` as `TRUE` the function generates the plot illustrated by Fig. 3.

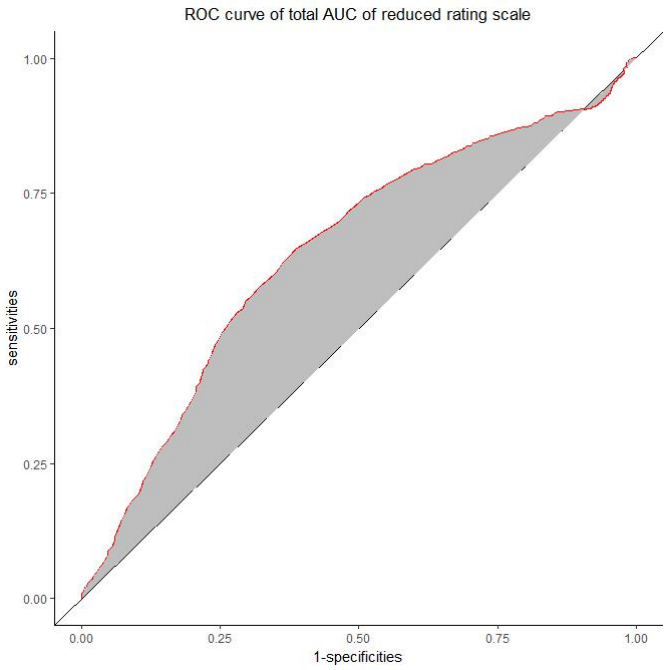


Figure 3: AUC of $alcohol + volatile.acidity + fixed.acidity$ for Wine dataset

We assume that by selecting the "best" attribute in a loop, we are able to reduce the number of attributes for the best preventiveness. In our case, having the largest AUC is the "best" criterion. Adding the next "best" attribute to the selected attribute from the subset of the remaining attributes

until AUC of all selected attributes decreases is the main idea of our heuristic. So far, each and every rating scale has been reduced.

The second demonstration example: DEvol classifiers

Somerville Happiness Survey (SHS) dataset has been used as the second example to demonstrate the use of Differential Evolution (DE) as a classifier to enhance the data be preprocessing. The use "survey" is warranted since it gathers "other data" but it has subscale (a part of it) for the happiness rating. This survey has been given sent out to a random sample of Somerville residents asking them to rate their personal happiness and their satisfaction with city services every second year since 2011. Every year, the survey is refined. We used data of year 2015 since this survey is the most mature. SHS dataset uses what is called a subscale. The processed survey data are available in the object `SHSData` in the **RatingScaleReduction**. It is a subscale marked s block "6" and shown by Fig. 4. The decision (class) attribute is the SHS survey question "3":
How satisfied are you with Somerville as a place to live?
mapped into 0-1 (0 for values less than 8 otherwise 1).

Figure 4: Survey for collecting SHS scale

For this subscale, we used Differential Evolution (DE) classifier by using `DEoptim` from **DEoptim** R package. `DEoptim` computes optimal weights (as a vector w) for a given data. Given data are modified by the scalar vector multiplication.

It is worth noticing that all rating scales can be improved by `DEoptim`. In the worse case scenario, the initial vector $[1, 1, \ldots, 1]$ will remain unchanged. In our case, DEval improved the predictability measured by AUC from 0.678 to 0.789.

```
> D.predict <- rowSums(attribute)
> roc(D, D.predict, plotROC = FALSE)$auc
Area under the curve: 0.6794
```

The RSR procedure gives the following reduction:

```
> rsrSum <- rsr(attribute, D, plotRSR = TRUE)
> rsrSum
$rsr.auc
[1] 0.6664868 0.6934700 0.7330005

$rsr.label
[1] "X6" "X1" "X4"

$summary
   AUC one variable AUC running total
X6        0.6664868         0.6664868
X1        0.6542094         0.6934700
X4        0.6083378         0.7330005
```

The results are illustrated by plots (see Fig. 5 and 6).

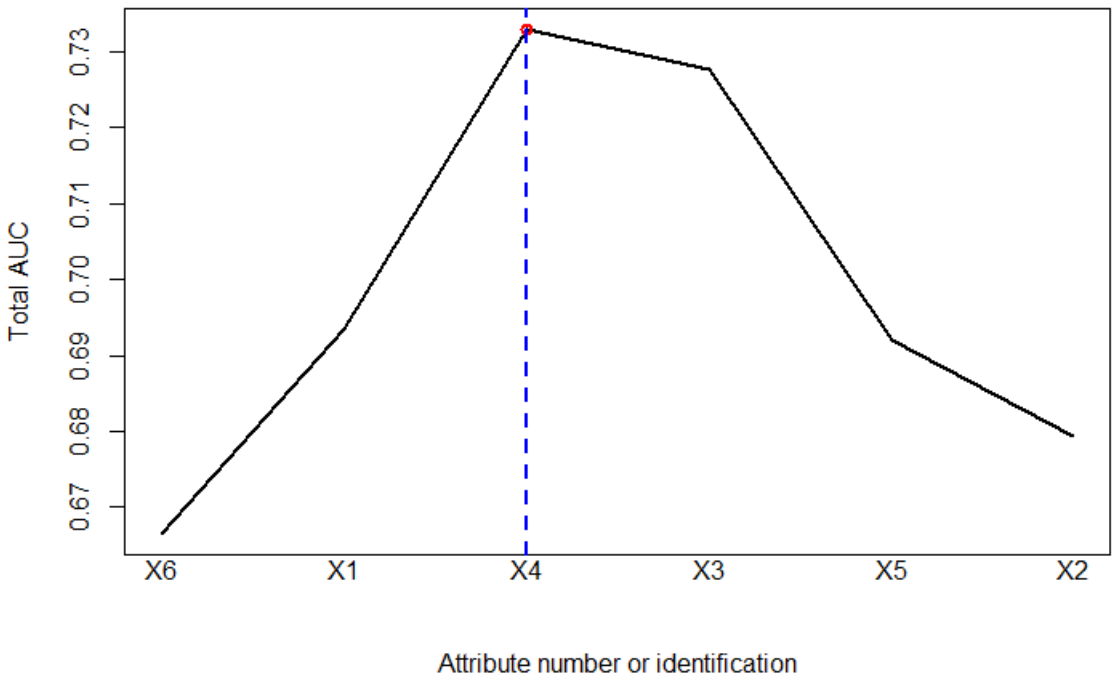


Figure 5: Somerville happiness survey dataset

Differential evolution (DE) as a classifier In RSR procedure, we change the classifier from the simple to the weighted total. The optimal weights are received using DE.

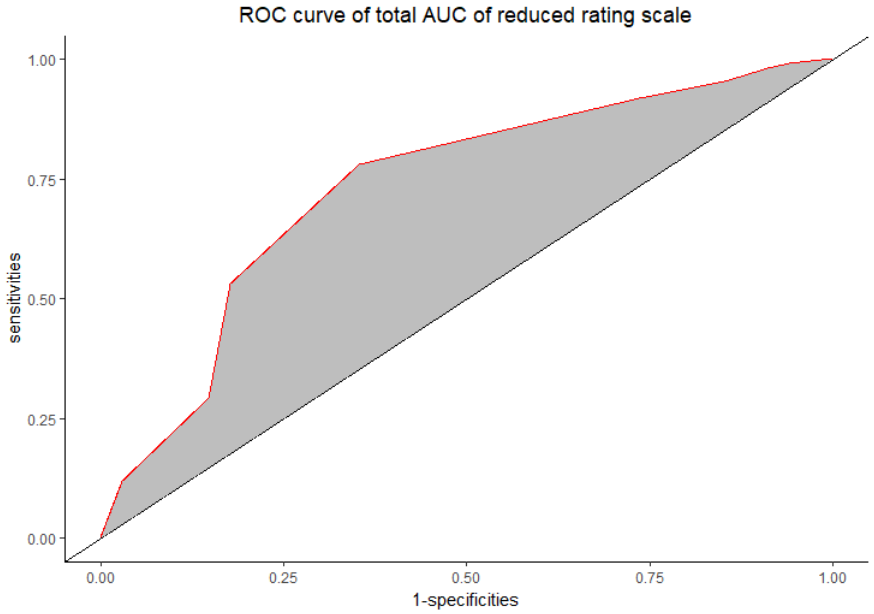


Figure 6: Somerville happiness survey dataset

The goal function is:

```
nsi<-function(x){
  D.predict <- rowSums(x*attribute)
  -1*roc(D, D.predict, plotROC = FALSE)$auc
}
```

where $x = x[j], j = 1, ..., 6$ are weights of items.

The initial lower and upper bounds, required by DE, are set to:

```
lower_nsi <- c(0.1,0.1, 0.1, 0.1, 0.1,0.1)
upper_nsi <- c(3,3,3,3,3,3)
```

DE optimization:

```
output <- DEoptim(nsi, lower_nsi, upper_nsi,
    DEoptim.control(itermax=10))

#ouput the optimize result of weights
> (weight.item.all <- output.all$optim$bestmem)
    par1     par2     par3     par4     par5     par6
1.405835 1.882602 1.758763 2.356450 1.123748 2.085982

#all items
> (aucResult <- -1*output$optim$bestval)
[1] 0.766055
```

In order to reduce the scale using DE, we have modified `RSR` procedure by changing D.predict formula to:

```
D.predict[j] = sum(x[1]*attMs[ ,1],...,x[1]*attMs[ ,j])
```

The sorted attribute matrix $attMs[i, j]$ we use in the loop:

```
results <- matrix(nrow = ncol(attMs), ncol = 2)
for (i in 2:ncol(attMs)) {
  mydata_new <- attMs[ ,1:i]

  lower_nsi <- rep(0.1,i)
  upper_nsi <- rep(3,i)

  output <- DEoptim(nsi, lower_nsi, upper_nsi, DEoptim.control(itermax=10))
  results[i, ] <- cbind(i, -1*output$optim$bestval)

}
```

We have obtained the reduction:

```
> results
     number of items AUCtotalDE
[1,]               1  0.5226659
[2,]               2  0.6991365
[3,]               3  0.7621425
[4,]               4  0.7664598
[5,]               5  0.7251754
[6,]               6  0.7232866

> names(attMs)
[1] "X6" "X1" "X4" "X3" "X5" "X2"
```

Plots are illustrated by Fig. 7 and 8

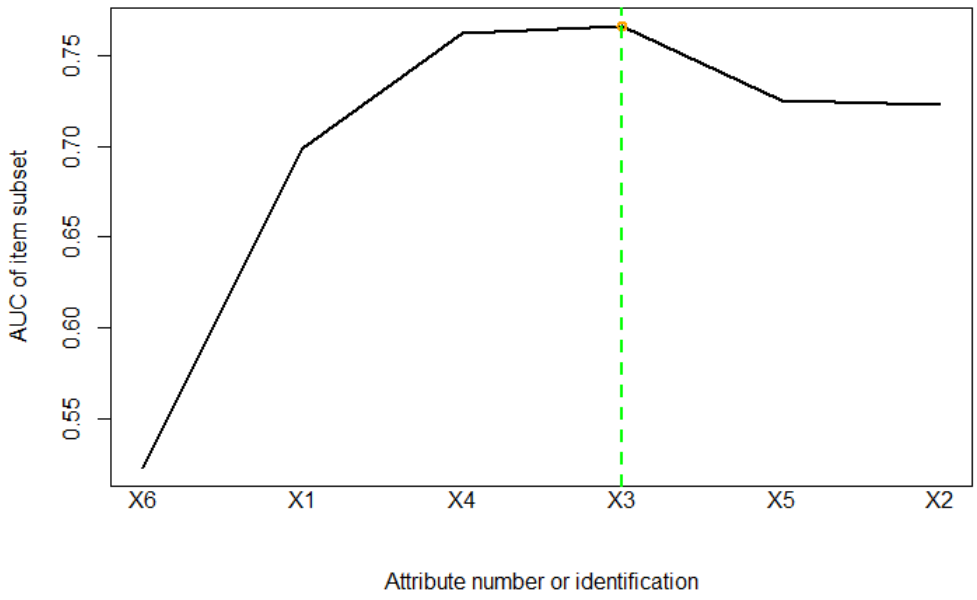


Figure 7: Enhanced (by DE) SHS dataset

The potential application targets

Rating scales are by far more important contributors to practically all branches of applied science and engineering than we can address by this study. Most examinations for granting scientific degrees are rating scales in various shapes and forms. Simplifying them (or reducing in size) is needed if the predictability is preserved or increased.

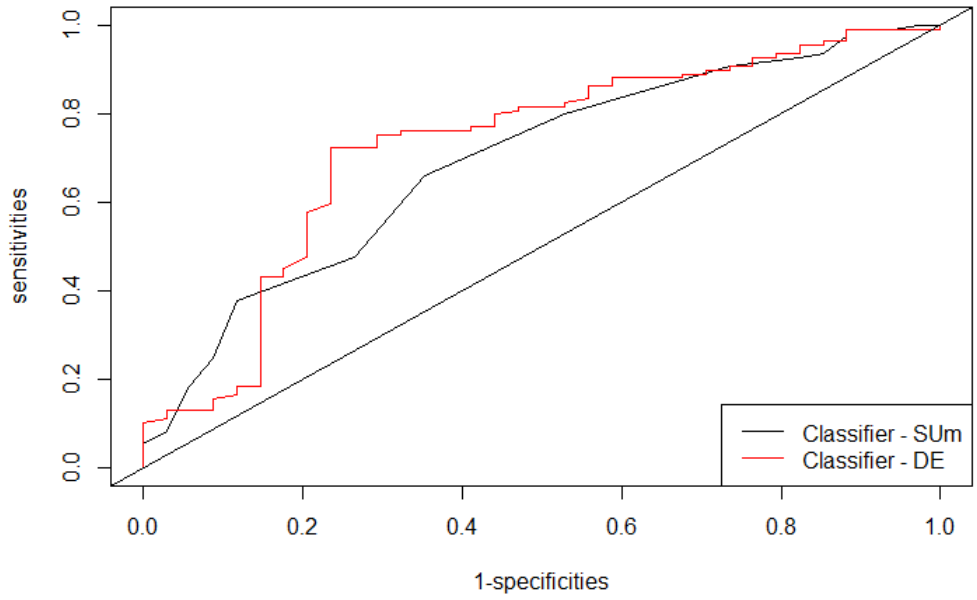


Figure 8: Enhanced (by DE) SHS dataset dataset

In bioinformatics, reporting trade-off in sensitivity and specificity, by using a Receiver Operating Characteristic (ROC) curve, is becoming a common practice. ROC plot has the sensitivity on the y axis, against the false discovery rate (1- specificity) on the x axis. ROC curve plot provides a visual tool to determine the boundary limit (or the separation threshold) of a subset (or a combination) of scale items for the potentially optimal combination of sensitivity and specificity. The area under the curve (AUC) of the ROC curve indicates the overall accuracy and the separation performance of the rating scale. It can be readily used to compare different item subsets. As a rule of thumb, the fewer scale items used to maximize the AUC of the ROC curve, the better.

World Health Organization estimates are included behind selected rating scales for mental disorder. Rating scales are of considerable importance for psychiatry where they are predominately used for screening patients for mental disorders such as:

- depression (see Koczkodaj et al. (2017)) which affects 60 million people worldwide according to WHO (2016),
- bipolar affective disorder (60 million people),
- dementia and cognitive impairment (47.5 million people)
- schizophrenia (21 million people),
- autism and autism spectrum disorders (e.g., Kakiashvili et al. (2012))
- addiction,
- personality and personality disorders,
- anxiety,
- ADHD;

and many other disorders.

Usually, there are many scales for each mental disorder. The most important for screening are global scales. Reducing these global rating scales makes them more usable as indicated in Koczkodaj et al. (2017). World Health Organization Media Centre reports that depression and anxiety disorders cost the global economy US $1 trillion each year" and it is no longer a local problem.

Conclusions

The presented method has reduced the number of the rating scale items (variables) to 28.57% from the original number of items (from 21 to 6). It means that over 70% of collected data was unnecessary. It is

not only an essential budgetary saving, as the data collection is usually expensive and may easily go into hundreds of thousands of dollars, but excessive data collection may contribute to data collection error increase. The more data are collected, the more errors may occur since a lack of concentration and boredom are realistic factors.

By using the proposed AUC ROC reduction method, the predictability has increased by approximately 0.5%. It may seem insignificant. However, for a large population, it is of considerable importance. In fact, WHO (2016) states that: "Taken together, mental, neurological and substance use disorders exact a high toll, accounting for 13% of the total global burden."

As pointed out, rating scales are used for mental disorders. According to WHO (2016), mental disorders are becoming a global problem.

The proposed use of AUC for reducing the number of rating scale items, as a criterion, is innovative and applicable to practically all rating scales. In the worse case scenario, no reduction takes place (the number of reduced attributes is 0). System **R** code is posted on the Internet (RatingScaleReduction) for the general use as a **R** package. Certainly, more validation cases would be helpful and the assistance will be provided to anyone who wishes to try this method using his/her data.

Future plans include using the presented method for measuring the harmful use of the Internet and for the improvement of numerous psychiatric scales. The reduced scales can be further enhanced by the method described in Kakiashvili et al. (2012) and Koczkodaj (1996).

Acknowledgments

The first author has been supported in part by the Euro Research grant "Human Capital".

The authors would also like to express appreciation to Tiffany Armstrong (Laurentian University, Computer Science), and Grant O. Duncan, Team Lead, Business Intelligence, Integration and Development, Health Sciences North, Sudbury, Ontario, Canada) for the editorial improvements of our text and their creative comments.

Bibliography

T. Fawcett. ROC graphs: Notes and practical considerations for researchers. *Machine learning*, 31(1): 1–38, 2004. [p44]

D. Goksuluk, S. Korkmaz, G. Zararsiz, and A. E. Karaagaoglu. easyROC: An interactive web-tool for ROC curve analysis using R language environment. *The R Journal*, 2016. [p44]

T. Kakiashvili, W. W. Koczkodaj, and M. Woodbury-Smith. Improving the medical scale predictability by the pairwise comparisons method: Evidence from a clinical data study. *Computer Methods and Programs in Biomedicine*, 105(3):210–216, 2012. URL https://doi.org/10.1016/j.cmpb.2011.09.011. [p53, 54]

W. W. Koczkodaj. Statistically accurate evidence of improved error rate by pairwise comparisons. *Perceptual and Motor Skills*, 82(1):43–48, 1996. [p54]

W. W. Koczkodaj, T. Kakiashvili, A. Szymańska, J. Montero-Marin, R. Araya, and J. Garcia-Campayo, K. Rutkowski, and D. Strzałka. How to reduce the number of rating scale items without predictability loss? *Scientometrics*, 111:581–593, 2017. [p44, 53]

X. Robin, N. Turck, A. Hainard, N. Tiberti, F. Lisacek, J.-C. Sanchez, and M. Müller. pROC: An open-source package for R and S+ to analyze and compare ROC curves. *BMC Bioinformatics*, 12:77, 2011. [p44]

T. Sing, O. Sander, N. Beerenwinkel, and T. Lengauer. ROCR: Visualizing classifier performance in R. *Bioinformatics*, 21(20):3940–3941, 2005. [p44]

W. Velden M. & Clark. Reduction of rating scale data by means of signal detection theory. *Psychophysics*, 25(6):517–518, 1979. doi:10.3758/BF03213831. [p44]

WHO. Mental disorders fact sheet. *"www.who.int/mediacentre/factsheets/fs396/en/"*, April, 2016. [p53, 54]

Waldemar W. Koczkodaj
Computer Science
Ramsey Lk Rd.

Laurentian University
Sudbury, Ontario P3E 2C6
Canada
wkoczkodaj@cs.laurentian.ca
http://cs.laurentian.ca/wkoczkodaj/info

Feng Li
Computer Science
Ramsey Lk Rd.
Laurentian University
Sudbury, Ontario P3E 2C6
Canada
spacejohn@live.cn

Alicja Wolny–Dominiak
Department of Statistical and Mathematical Methods in Economics
University of Economics in Katowice
ul. 1 Maja 50
40–287 Katowice
Poland
alicja.wolny-dominiak@uekat.pl
https://woali.github.io/rphdstatistics

mmpf: Monte-Carlo Methods for Prediction Functions

by Zachary M. Jones

Abstract Machine learning methods can often learn high-dimensional functions which generalize well but are not human interpretable. The **mmpf** package marginalizes prediction functions using Monte-Carlo methods, allowing users to investigate the behavior of these learned functions, as on a lower dimensional subset of input features: partial dependence and variations thereof. This makes machine learning methods more useful in situations where accurate prediction is not the only goal, such as in the social sciences where linear models are commonly used because of their interpretability.

Many methods for estimating prediction functions produce estimated functions which are not directly human-interpretable because of their complexity: for example, they may include high-dimensional interactions and/or complex nonlinearities. While a learning method's capacity to automatically learn interactions and nonlinearities is attractive when the goal is prediction, there are many cases where users want good predictions *and* the ability to understand how predictions depend on the features. **mmpf** implements general methods for interpreting prediction functions using Monte-Carlo methods. These methods allow any function which generates predictions to be be interpreted. **mmpf** is currently used in other packages for machine learning like **edarf** and **mlr** (Jones and Linder, 2016; Bischl et al., 2016).

Marginalizing Prediction Functions

The core function of **mmpf**, `marginalPrediction`, allows marginalization of a prediction function so that it depends on a subset of the features. Say the matrix of features $\mathbf{X}$ is partitioned into two subsets, $\mathbf{X}_u$ and $\mathbf{X}_{-u}$, where the former is of primary interest. A prediction function f (which in the regression case maps $\mathbf{X} \rightarrow \mathbf{y}$) where $\mathbf{y}$ is a real-valued vector might not be additive or linear in the columns of $\mathbf{X}_u$, making f difficult to interpret directly. To obtain the marginal relationship between $\mathbf{X}_u$ and f we could marginalize the joint distribution so that we obtain a function f_u which only depends on the relevant subset of the features:

$$f_u(\mathbf{X}_u) = \int f(\mathbf{X}_u, \mathbf{X}_{-u}) \mathbb{P}(\mathbf{X}_u | \mathbf{X}_{-u}) \mathbb{P}(\mathbf{X}_{-u}) d\mathbf{X}_{-\mathbf{u}}.$$

This however, can distort the relationship between $\mathbf{X}_u$ and f because of the inclusion of dependence between $\mathbf{X}_u$ and $\mathbf{X}_{-u}$ (specifically $\mathbf{X}_u | \mathbf{X}_{-u}$), which is unrelated to f. An alternative is to instead integrate against the marginal distribution of $\mathbf{X}_{-u}$ as in 12.1, as suggested by (Friedman, 2001):

$$\tilde{f}_u(\mathbf{X}_u) = \int f(\mathbf{X}_u, \mathbf{X}_{-u}) \mathbb{P}(\mathbf{X}_{-u}) d\mathbf{X}_{-\mathbf{u}}.$$

To illustrate this point, suppose data are generated from an additive model, $f(\cdot) = \mathbf{x_1} + \mathbf{x_2}$ and $(\mathbf{x}_1, \mathbf{x}_2) \sim \text{MVN}(\mathbf{0}, \Sigma)$ where the diagonal entries of Σ are 1 and the off-diagonals are 0.5. That is, $(\mathbf{x}_1, \mathbf{x}_2)$ are correlated by construction. Now if we want to understand how f depends on $\mathbf{x}_1$ we could integrate against the true joint distribution, as in 12.1. However, this distorts the relationship between $\mathbf{x}_1$ and f, because the conditional probability of $\mathbf{x}_1$ given $\mathbf{x}_2$ is higher at values of $(\mathbf{x}_1, \mathbf{x}_2)$ which are more extreme (due to their correlation). Since $\mathbf{x}_2$ is related to f, this has the effect of distorting the relationship between $\mathbf{x}_1$ and f, and, in this case, makes the relationship appear more extreme than it is, as can be seen in the left panel of Figure 1. This distortion of the relationship between $\mathbf{x}_1$ and f can be made more misleading if $\mathbf{x}_2$ interacts with $\mathbf{x}_1$ to produce f, or if $\mathbf{x}_2$ has a nonlinear relationship with f, as can be seen in the right panel of Figure 1.

Integrating against the marginal distribution of $\mathbf{x}_1$ recovers the true additive effect (left panel) and the average marginal effect ($\mathbf{x}_1 + 0.5\mathbf{x}_1\bar{\mathbf{x}}_2$, in the right panel) respectively.

Using mmpf

In practical settings we do not know $\mathbb{P}(\mathbf{X})$ or f. We can use $\hat{f}$, estimated from $(\mathbf{X}, \mathbf{y})$ as a plug-in estimator for f and can estimate $\mathbb{P}(\mathbf{X}_{-u})$ from the training data, allowing us to estimate the *partial dependence* of $\mathbf{X}_u$ on $\hat{f}$ Friedman (2001).

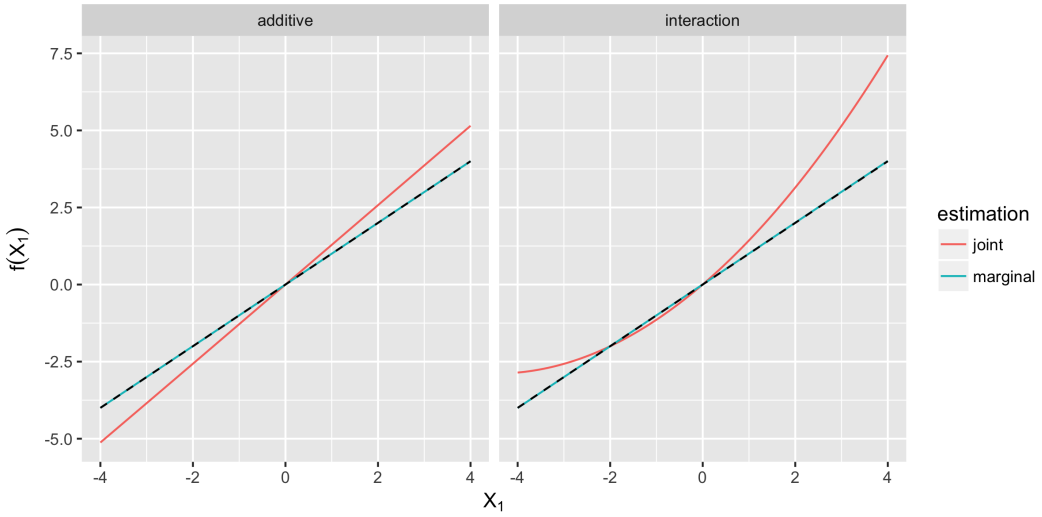


Figure 1: The marginal relationship between $\mathbf{x}_1$ and f as estimated by integrating against the marginal distribution of $\mathbf{x}_2$ (the blue line) or the joint distribution of $(\mathbf{x}_1, \mathbf{x}_2)$ (the red line). The true relationship is shown by the dashed line. In the left panel f is an additive function of $\mathbf{x}_1$ and $\mathbf{x}_2$ and in the right panel $\mathbf{x}_1$ and $\mathbf{x}_2$ interact via multiplication to produce f.

$$\hat{f}_u(\mathbf{X}_u) = \sum_{i=1}^{N} \hat{f}(\mathbf{X}_u, \mathbf{X}_{-u}^{(i)})$$

This the behavior of the prediction function at a vector or matrix of values for $\mathbf{X}_u$, averaged over the empirical marginal distribution of $\mathbf{X}_{-u}$.

The function `marginalPrediction` allows users to compute easily partial dependence and many variations thereof. The key arguments of `marginalPrediction` are the prediction function (`predict.fun`), the training data (`data`), the names of the columns of the training data which are of interest (`vars`), the number of points to use in the grid for $\mathbf{X}_u$, and the number of points to sample from $\mathbf{X}_{-u}$ (`n`, an integer vector of length 2). Additional arguments control how the grid is constructed (e.g., uniform sampling, user chosen values, non-uniform sampling), indicate the use of weights, and instruct how aggregation is done (e.g., deviations from partial dependence). Below is an example using the Iris data (Anderson, 1936):

```
library(mmpf)
library(randomForest)

data(iris)
iris.features = iris[, -ncol(iris)] # exclude the species column
fit = randomForest(iris.features, iris$Species)

mp = marginalPrediction(data = iris.features,
  vars = "Petal.Width",
  n = c(10, nrow(iris)), model = fit, uniform = TRUE,
  predict.fun = function(object, newdata) predict(object, newdata, type = "prob"))
print(mp)
##     Petal.Width    setosa versicolor virginica
##  1:   0.1000000 0.6374133  0.2337733 0.1288133
##  2:   0.3666667 0.6374133  0.2337733 0.1288133
##  3:   0.6333333 0.6356267  0.2350533 0.1293200
##  4:   0.9000000 0.1707200  0.5997333 0.2295467
##  5:   1.1666667 0.1688267  0.6016267 0.2295467
##  6:   1.4333333 0.1688133  0.5880800 0.2431067
##  7:   1.7000000 0.1640400  0.4242800 0.4116800
##  8:   1.9666667 0.1619867  0.2066667 0.6313467
##  9:   2.2333333 0.1619867  0.2047867 0.6332267
## 10:   2.5000000 0.1619867  0.2047867 0.6332267
```

In this case, $\hat{f}$ returns a probability of membership in each class for each value of the variable

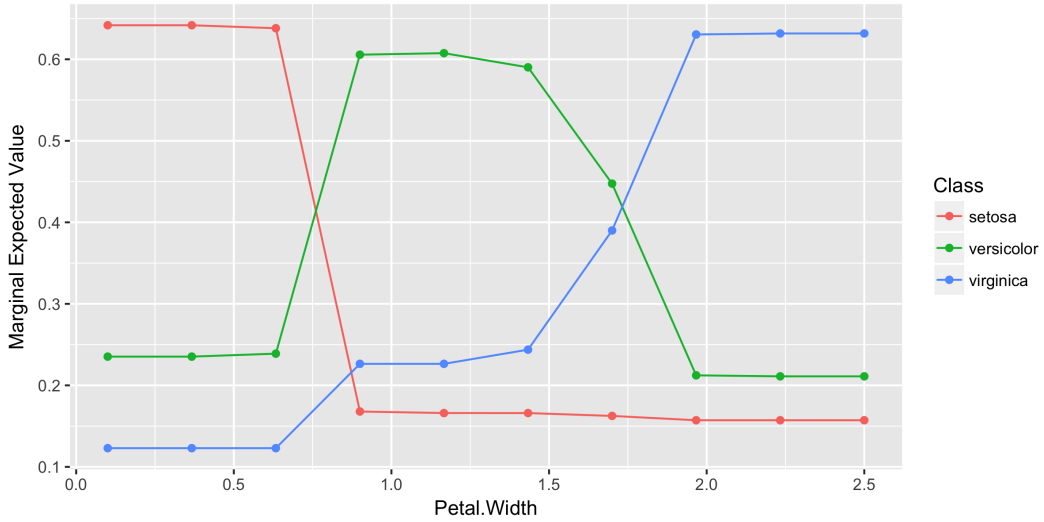


Figure 2: The expected value of $\hat{f}$ estimated by a random forest and then marginalized by Monte-Carlo integration to depend only on "`Petal.Width`."

"`Petal.Width`." This is computed based on the average prediction for each value of "`Petal.Width`" shown and all the observed values of the other variables in the training data. As can be readily observed, partial dependence can be easily visualized, as in Figure 2.

In fact, *any* function of the marginalized function $\hat{f}_u$ can be computed, including vector-valued functions. For example the expectation and variance of $\hat{f}_u$ can be simultaneously computed, the results of which are shown in Figures 3 and 4. Computing the variance of $\hat{f}_u$ can be used for detecting interactions between $\mathbf{X}_u$ and $\mathbf{X}_{-u}$ (Goldstein et al., 2015). If the variance of $\hat{f}_u(\mathbf{X}_u)$ is constant then this indicates that $\mathbf{X}_{-u}$ does not interact with $\mathbf{X}_u$, since, if it did, this would make $\hat{f}$ more variable in regions of the joint distribution wherein there is interaction between $\mathbf{X}_u$ and $\mathbf{X}_{-u}$.

```
mp.int = marginalPrediction(data = iris.features,
  vars = c("Petal.Width", "Petal.Length"),
  n = c(10, nrow(iris)), model = fit, uniform = TRUE,
  predict.fun = function(object, newdata) predict(object, newdata, type = "prob"),
  aggregate.fun = function(x) list("mean" = mean(x), "variance" = var(x)))

head(mp.int)
##    Petal.Width Petal.Length setosa.mean setosa.variance versicolor.mean
## 1:         0.1     1.000000   0.9549867    0.0011619193      0.04448000
## 2:         0.1     1.655556   0.9549867    0.0011619193      0.04448000
## 3:         0.1     2.311111   0.9530933    0.0011317899      0.04637333
## 4:         0.1     2.966667   0.4574667    0.0003524653      0.52818667
## 5:         0.1     3.622222   0.4550400    0.0002619447      0.53061333
## 6:         0.1     4.277778   0.4550400    0.0002619447      0.52472000
##    versicolor.variance virginica.mean virginica.variance
## 1:         0.001141889   0.0005333333      0.00000239821
## 2:         0.001141889   0.0005333333      0.00000239821
## 3:         0.001112236   0.0005333333      0.00000239821
## 4:         0.001154918   0.0143466667      0.00054076492
## 5:         0.001016158   0.0143466667      0.00054076492
## 6:         0.001556364   0.0202400000      0.00093196886
```

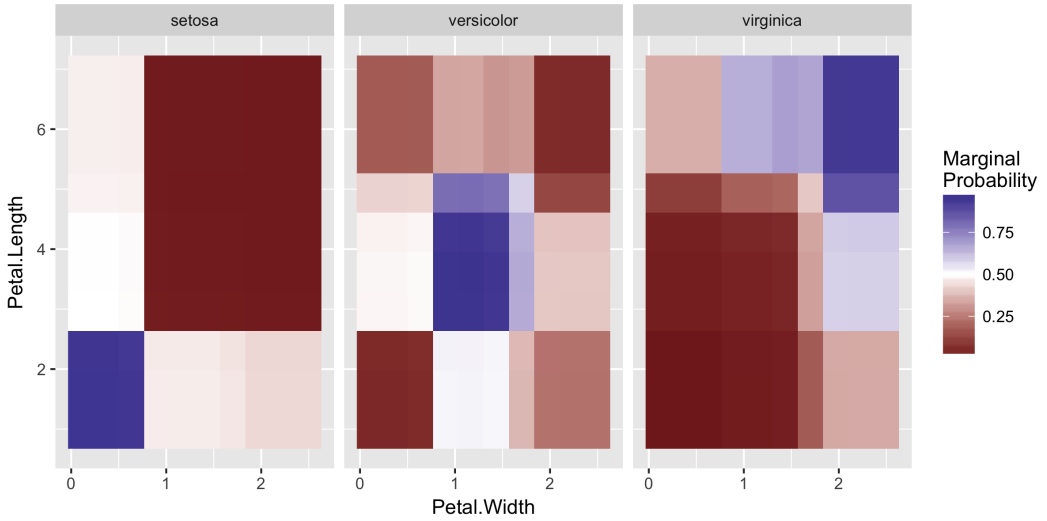


Figure 3: The expected value of $\hat{f}$ estimated by a random forest and then marginalized by Monte-Carlo integration to depend only on "`Petal.Width`" and "`Petal.Length`."

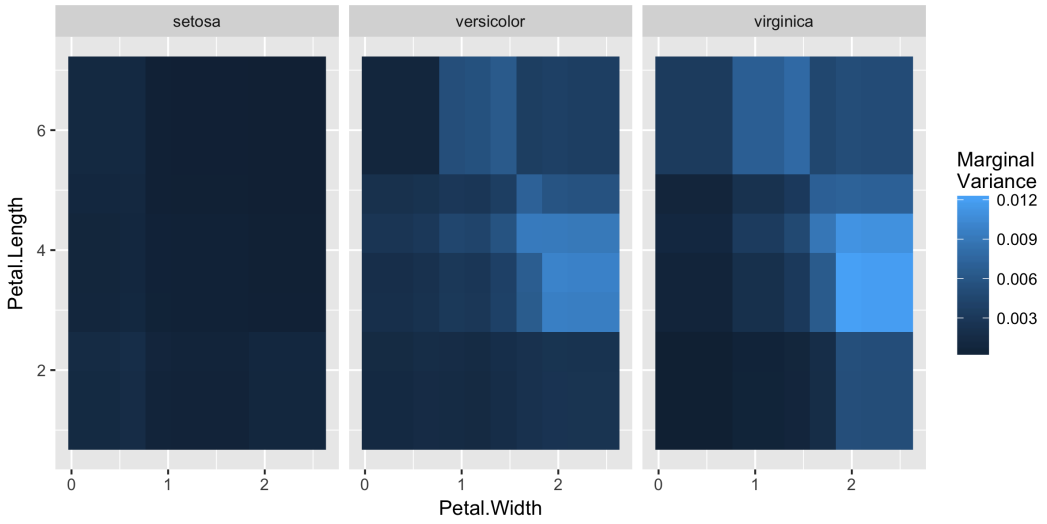


Figure 4: The variance of $\hat{f}$ estimated by a random forest and marginalized by Monte-Carlo integration to depend only on "`Petal.Width`" and "`Petal.Length`." Non-constant variance indicates interaction between these variables and those marginalized out of $\hat{f}$.

Bibliography

E. Anderson. The species problem in iris. *Annals of the Missouri Botanical Garden*, 23(3):457–509, 1936. [p57]

B. Bischl, M. Lang, L. Kotthoff, J. Schiffner, J. Richter, E. Studerus, G. Casalicchio, and Z. M. Jones. mlr: Machine learning in R. *Journal of Machine Learning Research*, 17(170):1–5, 2016. URL http://jmlr.org/papers/v17/15-066.html. [p56]

J. H. Friedman. Greedy function approximation: a gradient boosting machine. *Annals of Statistics*, pages 1189–1232, 2001. [p56]

A. Goldstein, A. Kapelner, J. Bleich, and E. Pitkin. Peeking inside the black box: Visualizing statistical learning with plots of individual conditional expectation. *Journal of Computational and Graphical Statistics*, 24(1):44–65, 2015. [p58]

Z. M. Jones and F. J. Linder. edarf: Exploratory data analysis using random forests. *The Journal of Open Source Software*, 1(6), 2016. URL https://doi.org/10.21105/joss.00092. [p56]

Zachary M. Jones
Pennsylvania State University
University Park, Pennsylvania
United States
zmj@zmjones.com

Generalized Additive Model Multiple Imputation by Chained Equations With Package ImputeRobust

by Daniel Salfran, Martin Spiess

Abstract Data analysis, common to all empirical sciences, often requires complete data sets. Unfortunately, real world data collection will usually result in data values not being observed. We present a package for robust multiple imputation (the **ImputeRobust** package) that allows the use of generalized additive models for location, scale, and shape in the context of chained equations. The paper describes the basics of the imputation technique which builds on a semi-parametric regression model (GAMLSS) and the algorithms and functions provided with the corresponding package. Furthermore, some illustrative examples are provided.

Introduction

A common approach to allow valid inferences in the presence of missing data is "Multiple Imputation" (MI) introduced by Rubin (1987). Application of MI can be summarized in three steps. The first step is to create $m > 1$ sets of completed data sets by replacing each missing value with m values drawn from an appropriate posterior predictive distribution ("imputations"). In the second step, the required statistical analysis technique is applied to each of the completed data sets as if it were a completely observed data set. The third step is the pooling step, where the results from the m analyses are combined to form the final results, and allows statistical inferences in the usual way.

Basically, there are two ways of specifying imputation models: Joint modelling and fully conditional specification. The joint modelling approach requires specification of a multivariate distribution for the variables whose values are not observed and drawing imputations from their predictive posterior distribution using Markov Chain Monte Carlo (MCMC) techniques. Within this framework, the most common assumption is that the data is multivariate normally distributed (Schafer, 1997). Other alternatives for categorical data are based on the latent normal model (Albert and Chib, 1993) or the general location model (Little and Rubin, 2002).

This methodology is attractive if the assumed multivariate distribution is an appropriate model for the data but may lack the flexibility needed to deal with complex data sets encountered in applications. In such cases, the joint modelling approach may be too restrictive because the typical specification of a multivariate distribution is not sufficiently flexible to allow different continuous and discrete distributions (He and Raghunathan, 2009). For example, most of the existing model-based methods and software implementations assume that the data originate from a multivariate normal distribution (e.g. Honaker et al., 2011; Templ et al., 2011; van Buuren, 2007). However, the assumption of normality is inappropriate as soon as there are outliers in the data, or in the case of skewed, heavy-tailed or multimodal distributions, potentially leading to deficient results (van Buuren, 2012; He and Raghunathan, 2012).

With the fully conditional specification, also known as multivariate imputation by chained equations (van Buuren and Groothuis-Oudshoorn, 2011), a univariate imputation model is specified for each variable with missing values conditional on other variables of the data set. Starting from initially bootstrapped imputations, subsequent imputations are drawn by iterating over conditional densities (van Buuren and Groothuis-Oudshoorn, 2011; van Buuren, 2007). This framework splits high-dimensional imputation models into multiple one-dimensional problems and is appealing as an alternative to joint modelling in cases where a proper multivariate distribution can not be found. The choice of imputation models in this setting can vary depending on the type of variable to be imputed, for example, parametric models like the Bayesian linear regression or logistic regression. Liu et al. (2013) studied the asymptotic properties of this iterative imputation procedure and provided sufficient conditions under which the imputation distribution converges to the posterior distribution of a joint model when the conditional models are compatible.

De Jong (2012) and de Jong et al. (2014) proposed a new imputation technique based on generalized additive models for location, scale, and shape, GAMLSS, (Rigby and Stasinopoulos, 2005), which is a class of univariate regression models, where the assumption of an exponential family is relaxed and replaced by a general distribution family. This allows a more flexible modelling than standard parametric imputation models, not only based on the location (e.g. the mean), but also the scale (e.g. variance), and the shape (e.g., skewness and kurtosis) of the conditional distribution of the dependent variable to be imputed given all other variables. The R package **ImputeRobust**, described

in the next section, provides the functions necessary to apply the GAMLSS imputation technique to missing data problems. It can be used as standalone tool or in combination with **mice** (van Buuren and Groothuis-Oudshoorn, 2011). Salfran (2018) provides an extensive comparison study of the new package and other Multiple Imputation techniques.

Robust imputation with gamlss and mice

The imputation method realized by the package **ImputeRobust** is based on a generalized additive model for location, scale and shape of the variable to be imputed. Specifically, we adopt the semi-parametric additive formulation of GAMLSS described in Stasinopoulos and Rigby (2007).

Let $Y = (Y_{ij})$, $i = 1, \ldots, n$ and $j = 1, \ldots, p$ be a matrix with n independent observations on p variables and let y_{ij} be the realization of variable Y_j with probability function $f(y_{ij}|\theta_{ij})$ conditional on parameters $\theta_{ij} = (\theta_{ij}^k)$, $k = 1, \ldots, 4$. Assume that g_k are known monotonic link functions relating the distribution parameters θ_{ij}^k to explanatory variables by the following equation:

$$g_k(\theta_{ij}^k) = \eta_k = X_k\beta_k + \sum_{l=1}^{L_k} h_{lk}(x_{lk}), \tag{1}$$

where θ_{ij}^k for $k = 1, 2, 3, 4$ are the location, scale and shape parameters of the distribution, X_k is a fixed known design matrix, β_k^T a vector of linear predictors, and $h_{lk}(x_{lk})$ are unknown smoothing functions of the explanatory variables.

Not all four parameters may be needed, depending on the conditional distribution $f(y_{ij}|\theta_{ij})$ which will be denoted as $\mathcal{D}$. The package **gamlss** (Rigby and Stasinopoulos, 2005) provides a wide range of possible continuous and discrete conditional distributions with varying number of parameters, although not all of these distributions have been adopted yet to create imputations (see Table 1 on section "Main functions").

The proposed imputation method selects the conditional distribution $\mathcal{D}$ for each of the variables to be imputed with the MICE algorithm (van Buuren and Groothuis-Oudshoorn, 2011). This distribution defaults to normal for continuous data, but other alternatives may be chosen. Users can take advantage of this option to restrict imputations to a certain range, e.g., by specifying a truncated normal distribution. Alternatives already included are Logit and Poisson models to handle binary and count data.

The chosen distribution $\mathcal{D}$ defines the type and number of parameters to be modelled, e.g., for the default normal distribution the mean and variance are estimated (individually for each point), but other distributions may require the estimation of the skewness and kurtosis in addition. Adopting models with more parameters increases their flexibility and thus may increase the chance that the imputation procedure is proper in the sense of Rubin (1987). On the other hand, larger sample sizes may be needed to identify the larger number of parameters.

Implementation

The implementation of the imputation method for our simulations uses the **gamlss** package (see Rigby and Stasinopoulos, 2005; Stasinopoulos and Rigby, 2007) in R to fit model (1) based on (penalized) maximum likelihood estimation and adopting the default link functions. Rigby and Stasinopoulos (2005) and Stasinopoulos and Rigby (2007) provide a description of the fitting algorithms used by this package. As smoothing functions h_{jk}, we use P-splines with 20 knots, a piecewise polynomial of degree three, a second order penalty and automatic selection of the smoothing parameter using the Local Maximum Likelihood criterion (see Eilers and Marx, 1996). To prevent abnormal termination of the algorithm, for example if samples are too small, the degree of the polynomial, the order of the penalty, or the stopping time of the fitting algorithm are reduced as a fallback strategies.

Let Y_j be one incompletely observed column of Y. The observed and missing parts of Y_j are denoted by Y_j^{obs} and Y_j^{mis}, respectively. Let $Y_{-j} = (Y_1, \ldots, Y_{j-1}, Y_{j+1}, \ldots, Y_p)$ denote the collection of variables in Y except Y_j. The package **gamlss** does not support Bayesian inference, hence it is not possible to obtain multiple imputations by drawing from the posterior predictive distribution. However, draws from the predictive posterior distribution are approximated by the bootstrap predictive distribution (Harris, 1989):

$$f^\star(Y_j^{mis}|Y_j^{obs}, Y_{-j}) = \int f(Y_j^{mis}|\tilde{\eta}, Y_{-j}^{mis}) f(\tilde{\eta}|\hat{\eta}(Y_j^{obs}, Y_{-j}^{obs}))d\tilde{\eta}\,, \tag{2}$$

where $\tilde{\eta}$ denotes the possible values of the imputation model parameters, $\hat{\eta}(Y_j^{obs}, Y_{-j}^{obs})$ is an estimator

of such parameters, and $f(\tilde{\eta}|\hat{\eta}(Y_j^{obs}, Y_{-j}^{obs}))$ is the sampling distribution of the imputation parameters evaluated at the estimated values. The sampling distribution is simulated with a parametric bootstrap acting as a replacement for the posterior distribution of the imputation parameters. Algorithm 1 describes the imputation process. For a clear presentation, we drop the index i.

Algorithm 1 GAMLSS imputation

1. Fit model $Y_j \sim \mathcal{D}(\hat{\mu}_j, \hat{\sigma}_j, \hat{\eta}_j, \hat{\tau}_j)$ using the observed data $\{Y_j^{obs}, Y_{-j}^{obs}\}$
2. Resample Y_j^{obs} as follows:
$$Y_{j\star}^{obs} \sim \mathcal{D}(\hat{\mu}_j, \hat{\sigma}_j, \hat{\eta}_j, \hat{\tau}_j)\,.$$
Define a bootstrap sample $B = \{Y_{j\star}^{obs}, Y_{-j}^{obs}\}$
3. Refit the above model using B. This leads to adapted estimators $\dot{\mu}_j, \dot{\sigma}_j, \dot{\eta}_j$ and $\dot{\tau}_j$. Draw n_{mis} imputations for Y_j^{mis} as follows:
$$\tilde{Y}_j^{mis} \sim \mathcal{D}(\dot{\mu}_j, \dot{\sigma}_j, \dot{\eta}_j, \dot{\tau}_j)\,.$$
4. Repeat m times steps 2 and 3 to generate m imputed data sets.

Main functions

The two main functions included in the **ImputeRobust** package are the imputing and fitting functions `mice.impute.gamlss()` and `ImpGamlssFit()`, respectively.

The function `ImpGamlssFit()` is internal and its job is to read in the data and model parameters and create a bootstrap predictive sampling function, i.e., it will work through steps 1 to 3 of Algorithm 1. The fitting step depends on the **gamlss** package, and the same control options described in Stasinopoulos and Rigby (2007) can be passed to `ImpGamlssFit()` when calling the `mice()` function. By default, `ImpGamlssFit()` uses P-splines as the smoothers in model (1), which are considered to be stable in the **gamlss** package, but sometimes the fitting algorithm may diverge. The implemented imputation method will catch this event and will try to correct it by automatically cutting down the complexity of the model.

The necessary formula objects for the model are automatically created by the function during execution time. The type of imputation model and its parameters, like the degree of the P-splines, can be controlled with the argument `gam.mod`. Another way of controlling the complexity of the fitting and imputation steps is the `lin.terms` argument. This argument can be used to specify variables by their name, that should enter model (1) linearly.

The default response distribution family used by the fitting and imputation methods is the normal distribution, but it can be modified with the argument `family`. The selected family determines how many parameters are to be modelled. To improve the stability of the imputation method, distributional parameters can be restricted to be the same for all units. The maximum number of parameters to be fitted is controlled by `n.ind.par`, that is, the value of k in equation (1). For example, if the Johnson' SU family (a four parameter continuous distribution) is selected and `n.ind.par = 2`, then the mean and the variance are modelled individually with p-splines, but the shape parameters are restricted to be the same for all units and are modelled as contant values.

The function `mice.impute.gamlss()` has the same structure as the imputation methods included in the **mice** package, meaning that the named method `"gamlss"` can be directly passed as an argument to the `mice()` function. This function also passes all modified default arguments to the fitting function.

Additional functions are included in the package to set the `family` and `n.ind.par` arguments to non-default values. This allows users to mix different gamlss imputation methods within one call to the function `mice()`. The functions are variantes of `mice.impute.gamlss()` where `"gamlss"` is replaced with a method from Table 1 with the syntax `mice.impute.method()`. The name of the function is a reference to the corresponding family from `gamlss.family` (see Stasinopoulos and Rigby, 2007)

Method	Model distribution
gamlssNO	Normal
gamlssBI	Binomial
gamlssGA	Gamma
gamlssJSU	Johnson's SU
gamlssPO	Poisson
gamlssTF	Student's t
gamlssZIBI	Zero inflated Binomial
gamlssZIP	Zero inflated Poisson

Table 1: Included univariate gamlss imputation models.

Usage

The imputation methods provided by **ImputeRobust** add a new semiparametric method to the already included methods in **mice**. This means that it can be used directly with the function `mice()`.

Simple example

As an illustration let us consider an example using the proposed method to estimate the parameters in a linear regression model with multiple imputation. To do this we created a data set with $n = 500$ units composed of four independent variables and one dependent variable with the following code:

```
> set.seed(19394)
> n <- 500
> mu <- rep(0, 4)
> Sigma <- diag(4)
> Sigma[1,2] <- 0.15; Sigma[1,3] <- 0.1; Sigma[1,4] <- -0.1
> Sigma[2,3] <- 0.25; Sigma[2,4] <- 0.05
> Sigma[lower.tri(Sigma)] = t(Sigma)[lower.tri(Sigma)]
> require("MASS")
> rawvars <- mvrnorm(n, mu = mu, Sigma = Sigma)
> pvars <- pnorm(rawvars)
> X.1 <- rawvars[,1]
> X.2 <- qchisq(pvars, 3)[,3]
> X.3 <- qpois(pvars, 2.5)[,2]
> X.4 <- qbinom(pvars, 1, .4)[,4]
> data <- cbind(X.1, X.2, X.3, X.4)
> beta <- c(1.8, 1.3, 1, -1)
> sigma <- 4.2
> y <- data %*% beta + rnorm(n, 0, sigma)
> data <- data.frame(y, data)
```

Thus, we obtain correlated covariates $X.1, \ldots, X.4$ which are random draws from specific distributions, that is, values for $X.1$ are drawn from a standard normal random distribution, values for $X.2$ are drawn from a χ^2 distribution with three degrees of freedom, values for $X.3$ are drawn from a Poisson distribution with parameter $\lambda = 2.5$, and values for $X.4$ come from a Bernoulli distribution with parameter $\pi = 0.4$. The dependent variable, y, is created according to the linear regression model:

$$y_i = \beta_0 + X.1\beta_1 + X.2\beta_2 + X.3\beta_3 + X.4\beta_4 + \epsilon, \quad \epsilon \sim N(0, \sigma^2). \tag{3}$$

The vector of linear predictors, β, and the error variance, σ^2, are chosen so that the coefficient of determination, R^2, is 0.5.

In this first example, we create missing values in $X.2$ to $X.4$ with a monotone MAR mechanism dependent on y and $X.1$ as shown:

```
> r.s <- cbind(y, X.1) %*% c(2,1)
> r.s <- scale(r.s)
> pos <- cut(r.s, quantile(r.s, c(0, .5, 1)), include.lowest=TRUE)
> p.r <- as.numeric(c(.9, .2))
> p.r <- as.vector(p.r[pos])
> R2 <- as.logical(rbinom(length(p.r),1,p.r))
```

```
> r.s <- cbind(y[!R2], X.1[!R2]) %*% c(2,1)
> r.s <- scale(r.s)
> pos <- cut(r.s, quantile(r.s, c(0, .4, 1)), include.lowest=TRUE)
> p.r <- as.numeric(c(.32, .27))
> p.r <- as.vector(p.r[pos])
> R3 <- as.logical(rbinom(length(p.r),1,p.r))
> R4 <- runif(nrow(data[!R2,][!R3,]), 0, 1) >= .25
> data$X.2[!R2] <- NA
> data$X.3[!R2][!R3] <- NA
> data$X.4[!R2][!R3][!R4] <- NA
```

More precisely, to generate missing values in $X.2$ and $X.3$, we first generate variables $r^* = 2y + X.1$. Let Ri be a response indicator, where $Ri = 1$ if $X.i$ is observed and $Ri = 0$ if $X.i$ is not observed. Then missing values in $X.2$ and $X.3$ are generated according to

$$\Pr(R2 = 1|r^*) = \begin{cases} 0.9 & \text{if } r^* \leq r^*_{0.5} \\ 0.2 & \text{else,} \end{cases}, \Pr(R3 = 1|r^*, R2 = 0) = \begin{cases} 0.32 & \text{if } r^* \leq r^*_{0.4} \\ 0.27 & \text{else,} \end{cases}$$

and $\Pr(R3 = 1|R2 = 1) = \Pr(R3 = 1|r^*, R2 = 1) = 1$. Finally, under both conditions, missing values in $X.4$ are generated independently from y and $X.1$ with probability 0.25 for those units for which $X.3$ is not observed. The frequency of missing values and the pattern can be visualized with the **mice** function `md.pattern()`:

```
> library(ImputeRobust)
> md.pattern(data)
    y X.1 X.4 X.3 X.2
276 1   1   1   1   1   0
 66 1   1   1   1   0   1
127 1   1   1   0   0   2
 31 1   1   0   0   0   3
    0   0  31 158 224 413
```

The output is generated by the **mice** package, for details see van Buuren and Groothuis-Oudshoorn (2011). It can be seen that only 276 out of the 500 units are fully observed and that the missing pattern is monotone with the smallest amount of missing values in $X.4$ followed by $X.2$. $X.3$ has the highest amount of missing values.

The imputation task can be performed with a simple call of the `mice()` function:

```
> predictorMatrix <- matrix(c(rep(c(0,0,1,1,1),2), rep(0,5), c(0,0,1,0,0),
+                           c(0,0,1,1,0)), nrow = 5)
> imps <- mice(data, method = "gamlss", predictorMatrix = predictorMatrix,
+              visitSequence = "monotone", maxit = 1, seed = 8913)
 iter imp variable
  1   1  X.4  X.3  X.2
  1   2  X.4  X.3  X.2
  1   3  X.4  X.3  X.2
  1   4  X.4  X.3  X.2
  1   5  X.4  X.3  X.2
```

In this example with a monotone missing pattern, the missing values are imputed in accordance with Rubin (1987, p. 171). The variables are ranked according to the amount of missing values and imputations are drawn starting with the most frequently observed incomplete variable. In the next step, the most frequently observed variable of the remaining incompletely observed variables is imputed. This procedure continues until all variables with missing values are completed, using completely observed but also completed variables to impute the next. In `mice()` the arguments `visitSequence` and `predictorMatrix` control the column order in which incomplete variables are imputed and which variables serve as predictors in each imputation model, respectively. By setting these arguments to non-default values the required order of imputing values is achieved.

The result is an object of class Multiply Imputed Data Set (`mids`) with contents:

```
> print(imps)
Multiply imputed data set
Call:
mice(data = data, method = "gamlss", predictorMatrix = predictorMatrix,
    visitSequence = "monotone", maxit = 1, seed = 8913)
```

```
Number of multiple imputations:  5
Missing cells per column:
  y X.1 X.2 X.3 X.4
  0   0 224 158  31
Imputation methods:
       y      X.1      X.2      X.3      X.4
"gamlss" "gamlss" "gamlss" "gamlss" "gamlss"
VisitSequence:
X.4 X.3 X.2
  5   4   3
PredictorMatrix:
    y X.1 X.2 X.3 X.4
y   0   0   0   0   0
X.1 0   0   0   0   0
X.2 1   1   0   1   1
X.3 1   1   0   0   1
X.4 1   1   0   0   0
Random generator seed value:  8913
```

The argument `method = "gamlss"` in the `mice` function call implies by default, that imputations are drawn assuming a normal distribution for the response variable in the imputation model given all covariables. This has been shown to lead to acceptable results in situations in which even non-plausible values are imputed (de Jong et al., 2014). A useful way of inspecting the distribution of the original and an imputed data is the `stripplot()` function as shown in figure 1.

```
> library(lattice)
> stripplot(imps)
```

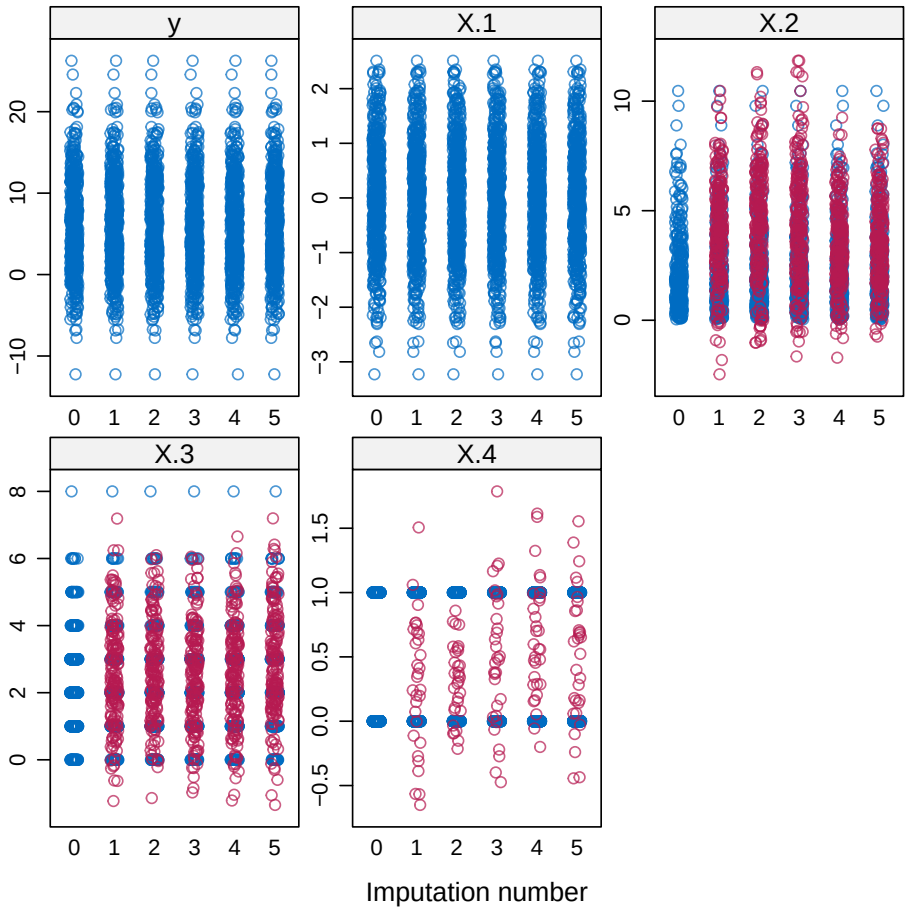


Figure 1: Stripplot of the five variables in the original and the five imputed data sets. Observed data values are blue and imputed data values are red.

The model of interest, as per equation (3), is the linear regression of y on $X.1$, $X.2$, $X.3$, and $X.4$ that created the original data set. The true value of the regression coefficient is `c(1.8,1.3,1,-1)`. The imputed data sets can be analysed as follows:

```
> fit <- with(imps, lm(y ~ X.1 + X.2 + X.3 + X.4))
> print(pool(fit))
Call: pool(object = fit)

Pooled coefficients:
(Intercept)         X.1         X.2         X.3         X.4
  1.0729769   1.7565698   1.1600009   0.6262502  -0.4023920

Fraction of information about the coefficients missing due to nonresponse:
(Intercept)         X.1         X.2         X.3         X.4
  0.4477033   0.5111319   0.9276184   0.7934569   0.2925600

> round(summary(pool(fit)), 2)
             est   se     t    df Pr(>|t|) lo 95 hi 95 nmis  fmi lambda
(Intercept)  1.07 0.57  1.87 22.89     0.07 -0.11  2.26   NA 0.45   0.40
X.1          1.76 0.26  6.79 17.73     0.00  1.21  2.30    0 0.51   0.46
X.2          1.16 0.29  4.00  4.47     0.01  0.39  1.93  224 0.93   0.90
X.3          0.63 0.27  2.33  6.89     0.05 -0.01  1.26  158 0.79   0.74
X.4         -0.40 0.46 -0.87 49.39     0.39 -1.33  0.52   31 0.29   0.26
```

Modifying the imputation model

The default behaviour of the `"gamlss"` method of using a normal distribution can be overridden by setting the argument `family` to any distribution contained in the **gamlss.dist** package (Stasinopoulos et al., 2017). This will define globally the new response distribution for all imputation methods that call the `"gamlss"` method. If the user wants to set different distribution families, to suit the particularities of the variables to be imputed, several functions fully compatible with `mice()` are already included.

In the previous example, variables $X.3$ and $X.4$ were treated as continuous variables in the imputation step, and it was possible for $X.2$ to take on negative values, even though is a strictly positive variable. We will show now how the GAMLSS imputation model can be adjusted to accommodate different types of distributions simultaneously. For the new imputation task, we use a Gamma distribution for $X.2$, a Poisson distribution for the imputation of $X.3$, and a Binomial distribution for $X.4$. The incomplete data set is the same as before, where $X.1$ and y need not to be imputed.

```
> imps <- mice(data, method = c("", "", "gamlssGA", "gamlssPO", "gamlssBI"),
+              seed = 8913)
 iter imp variable
  1   1  X.2  X.3  X.4
  1   2  X.2  X.3  X.4
  1   3  X.2  X.3  X.4
  1   4  X.2  X.3  X.4
  1   5  X.2  X.3  X.4
  2   1  X.2  X.3  X.4
  2   2  X.2  X.3  X.4
  ...
```

Figure 2 shows the effect of the changes in the response distribution for the imputation model. Choosing the Gamma distribution for $X.2$, the Poisson distribution for $X.3$ and the Binomial distribution for $X.4$ imputes realistic values as compared to, e.g., choosing the normal distribution. Whether to favor the imputation of realistic values or not, is a different problem. Since the final goal is statistical validity of MI based estimation (Rubin, 1996), in some applications it may be better to allow for the imputation of "unrealistic" while using a flexible model, for example, impute continuous values when the variable to be imputed is a count variable (see de Jong, 2012; Salfran, 2018).

```
> stripplot(imps)
```

The model of interest in this example is the same as in section 14.4.1. The results of running the analysis are slightly different due to choosing different imputation models.

```
> fit <- with(imps, lm(y ~ X.1 + X.2 + X.3 + X.4))
> round(summary(pool(fit)), 2)
```

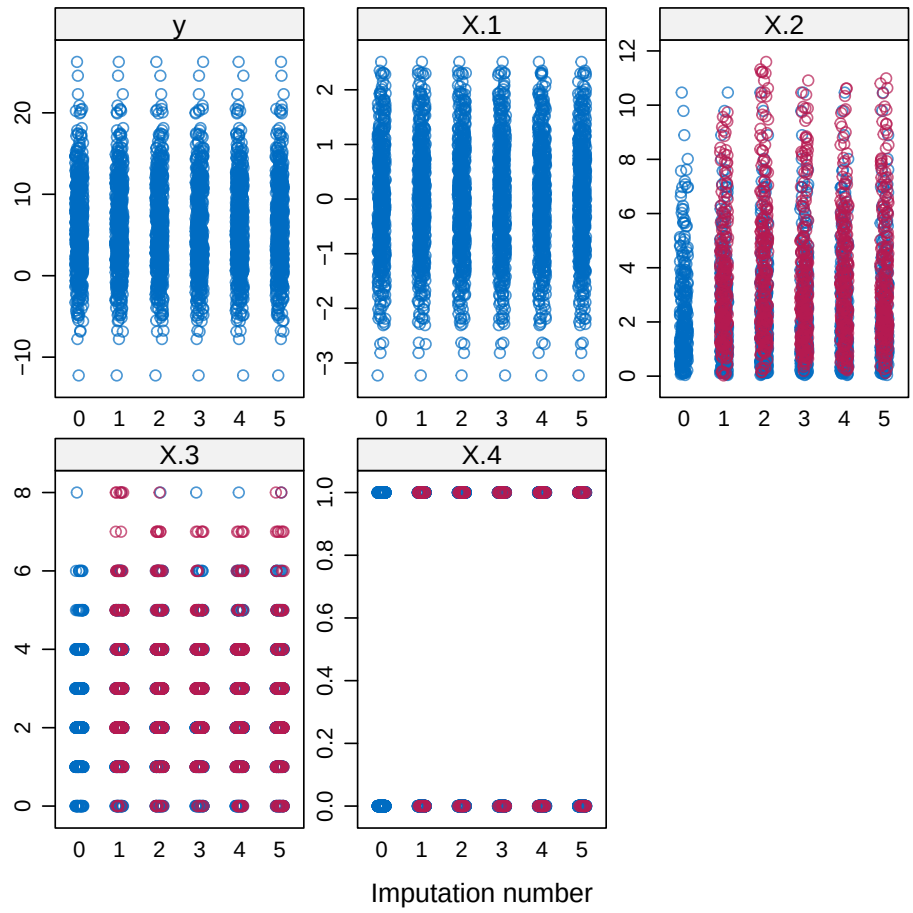


Figure 2: Stripplot of the five variables in the original data and the five imputed data sets. Observed data values are blue and imputed data values are red.

```
              est   se     t     df Pr(>|t|) lo 95 hi 95 nmis  fmi lambda
(Intercept)  0.78 0.48  1.61  40.46     0.11 -0.20  1.76   NA 0.33   0.30
X.1          1.78 0.25  7.27  18.00     0.00  1.27  2.30    0 0.51   0.46
X.2          1.19 0.13  8.89  10.43     0.00  0.90  1.49  224 0.66   0.60
X.3          0.70 0.21  3.37   8.90     0.01  0.23  1.18  158 0.71   0.65
X.4         -0.32 0.40 -0.80 127.39     0.43 -1.12  0.48   31 0.16   0.15
```

Chronic kidney disease data

A real world example data set was retrieved from the Machine Learning Database Repository at the University of California, Irvine (Lichman, 2013). The dataset contains 400 observations from which some variables were selected. The data already contained 149 rows where some values were missing. This example illustrates some of the specific extra arguments that can be passed to the imputation method, mainly with the objective of controlling the speed of the `gamlss()` function.

```
> library(RWeka)
> data <- read.arff("data/chronic_kidney_disease_full.arff")
> data <- data[,c("age", "bp", "bgr", "bu", "sc", "sod", "pot", "hemo",
+              "class")]
> data$class <- ifelse(data$class == "ckd", 1, 0)
```

The missing data pattern is non-monotone. The fully conditional specification approach behind **mice** will handle this case. Running `mice` with the default fitting arguments of `gamlss` in complex data sets like this would take a long time. There are several ways in which the speed of the imputation can be adjusted. One alternative is by changing the values controlling the GAMLSS fitting algorithm. The arguments `n.cyc`, `bf.cyc`, `cyc` control respectively the number of cycles of the outer, backfitting, and inner algorithms of gamlss (Rigby and Stasinopoulos, 2005; Stasinopoulos and Rigby, 2007). Other

arguments that can be changed are the convergence criterions of the outer and inner algorithms with the `c.crit` or `cc` arguments, respectively.

The choice of imputation model can also be modified by the user. The relevant argument is `gam.mod`. The default imputation model is a P-spline with two degrees of freedom and second order differences, this amounts to `gam.mod = list(type = "pb",par = list(degree = 2,order = 2)`. The degrees and order can be changed. Also, the model can be set to be linear using `type = "linear"`.

The following code shows the result of imputing missing values of the chronic kidney disease data set. With the current parameters and without any other optimization this took 10 minutes computing time with a Core i7-3520M CPU.

```
> meth <- c("gamlssJSU", "gamlssJSU", "gamlssJSU", "gamlssJSU", "gamlssJSU",
+           "gamlssJSU", "gamlssJSU", "gamlssJSU", "gamlss")
> imps <- mice(data, method = meth, n.cyc = 3, bf.cyc = 2, cyc = 2,
+              maxit = 5, m = 5, seed = 8901,
+              gam.mod = list(type = "linear"))
 iter imp variable
  1   1  age  bp  bgr  bu  sc  sod  pot  hemo
  1   2  age  bp  bgr  bu  sc  sod  pot  hemo
  1   3  age  bp  bgr  bu  sc  sod  pot  hemo
  1   4  age  bp  bgr  bu  sc  sod  pot  hemo
  1   5  age  bp  bgr  bu  sc  sod  pot  hemo
  2   1  age  bp  bgr  bu  sc  sod  pot  hemo
  2   2  age  bp  bgr  bu  sc  sod  pot  hemo
  ...
```

Figure 3 shows that even with an extreme simplification of the fitting algorithm of `gamlss`. The imputation method still manages to produce acceptable values.

```
> stripplot(imps)
```

We used the imputed data set to fit a logistic regression in which we modelled the probability of having chronic kidney disease as a function of age and some other blood related information.

```
> fit <- with(imps, glm(class ~ age + bp + bgr + sc + sod + pot + hemo,
+                       family=binomial(link='logit')))
> round(summary(pool(fit)), 2)
             est    se     t     df Pr(>|t|) lo 95 hi 95 nmis  fmi lambda
(Intercept) 33.04 12.85  2.57  79.40     0.01  7.47 58.61   NA 0.21   0.19
age         -0.03  0.02 -1.16  33.22     0.25 -0.07  0.02    9 0.36   0.32
bp           0.09  0.04  2.45 134.55     0.02  0.02  0.16   12 0.15   0.13
bgr          0.03  0.01  2.29  12.24     0.04  0.00  0.07   44 0.61   0.55
sc           4.27  1.59  2.69  14.12     0.02  0.87  7.67   17 0.57   0.51
sod         -0.15  0.09 -1.62  45.86     0.11 -0.34  0.04   87 0.30   0.27
pot         -0.68  0.38 -1.82  45.99     0.08 -1.44  0.07   88 0.30   0.27
hemo        -1.72  0.37 -4.65  30.81     0.00 -2.48 -0.97   52 0.38   0.34
```

Conclusion

The imputation method based on gamlss is a fairly new imputation technique which is provided to properly handle situations in which fully parametric assumptions with respect to the conditional distribution of the variable to be imputed is questionable. The technique is based on the idea of attaining imputations avoiding possibly misspecified imputation models. Salfran (2018), de Jong et al. (2014) and de Jong (2012) showed that this technique outperforms standard imputation techniques in several scenarios. The **ImputeRobust** package is a step forward in the development of an imputation method that requires weak distributional assumptions, but still allows valid inference. It extends the set of applications of the existing GAMLSS package by allowing it to interact with the **mice** package.

By building on the popular **mice** package, the advantages of standard and robust imputation procedures are available in a user friendly way. Further research will focus on improving the efficiency of this semi-parametric imputation technique, requiring only weak assumptions for generating multiple imputations but still allowing valid inferences.

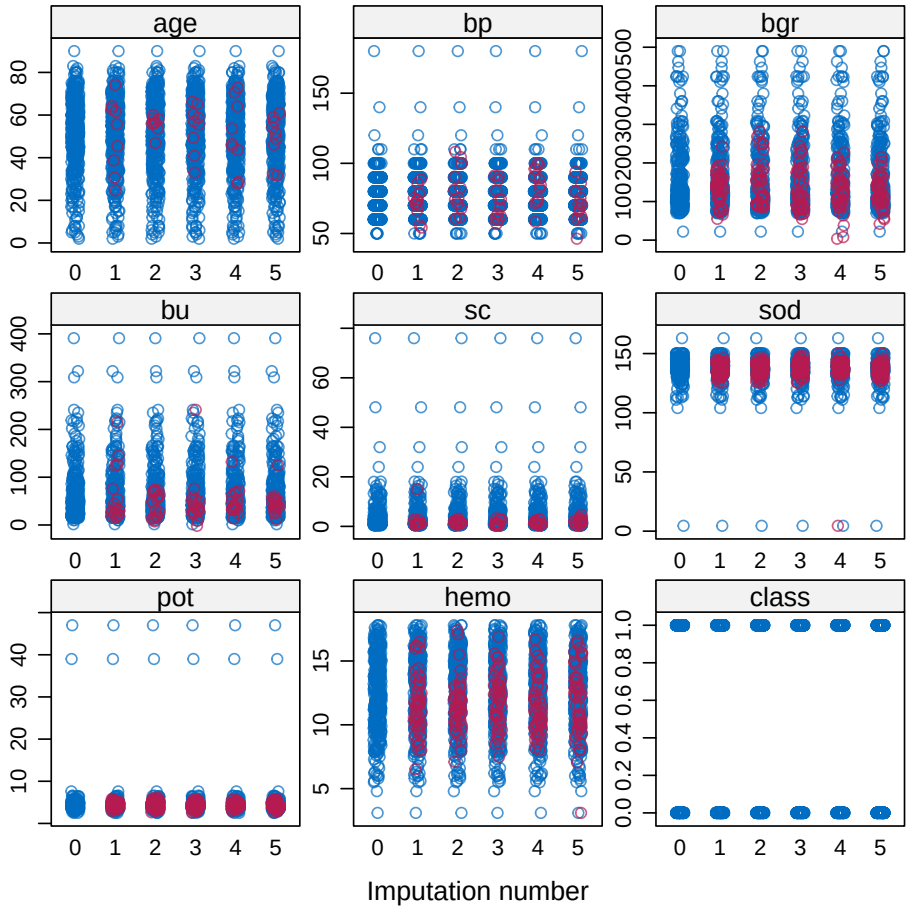


Figure 3: Stripplot of the chronic kidney data set variables with missings and five imputed data sets. Observed data is in blue and imputed data in red.

Acknowledgements

The authors gratefully acknowledge financial support from the German Science Foundation via grant SP 930/8-1.

Bibliography

J. H. Albert and S. Chib. Bayesian analysis of binary and polychotomous response data. *Journal of the American Statistical Association*, 88(422):669–679, 1993. ISSN 1537-274X. URL https://doi.org/10.1080/01621459.1993.10476321. [p61]

R. de Jong. *Robust Multiple Imputation*. PhD thesis, Universität Hamburg, 2012. URL http://ediss.sub.uni-hamburg.de/volltexte/2012/5971/. [p61, 67, 69]

R. de Jong, S. van Buuren, and M. Spiess. Multiple imputation of predictor variables using generalized additive models. *Communications in Statistics - Simulation and Computation*, 45(3):968–985, 2014. ISSN 1532-4141. URL https://doi.org/10.1080/03610918.2014.911894. [p61, 66, 69]

P. H. C. Eilers and B. D. Marx. Flexible smoothing with b -splines and penalties. *Statistical Science*, 11 (2):89–121, 1996. ISSN 0883-4237. URL https://doi.org/10.1214/ss/1038425655. [p62]

I. R. Harris. Predictive fit for natural exponential families. *Biometrika*, 76(4):675–684, 1989. ISSN 1464-3510. URL https://doi.org/10.1093/biomet/76.4.675. [p62]

Y. He and T. E. Raghunathan. On the Performance of Sequential Regression Multiple Imputation Methods with Non Normal Error Distributions. *Communications in Statistics - Simulation and Computation*, 38(4):856–883, 2009. ISSN 0361-0918. URL https://doi.org/10.1080/03610910802677191. [p61]

Y. He and T. E. Raghunathan. Multiple imputation using multivariate gh transformations. *Journal of Applied Statistics*, 39(10):2177–2198, 2012. ISSN 0266-4763. URL https://doi.org/10.1080/02664763.2012.702268. [p61]

J. Honaker, G. King, and M. Blackwell. Amelia II: A Program for Missing Data. *Journal of Statistical Software*, 45(7):1–47, 2011. ISSN 1548-7660. URL https://doi.org/10.18637/jss.v045.i07. [p61]

M. Lichman. UCI machine learning repository, 2013. URL http://archive.ics.uci.edu/ml. [p68]

R. J. A. Little and D. B. Rubin. *Statistical Analysis with Missing Data*. John Wiley & Sons, 2002. ISBN 9780471183860. URL https://doi.org/10.1002/9781119013563. [p61]

J. Liu, A. Gelman, J. Hill, Y.-S. Su, and J. Kropko. On the stationary distribution of iterative imputations. *Biometrika*, 101(1):155–173, 2013. ISSN 1464-3510. URL https://doi.org/10.1093/biomet/ast044. [p61]

R. A. Rigby and D. M. Stasinopoulos. Generalized additive models for location, scale and shape (with discussion). *Journal of the Royal Statistical Society C*, 54(3):507–554, 2005. ISSN 1467-9876. URL https://doi.org/10.1111/j.1467-9876.2005.00510.x. [p61, 62, 68]

D. B. Rubin. *Multiple Imputation for Nonresponse in Surveys*. John Wiley & Sons, 1987. ISBN 978-0-471-65574-9. URL https://doi.org/10.1002/9780470316696. [p61, 62, 65]

D. B. Rubin. Multiple imputation after 18+ years. *Journal of the American Statistical Association*, 91(434):473–489, 1996. ISSN 1537-274X. URL https://doi.org/10.1080/01621459.1996.10476908. [p67]

D. Salfran. *Multiple Imputation for Complex Data Sets*. PhD thesis, Universität Hamburg, 2018. URL http://ediss.sub.uni-hamburg.de/volltexte/2018/9058/. [p62, 67, 69]

J. Schafer. *Analysis of Incomplete Multivariate Data*. Chapman & Hall, 1997. ISBN 9781439821862. URL https://doi.org/10.1201/9781439821862. [p61]

D. M. Stasinopoulos and R. A. Rigby. Generalized additive models for location scale and shape (gamlss). *Journal of Statistical Software*, 23(7), 2007. ISSN 1548-7660. URL https://doi.org/10.18637/jss.v023.i07. R package version 5.0-2. [p62, 63, 68]

M. Stasinopoulos, B. R. with contributions from Calliope Akantziliotou, G. Heller, R. Ospina, N. Motpan, F. McElduff, V. Voudouris, M. Djennad, M. Enea, A. Ghalanos, and C. Argyropoulos. *Gamlss.dist: Distributions for Generalized Additive Models for Location Scale and Shape*, 2017. URL https://CRAN.R-project.org/package=gamlss.dist. R package version 5.0-2. [p67]

M. Templ, A. Kowarik, and P. Filzmoser. Iterative stepwise regression imputation using standard and robust methods. *Computational Statistics & Data Analysis*, 55(10):2793–2806, 2011. ISSN 0167-9473. URL https://doi.org/10.1016/j.csda.2011.04.012. [p61]

S. van Buuren. Multiple imputation of discrete and continuous data by fully conditional specification. *Statistical Methods in Medical Research*, 16(3):219–242, 2007. ISSN 1477-0334. URL https://doi.org/10.1177/0962280206074463. [p61]

S. van Buuren. *Flexible Imputation of Missing Data*. Chapman and Hall/CRC, 2012. ISBN 9781439868256. URL https://doi.org/10.1201/b11826. [p61]

S. van Buuren and K. Groothuis-Oudshoorn. Mice: Multivariate imputation by chained equations. *Journal of Statistical Software*, 45(3), 2011. ISSN 1548-7660. URL https://doi.org/10.18637/jss.v045.i03. R package version 2.30. [p61, 62, 65]

Daniel Salfran
Psychological Methods and Statistics
Faculty of Psychology and Movement Science
Universität Hamburg
Von-Melle-Park 5, 20146 Hamburg
Germany
danielsalfran@gmail.com

Martin Spiess
Psychological Methods and Statistics
Faculty of Psychology and Movement Science
Universität Hamburg

Von-Melle-Park 5, 20146 Hamburg
Germany
martin.spiess@uni-hamburg.de

MGLM: An R Package for Multivariate Categorical Data Analysis

by Juhyun Kim, Yiwen Zhang, Joshua Day, Hua Zhou

Abstract Data with multiple responses is ubiquitous in modern applications. However, few tools are available for regression analysis of multivariate counts. The most popular multinomial-logit model has a very restrictive mean-variance structure, limiting its applicability to many data sets. This article introduces an R package **MGLM**, short for multivariate response generalized linear models, that expands the current tools for regression analysis of polytomous data. Distribution fitting, random number generation, regression, and sparse regression are treated in a unifying framework. The algorithm, usage, and implementation details are discussed.

Introduction

Multivariate categorical data arises in many fields, including genomics, image analysis, text mining, and sports statistics. The multinomial-logit model (Agresti, 2002, Chapter 7) has been the most popular tool for analyzing such data. However, it is limiting due to its specific mean-variance structure and the strong assumption that the counts are negatively correlated. Models that address over-dispersion relative to a multinomial distribution and incorporate positive and/or negative correlation structures would offer greater flexibility for analysis of polytomous data.

In this article, we introduce an R package **MGLM**, short for multivariate response generalized linear models. The **MGLM** package provides a unified framework for random number generation, distribution fitting, regression, hypothesis testing, and variable selection for multivariate response generalized linear models, particularly four models listed in Table 1. These models considerably broaden the class of generalized linear models (GLM) for analysis of multivariate categorical data.

MGLM overlaps little with existing packages in R and other softwares. The standard multinomial-logit model is implemented in several R packages (Venables and Ripley, 2002) with **VGAM** (Yee, 2010, 2015, 2017) being the most comprehensive. We include the classical multinomial-logit regression model in **MGLM** not only for completeness, but also to complement it with various penalty methods for variable selection and regularization. If invoked by the group penalty, **MGLM** is able to perform variable selection at the predictor level for easier interpretation. This is different from the elastic net penalized multinomial-logit model implemented in **glmnet** (Friedman et al., 2010), which does selection at the matrix entry level. Although **MGLM** focuses on regression, it also provides distribution fitting and random number generation for the models listed in Table 1. **VGAM** and **dirmult** (Tvedebrink, 2010) packages can estimate the parameters of the Dirichlet-multinomial (DM) distribution using Fisher's scoring and Newton's method respectively. As indicated in the manual (Yee, 2017), the convergence of Fisher's scoring method may be slow due to the difficulty in evaluating the expected information matrix. Further the Newton's method is unstable as the log-likelihood function may be non-concave. As explained later, **MGLM** achieves both stability and efficiency via a careful algorithmic design. In SAS, `PROC LOGISTIC` can fit multinomial-logit model. In Matlab, the `mnrfit` function fits multinomial-logit regression. Alternative link functions (probit, loglog, complementary loglog) are implemented only for ordinal responses. Other regression models in Table 1 are not implemented in either SAS or Matlab.

There are some limitations to the **MGLM**. First, **MGLM** only handles nominal responses; ordinal responses are not incorporated in current implementation. **MGLM** also does not allow covariates to take a different value for each category, which can be useful in applications such as modeling consumer choice among a discrete number of products (Yee, 2015, Chapter 14). Lastly, current implementation of **MGLM** does not permit parameter constraints.

MGLM provides standard errors for all estimates, reports significance of regression covariates based on the Wald test (default) or the likelihood ratio test (LRT), and outputs the AIC (Akaike information criterion) and BIC (Bayesian information criterion) of the fitted model to aid model choice by users. Model selection via regularization is automated by computing the solution path on a grid of tuning parameter values and then reporting the regularized estimate with minimal BIC. With large data sets in mind, we pay particular attention to computational efficiency. For numerical examples in this article, we report the run times whenever possible. The results are obtained on a laptop with Intel Core i7 CPU @ 2.9GHz and 16G memory using **MGLM** 0.1.0 under R 3.4.3.

Model	No. regress. param.	Correlation	**MGLM** code
Multinomial	$p(d-1)$	Negative	`dist='MN'`
Dirichlet-multinomial	pd	Negative	`dist='DM'`
Negative multinomial	$p(d+1)$ or $pd+1$	Positive	`dist='NegMN'`
Gen. Dirichlet-multinomial	$2p(d-1)$	Negative and positive	`dist='GDM'`

Table 1: Multivariate generalized linear model implemented in the **MGLM** package. d is the number of categories and p is the number of predictors in the regression model.

Multivariate generalized linear models (MGLM)

This section details the models implemented in **MGLM**. Table 1 summarizes the multivariate models implemented in the R package. They are multivariate analogs of binomial, beta-binomial, and negative binomial models. For each model, we specify the probability mass function of the response vector $\mathbf{y}$, the link function that relates distribution parameters to covariates, and the log-likelihood function of a finite sample. We start from the classical multinomial-logit model.

Multinomial (MN) model

The probability mass function of a d dimensional multinomial sample $\mathbf{y} = (y_1, \ldots, y_d)^T$ with batch size $m = \sum_{j=1}^d y_j$ and parameter $\boldsymbol{p} = (p_1, \ldots, p_d)$ is

$$f(\mathbf{y}|\boldsymbol{p}) = \binom{m}{\mathbf{y}} \prod_{j=1}^d p_j^{y_j}.$$

The parameter $\boldsymbol{p}$ is linked to the covariate vector $\mathbf{x} \in \mathbb{R}^p$ via the multinomial-Poisson transformation (Baker, 1994; Lang, 1996)

$$p_j = \frac{e^{\mathbf{x}^T \boldsymbol{\beta}_j}}{\sum_{j'} e^{\mathbf{x}^T \boldsymbol{\beta}_{j'}}}, \quad j = 1, \ldots, d,$$

where $\boldsymbol{\beta}_1, \ldots, \boldsymbol{\beta}_d \in \mathbb{R}^p$ are the regression coefficients. For identifiability, we constrain $\boldsymbol{\beta}_d = \mathbf{0}$ and only estimate $\boldsymbol{\beta}_1, \ldots, \boldsymbol{\beta}_{d-1}$, which are collected in the regression coefficient matrix $\boldsymbol{B} \in \mathbb{R}^{p \times (d-1)}$. Given independent observations $(\mathbf{y}_i, \mathbf{x}_i)$, $i = 1, \ldots, n$, the log-likelihood is

$$\ell(\boldsymbol{B}) = \sum_{i=1}^n \sum_{j=1}^d y_{ij} \left(\mathbf{x}_i^T \boldsymbol{\beta}_j - \ln \sum_{j'=1}^d e^{\mathbf{x}_i^T \boldsymbol{\beta}_{j'}} \right) + \sum_{i=1}^n \ln \binom{m_i}{\mathbf{y}_i}.$$

Because the log-sum-exponential mapping $(\eta_1, \ldots, \eta_d)^T \mapsto \ln \sum_j e^{\eta_j}$ is convex (Boyd and Vandenberghe, 2004, p72), the log-likelihood function is concave. This nice feature makes maximum likelihood estimation relatively easy for multinomial-logit model. Unfortunately, convexity is lost in other models listed in Table 1.

Dirichlet-multinomial (DM) model

The mean-variance structure of the MN model does not allow over-dispersion, which is common in real data. DM distribution models the probability parameter $\boldsymbol{p}$ in the MN model by a Dirichlet distribution. The probability mass of a d-category count vector $\mathbf{y}$ over $m = \sum_j y_j$ trials under DM with parameter $\boldsymbol{\alpha} = (\alpha_1, \ldots, \alpha_d)$, $\alpha_j > 0$ and proportion vector $\mathbf{p} \in \Delta_d = \{(p_1, \ldots, p_d) : p_j \geq 0, \sum_j p_j = 1\}$ is

$$
\begin{aligned}
f(\mathbf{y}|\boldsymbol{\alpha}) &= \int_{\Delta_d} \binom{m}{\mathbf{y}} \prod_j p_j^{y_j} \frac{\Gamma(\sum_j \alpha_j)}{\prod_j \Gamma(\alpha_j)} \prod_j p_j^{\alpha_j - 1} \, d\boldsymbol{p} \\
&= \binom{m}{\mathbf{y}} \prod_{j=1}^{d} \frac{\Gamma(\alpha_j + y_j)}{\Gamma(\alpha_j)} \frac{\Gamma(\sum_j \alpha_j)}{\Gamma(\sum_j \alpha_j + \sum_j y_j)} \\
&= \binom{m}{\mathbf{y}} \frac{\prod_{j=1}^{d} (\alpha_j)_{y_j}}{(\sum_j \alpha_j)_m},
\end{aligned}
$$

where $(a)_k = a(a+1)\cdots(a+k-1)$ for nonnegative integer k denotes the rising factorial. The last equality is due to the identity $\Gamma(a+k)/\Gamma(a) = (a)_k$ (Graham et al., 1994). Because the data y_j and parameter α_j are intertwined in the gamma (or rising factorial) terms and do not factorize, the DM distribution does *not* belong to the exponential family. To incorporate covariates, the inverse link function

$$
\alpha_j = e^{\mathbf{x}^T \boldsymbol{\beta}_j}, \quad j = 1, \ldots, d
$$

relates the parameter $\boldsymbol{\alpha} = (\alpha_1, \ldots, \alpha_d)$ of the DM distribution to the covariates $\mathbf{x}$. The log-likelihood for independent data points $(\mathbf{y}_i, \mathbf{x}_i)$, $i = 1, \ldots, n$, takes the form

$$
\ell(\boldsymbol{B}) = \sum_{i=1}^{n} \sum_{j=1}^{d} \sum_{k=0}^{y_{ij}-1} \ln\left(e^{\mathbf{x}_i^T \boldsymbol{\beta}_j} + k\right) - \sum_{i=1}^{n} \sum_{k=0}^{m_i - 1} \ln\left(\sum_{j=1}^{d} e^{\mathbf{x}_i^T \boldsymbol{\beta}_j} + k\right) + \sum_{i=1}^{n} \ln \binom{m_i}{\mathbf{y}_i}
$$

with $\boldsymbol{B} = (\boldsymbol{\beta}_1, \ldots, \boldsymbol{\beta}_d) \in \mathbb{R}^{p \times d}$ collecting all regression coefficients. The log-likelihood, as a difference of two convex terms, is not concave in general.

Negative multinomial (NegMN) model

The counts Y_j in both MN and DM models are negatively correlated, restricting their use for counts with positive correlation. The NegMN distribution provides a natural model for positively correlated count data. The probability mass of a count vector $\mathbf{y}$ under a NegMN distribution with parameter $(p_1, \ldots, p_{d+1}, \phi)$, $\sum_{j=1}^{d+1} p_j = 1$, $p_j, \phi > 0$, is

$$
f(\mathbf{y}|\boldsymbol{p}, \phi) = \binom{\phi + m - 1}{m} \binom{m}{\mathbf{y}} \prod_{j=1}^{d} p_j^{y_j} p_{d+1}^{\phi} = \frac{(\phi)_m}{m!} \binom{m}{\mathbf{y}} \prod_{j=1}^{d} p_j^{y_j} p_{d+1}^{\phi},
$$

where $\binom{\phi+m-1}{m} = \frac{(\phi)_m}{m!}$ is the general binomial coefficient. The parameter ϕ and the data m do not factorize. Therefore, NegMN does *not* belong to the exponential family when ϕ is unknown. We use the inverse link functions

$$
p_j = \frac{e^{\mathbf{x}^T \boldsymbol{\alpha}_j}}{1 + \sum_{j=1}^{d} e^{\mathbf{x}^T \boldsymbol{\alpha}_j}}, 1 \le j \le d, \quad p_{d+1} = \frac{1}{1 + \sum_{j=1}^{d} e^{\mathbf{x}^T \boldsymbol{\alpha}_j}}, \quad \phi = e^{\mathbf{x}^T \boldsymbol{\beta}}
$$

to relate the covariates $\mathbf{x}$ to distribution parameter $(p_1, \ldots, p_{d+1}, \phi)$. Let $\boldsymbol{B} = (\boldsymbol{\alpha}_1, \ldots, \boldsymbol{\alpha}_d, \boldsymbol{\beta}) \in \mathbb{R}^{p \times (d+1)}$ collect all the regression coefficients. Given n independent data points $(\mathbf{y}_i, \mathbf{x}_i)$, $i = 1, \ldots, n$, the log-likelihood is

$$
\begin{aligned}
\ell(\boldsymbol{B}) = &\sum_{i=1}^{n} \sum_{k=0}^{m_i - 1} \ln\left(e^{\mathbf{x}_i^T \boldsymbol{\beta}} + k\right) - \sum_{i=1}^{n} \left(e^{\mathbf{x}_i^T \boldsymbol{\beta}} + m_i\right) \ln\left(\sum_{j=1}^{d} e^{\mathbf{x}_i^T \boldsymbol{\alpha}_j} + 1\right) \\
&+ \sum_{i=1}^{n} \sum_{j=1}^{d} y_{ij} \mathbf{x}_i^T \boldsymbol{\alpha}_j - \sum_{i=1}^{n} \sum_{j=1}^{d} \ln y_{ij}!,
\end{aligned}
$$

which in general is not concave in $\boldsymbol{\alpha}_j$ and $\boldsymbol{\beta}$.

In some applications the over-dispersion parameter ϕ may not depend on the covariates $\mathbf{x}$. **MGLM** offers the option to model the responses $\mathbf{y}_i$ to share a common dispersion parameter ϕ, without linking it to the covariates. In this case, the log-likelihood is

$$\ell(\boldsymbol{\alpha}_1,\ldots,\boldsymbol{\alpha}_d,\phi) = \sum_{i=1}^{n}\sum_{k=0}^{m_i-1}\ln(\phi+k) - \sum_{i=1}^{n}(\phi+m_i)\ln\left(\sum_{j=1}^{d}e^{\mathbf{x}_i^T\boldsymbol{\alpha}_j}+1\right) + \sum_{i=1}^{n}\sum_{j=1}^{d}y_{ij}\mathbf{x}_i^T\boldsymbol{\alpha}_j - \sum_{i=1}^{n}\sum_{j=1}^{d}\ln y_{ij}!.$$

The model size is $pd+1$ and **MGLM** outputs the estimates $(\hat{\boldsymbol{\alpha}}_1,\ldots,\hat{\boldsymbol{\alpha}}_d,\hat{\phi})$ and their standard errors.

Generalized Dirichlet-multinomial (GDM) model

In the previous three models, the multivariate counts have either pairwise negative correlation (MN and DM) or pairwise positive correlation (NegMN). It is possible to relax these restrictions by choosing a more flexible mixing distribution for the probability vector $\boldsymbol{p}$ in MN model. Connor and Mosimann (1969) suggest a generalization of the Dirichlet distribution that meets this challenge. The resulting admixed distribution, called the GDM distribution, provides a flexible model for multivariate categorical data with a general correlation structure (Bouguila, 2008; Zhou and Lange, 2010).

The probability mass of a count vector $\mathbf{y}$ over $m = \sum_j y_j$ trials under GDM with parameter $(\boldsymbol{\alpha},\boldsymbol{\beta}) = (\alpha_1,\ldots,\alpha_{d-1},\beta_1,\ldots,\beta_{d-1})$, $\alpha_j,\beta_j>0$, is

$$\begin{aligned} f(\mathbf{y}|\boldsymbol{\alpha},\boldsymbol{\beta}) &= \binom{m}{\mathbf{y}}\prod_{j=1}^{d-1}\frac{\Gamma(\alpha_j+y_j)}{\Gamma(\alpha_j)}\frac{\Gamma(\beta_j+z_{j+1})}{\Gamma(\beta_j)}\frac{\Gamma(\alpha_j+\beta_j)}{\Gamma(\alpha_j+\beta_j+z_j)} \\ &= \binom{m}{\mathbf{y}}\prod_{j=1}^{d-1}\frac{(\alpha_j)_{y_j}(\beta_j)_{z_{j+1}}}{(\alpha_j+\beta_j)_{z_j}}, \end{aligned} \tag{1}$$

where $z_j = \sum_{k=j}^{d} y_k$. Again it is clear that the GDM distribution does not belong to the exponential family, since the parameter α_j,β_j and data y_j,z_j do not factorize.

We propose the inverse link functions

$$\alpha_j = e^{\mathbf{x}^T\boldsymbol{\alpha}_j},\quad \beta_j = e^{\mathbf{x}^T\boldsymbol{\beta}_j},\quad 1\le j\le d-1,$$

to relate the covariates $\mathbf{x}$ and parameter $(\alpha_1,\ldots,\alpha_{d-1},\beta_1,\ldots,\beta_{d-1})$ of the GDM distribution (1). Let $\boldsymbol{B} = (\boldsymbol{\alpha}_1,\ldots,\boldsymbol{\alpha}_{d-1},\boldsymbol{\beta}_1,\ldots,\boldsymbol{\beta}_{d-1}) \in \mathbb{R}^{p\times 2(d-1)}$ collect all the regression coefficients. Given n independent data points $(\mathbf{y}_i,\mathbf{x}_i)$, the log-likelihood is

$$\ell(\boldsymbol{B}) = \sum_{i=1}^{n}\sum_{j=1}^{d-1}\left[\sum_{k=0}^{y_{ij}-1}\ln\left(e^{\mathbf{x}_i^T\boldsymbol{\alpha}_j}+k\right) + \sum_{k=0}^{z_{i,j+1}-1}\ln\left(e^{\mathbf{x}_i^T\boldsymbol{\beta}_j}+k\right) - \sum_{k=0}^{z_{ij}-1}\ln\left(e^{\mathbf{x}_i^T\boldsymbol{\alpha}_j}+e^{\mathbf{x}_i^T\boldsymbol{\beta}_j}+k\right)\right] + \sum_{i=1}^{n}\ln\binom{m_i}{\mathbf{y}_i}.$$

Again the log-likelihood is not concave in general.

Which model to use?

When there are no covariates, multinomial model is a special case of the DM models, which in turn is a sub-model of the GDM model. That is, MN $\subset$ DM $\subset$ GDM. The distribution fitting function `MGLMfit` reports the p-value of the LRT for comparing the fitted model with the most commonly used multinomial model. NegMN model does not have such a relationship with any of the other three models. Therefore, no LRT is performed when `dist="NegMN"` in the distribution fitting function `MGLMfit`.

For regression, there is no nesting structure among the models in Table 1. The regression function `MGLMreg` outputs AIC and BIC of the fitted model to aid users in choosing an appropriate regression model for their data.

Standard errors and testing

Standard errors for both distribution fitting (`MGLMfit`) and regression estimates (`MGLMreg`) are calculated based on the observed information matrix, as it provides a reasonable approximation to the expected information matrix and is even preferred as argued by Efron and Hinkley (1978).

Unlike regression for univariate responses, the **MGLM** regression parameter is a matrix $\boldsymbol{B} = (\boldsymbol{\beta}_1, \ldots, \boldsymbol{\beta}_{d_e}) \in \mathbb{R}^{p \times d_e}$ with each row $\boldsymbol{B}_{k,\cdot}$ corresponding to the effects of one predictor. Here $d_e = d - 1$ (MN), d (DM), $d + 1$ (NegMN), or $2(d - 1)$ (GDM). Therefore, the hypotheses for testing the significance of the k-th covariate are:

$$H_0 : \|\boldsymbol{B}_{k,\cdot}^T\|_2 = 0 \quad \text{vs} \quad H_a : \|\boldsymbol{B}_{k,\cdot}^T\| \neq 0.$$

By default, `MGLMreg` assesses the significance of each predictor by the Wald test. Let

$$\widehat{\boldsymbol{\Sigma}} = \boldsymbol{I}_{\text{obs}}^{-1}(\widehat{\boldsymbol{B}}) = [-\nabla^2 \ell(\widehat{\boldsymbol{B}})]^{-1} \in \mathbb{R}^{pd_e \times pd_e}$$

be the inverse of the observed information matrix at the maximum likelihood estimate (MLE) $\widehat{\boldsymbol{B}}$. Then the Wald statistic is computed as

$$W_k = \widehat{\boldsymbol{B}}_{k,\cdot} \widehat{\boldsymbol{\Sigma}}_{k,k} \widehat{\boldsymbol{B}}_{k,\cdot}^T,$$

where $\widehat{\boldsymbol{\Sigma}}_{k,k}$ is the sub-matrix of $\widehat{\boldsymbol{\Sigma}}$ obtained by selecting rows and columns corresponding to the entries of $\boldsymbol{B}_{k,\cdot}$. W_k is asymptotically distributed as a chi-square distribution with d_e degrees of freedom under the null distribution, yielding the p-values reported by `MGLMreg`. Users can also easily invoke the LRT by calling `MGLMreg` with option `LRT=TRUE`. LRT requires more computation than Wald test as it needs to perform MLE on p sub-models.

Regularization

The number of parameters, pd_e, in the multivariate response models can be unwieldy with a moderate to large number of predictors. When the sample size n is limited or even smaller than pd_e, regularization is necessary for variable selection, model interpretation, and improving the risk property of the estimate. In general, the `MGLMsparsereg` function solves the regularized problem

$$\min_{\boldsymbol{B}} \; -\ell(\boldsymbol{B}) + J(\boldsymbol{B}),$$

where ℓ is the log-likelihood function and J is a penalty term. The choice of J depends on specific applications. We implemented three classes of penalties. Below, $\mathcal{S}$ is an index set for the set of predictors subject to regularization, which can be selectively specified by the users.

In *elementwise regularization* (`penalty='sweep'`),

$$J(\boldsymbol{B}) = \sum_{k \in \mathcal{S}} \sum_{j=1}^{d_e} P_\eta(|\beta_{kj}|, \lambda),$$

where P is a scalar penalty function, λ is the penalty tuning constant, and η is a parameter indexing member of a penalty family. Choices of the penalty families include: power family (Frank and Friedman, 1993), where $P_\eta(|x|, \lambda) = \lambda |x|^\eta$, $\eta \in (0, 2]$, and in particular lasso (Tibshirani, 1996) ($\eta = 1$) and ridge ($\eta = 2$); elastic net (Zou and Hastie, 2005), where $P_\eta(|x|, \lambda) = \lambda[(\eta - 1)x^2/2 + (2 - \eta)|x|]$, $\eta \in [1, 2]$; SCAD (Fan and Li, 2001), where $\partial / \partial |x| P_\eta(|x|, \lambda) = \lambda \left\{ 1_{\{|x| \leq \lambda\}} + (\eta\lambda - |x|)_+ / (\eta - 1)\lambda 1_{\{|x| > \lambda\}} \right\}$, $\eta > 2$; and MC+ penalty (Zhang, 2010), where $P_\eta(|x|, \lambda) = \left\{ \lambda |x| - x^2/(2\eta) \right\} 1_{\{|x| < \eta\lambda\}} + 0.5\lambda^2 \eta 1_{\{|x| \geq \eta\lambda\}}$. The special case of elastic net for the multinomial-logit model is also implemented in the popular **glmnet** package by Friedman et al. (2010) using coordinate descent algorithm. **MGLM** implements a generic accelerated proximal gradient method that applies to the following two penalties too.

In *groupwise regularization* (`penalty='group'`),

$$J(\boldsymbol{B}) = \lambda \sum_{k \in \mathcal{S}} \|\boldsymbol{B}_{k,\cdot}\|_2 = \lambda \sum_{k \in \mathcal{S}} \left(\sum_{j=1}^{d_e} \beta_{kj}^2 \right)^{1/2}.$$

The group penalty (Yuan and Lin, 2006; Meier et al., 2008) achieves variable selection at the predictor level and leads to a more interpretable model.

Shrinkage and sparsity in terms of the rank of $\boldsymbol{B}$ is achieved by the *nuclear norm* regularization (`penalty='nuclear'`)

$$J(\boldsymbol{B}) = ||\boldsymbol{B}||_* = \lambda \sum_i \sigma_i(\boldsymbol{B}),$$

where $\sigma_i(\boldsymbol{B})$'s are the singular values of the matrix $\boldsymbol{B}$. The nuclear norm $||\boldsymbol{B}||_*$ is a suitable convex relaxation of the rank of a matrix parameter. It encourages low rank of the parameter matrix estimate and has been successfully employed in the matrix completion problem (Mazumder et al., 2010), regression with matrix covariates (Zhou and Li, 2014) and predicting at-bat results in baseball (Powers et al.).

The wrapper function `MGLMtune` facilitates the tuning procedure by performing the regularized estimation over a grid of 30 (default) tuning parameter values using warm start and reports the optimal tuning parameter according to BIC. Users can also supply their own grid points.

Optimization algorithms and implementation

As the DM, NegMN, and GDM distributions do not belong to the exponential family, the usual iteratively reweighted least squares method for maximum likelihood estimation of GLM does *not* apply. The main issue lies in the difficulty of calculating the expected information matrix, which involves evaluating numerous tail probabilities of the marginal distribution (Paul et al., 2005; Zhou and Lange, 2010; Zhou and Zhang, 2012). On the other hand, Newton's method suffers from instability since the log-likelihood functions are non-concave in general.

For distribution fitting, Zhou and Lange (2010) derive stable algorithms based on the minorization-maximization (MM) principle (Lange et al., 2000). Similar to the classical expectation-maximization algorithm, MM algorithm increases the objective value at each iteration and converges to a stationarity point of objective function under mild regularity conditions.

For regression models, Zhang et al. (2017) propose an iteratively reweighted Poisson regression (IRPR) method for maximum likelihood estimation. Their derivation again hinges upon the MM principle, resulting in the much desirable stability of the IRPR algorithm which is critical as the number of parameters is potentially large.

In practice, MM algorithm may suffer from slow convergence especially in the proximity of the optimum. **MGLM** organically combines the MM algorithm and the Newton's method. At each iteration, it computes both MM and Newton updates and chooses the one that results in a higher log-likelihood. Thus, stability is ensured as the log-likelihood always increases. When sufficiently close to the optimum, Newton's method takes over and quickly converges to the MLE in just a few iterations.

An added advantage of the MM algorithm is that it separates parameters and embraces parallel computing (Zhou et al., 2010). Each iteration of the IRPR algorithm involves solving d_e independent Poisson regressions (Zhang et al., 2017) that can be carried out in parallel. Building upon the **parallel** package in R, the `MGLMreg` regression function supports multi-threading on a multi-core machine.

For sparse regression in the `MGLMsparsereg` function, we implemented the accelerated proximal gradient (Nesterov) method (Zhang et al., 2017). Each iteration only involves evaluation of the gradient of the negative log-likelihood, followed by the elementwise, groupwise, or singular value thresholding according to the regularization being used, and thus scales well with the problem size.

The R package aspect

Three main functions of the **MGLM** package are `MGLMfit` for fitting multivariate distributions, `MGLMreg` for fitting multivariate response regressions, and `MGLMsparsereg` for fitting sparse regressions. The package adopts `S4` object system. In this section, we demonstrate their basic usage using a simulated RNA-seq data set. The R vignette included in the package provides more extensive examples.

The `rnaseq` data that comes with the package is simulated from the **isoform** package (Sun, 2014; Sun et al., 2015) and mimics the real counts generated by the RNA-seq platforms. The simulation mechanism follows the biological process and has nothing to do with the models in Table 1.

In this example, 6 exons are expressed in a gene. Each observation consists of the expression levels (represented by counts) of each exon along with covariates `totalReads`, `treatment`, `gender`, and `age` of an individual. 200 observations are simulated. In the generative model, exon expression levels are affected by intercept, number of total reads (on log scale), and treatment. Age and gender are unrelated to the exon counts.

```
R> library("MGLM")
R> data("rnaseq")
R> data <- rnaseq[, 1:6]
R> head(rnaseq, n = 3)

     X1  X2 X3  X4 X5 X6 totalReads treatment gender age
  1 295  65 19 114 54 20   28317494         0      0  57
  2 213 126 12  96 50  4   20015549         0      0  67
  3 322 147 23 245 42 19   35318251         0      1  37

R> dim(rnaseq)

  [1] 200  10
```

Distribution fitting

We first ignore covariates and demonstrate distribution fitting and model selection with BIC and LRT. The multi-categorical distribution fitting is performed by the function `MGLMfit`.

The primary inputs of the function are the multi-categorical count data in the format of data frame or matrix and the distribution intended to fit. The user can also use the `weight` argument to specify the weight of each observation. Initial values of the iterative algorithm can also be determined by the user using `init` argument. The function also has `epsilon`, `maxiters`, and `display` to control the convergence threshold, maximum number of iterations to run, and whether to display the result from each iteration, respectively.

The outputs are returned in a list of class "MGLMfit", including parameter estimates, their standard errors, log-likelihood, AIC, BIC, number of iterations to converge. The inversed information matrices and gradients are also returned, but are not printed, in order to keep the printed output concise. When fitting DM and GDM distribution, we also perform LRT, testing against the null hypothesis of using multinomial model. p-values of the LRTs are returned.

The following snippets fit the DM

```
R> system.time (
+    dmFit <- MGLMfit(data, dist = "DM")
+  )

     user  system elapsed
    0.255   0.007   0.263

R> dmFit

           estimate       SE
  alpha_X1 6.128117 0.327888
  alpha_X2 2.413647 0.139676
  alpha_X3 1.625627 0.099424
  alpha_X4 6.822929 0.362568
  alpha_X5 2.214236 0.129233
  alpha_X6 0.784028 0.051369

  Distribution: Dirichlet Multinomial
  Log-likelihood: -4968.666
  BIC: 9969.121
  AIC: 9949.331
  LRT test p value: <0.0001
  Iterations: 6
```

and the GDM model

```
R> system.time(
+    gdmFit <- MGLMfit(data, dist = "GDM")
+  )

     user  system elapsed
    0.380   0.019   0.399

R> gdmFit
```

```
           estimate       SE
  alpha_X1  3.741846 0.367088
  alpha_X2  2.400909 0.815431
  alpha_X3  1.558396 0.233136
  alpha_X4  6.988354 1.164764
  alpha_X5 20.689398 0.149279
  beta_X1   8.026379 0.966502
  beta_X2  11.038376 0.725978
  beta_X3   8.961428 0.264520
  beta_X4   2.702723 2.871718
  beta_X5   4.854816 0.648271

  Distribution: Generalized Dirichlet Multinomial
  Log-likelihood: -4841.231
  BIC: 9735.446
  AIC: 9702.463
  LRT test p value: <0.0001
  Iterations: 59
```

Both `dmFit` and `gdmFit` give the LRT p-value for comparing the fitted model with the MN model. In this example, both are significantly better than MN, with p-values $\ll 0.05$. To compare DM and GDM, we can either compare their BICs or perform a formal LRT.

```
R> LRT <- -2 * (logLik(dmFit) - logLik(gdmFit))
R> pchisq(LRT, ncol(data) - 2, lower.tail = FALSE)

  [1] 5.817352e-54
```

Both suggest that GDM provides a significantly better fit than DM.

The NegMN model

```
R> system.time(
+    negmnFit <- MGLMfit(data, dist = "NegMN")
+  )

     user  system elapsed
    0.006   0.002   0.009

R> print(negmnFit)

        estimate       SE
  p_X1  0.311486 0.001362
  p_X2  0.106491 0.000850
  p_X3  0.098373 0.000819
  p_X4  0.350496 0.001425
  p_X5  0.094263 0.000803
  p_X6  0.021220 0.000389
  phi  12.232569 1.229253

  Distribution: Negative Multinomial
  Log-likelihood: -20673.71
  BIC: 41384.52
  AIC: 41361.43
  LRT test p value:
  Iterations: 3
```

yields a much larger BIC than those of DM and GDM. LRT does not apply here, since NegMN is not a sub-model of the other three.

Regression

The more interesting question is whether the covariates have a significant relationship to the responses. Regressions are performed using function `MGLMreg`. First, the regression formula is required by the `MGLMreg` function. When specifying the regression formula, the response matrix is on the left hand side and the covariates on the right, following the convention in `lm` and `glm`. The model is specified

using the `dist` argument. The input argument `data` is optional. When specified, the formula is built based on the supplied data frame; otherwise, the function searches through the global environment for the data elements. Similar to the distribution fitting function, weights of the observations can be specified by the `weight` argument, and initial values can be supplied using `init` arguments. Parallel computing is also implemented in the package. Setting `parallel=TRUE` and giving the number of cores using the argument `core` invokes multi-threading. The other arguments used to control the algorithm convergence and display include `epsilon`, `maxiters`, `display`, and `LRT`.

The output of `MGLMreg` is a list of class "MGLMreg", containing the estimated regression coefficients, their standard errors, Wald test statistics and their corresponding p-values for each predictor, log-likelihood, AIC, BIC, gradient and Hessian matrix at the estimate, number of iterations, and fitted values.

We fit the four regression models in Table 1 with all 5 covariates: intercept, number of total reads (on log scale), treatment, age, and gender.

A few observations can be made from the following output. BIC indicates the GDM model as the best fit, followed by the DM model. This is consistent with the distribution fitting results. Note that the hypothesis testing results in the four models are different. In the MN model, all covariates are significant; however, this is not true because age and gender have no effects in the generative model. DM also wrongly identifies age as a significant predictor. Only the GDM model correctly identifies true significant predictors.

MN regression

```
R> system.time(
+    mnreg <- MGLMreg(formula = cbind(X1, X2, X3, X4, X5, X6) ~ log(totalReads) +
+                        treatment + age + gender, data = rnaseq, dist = "MN")
+  )

     user  system elapsed
    0.142   0.006   0.149

R> print(mnreg)

  Call: MGLMreg(formula = cbind(X1, X2, X3, X4, X5, X6) ~ log(totalReads) +
      treatment + age + gender, data = rnaseq, dist = "MN")

  Coefficients:
              X1        X2        X3        X4        X5
  [1,]  4.942732  5.009649 -3.792216  4.435434  4.027689
  [2,] -0.112845 -0.170222  0.219277 -0.107260 -0.120928
  [3,] -0.022655 -0.043099  2.745277  1.405742  0.092246
  [4,] -0.006187 -0.009709 -0.005907 -0.010945 -0.009599
  [5,]  0.032676  0.100389  0.020663  0.103859  0.009514

  Hypothesis test:
                   wald value    Pr(>wald)
  (Intercept)       144.88789 1.634268e-29
  log(totalReads)    69.92572 1.061922e-13
  treatment       18364.13260 0.000000e+00
  age                79.91023 8.762650e-16
  gender             52.33670 4.601575e-10

  Distribution: Multinomial
  Log-likelihood: -7506.393
  BIC: 15145.24
  AIC: 15062.79
  Iterations: 6
```

DM regression

```
R> system.time(
+    dmreg <- MGLMreg(formula = cbind(X1, X2, X3, X4, X5, X6) ~ log(totalReads) +
+                        treatment + age + gender, data = rnaseq, dist = "DM")
+  )
```

```
     user  system elapsed
    0.182   0.004   0.187

R> print(dmreg)

  Call: MGLMreg(formula = cbind(X1, X2, X3, X4, X5, X6) ~ log(totalReads) +
      treatment + age + gender, data = rnaseq, dist = "DM")

  Coefficients:
             X1        X2        X3        X4        X5        X6
  [1,] -0.895850 -1.096921 -8.997414 -1.736871 -1.774227 -5.646822
  [2,]  0.221988  0.186919  0.536572  0.252679  0.216672  0.347271
  [3,] -0.679291 -0.686881  1.835585  0.707954 -0.546469 -0.543134
  [4,]  0.010245  0.005227  0.009134  0.004252  0.006090  0.011642
  [5,] -0.026177  0.040244 -0.052842  0.023178 -0.058339 -0.039139

  Hypothesis test:
                   wald value Pr(>wald)
  (Intercept)       14.579069  0.023796
  log(totalReads)    8.502549  0.203547
  treatment       1851.437449  0.000000
  age               13.131512  0.040994
  gender             4.133364  0.658634

  Distribution: Dirichlet Multinomial
  Log-likelihood: -4386.941
  BIC: 8932.831
  AIC: 8833.882
  Iterations: 9
```

GDM regression

```
R> system.time(
+    gdmreg <- MGLMreg(formula = cbind(X1, X2, X3, X4, X5, X6) ~ log(totalReads) +
+                         treatment + age + gender, data = rnaseq, dist = "GDM")
+  )

     user  system elapsed
    0.219   0.003   0.224

R> print(gdmreg)

  Call: MGLMreg(formula = cbind(X1, X2, X3, X4, X5, X6) ~ log(totalReads) +
      treatment + age + gender, data = rnaseq, dist = "GDM")

  Coefficients:
        alpha_X4  alpha_X1  alpha_X2   alpha_X3  alpha_X5   beta_X4
  [1,]  5.987993 -7.056673  0.456088 -10.120738  2.639396  4.661089
  [2,] -0.215099  0.555973  0.039553   0.720358 -0.016121 -0.140896
  [3,] -0.047691 -0.329320  0.979359   0.099958  0.063393  0.628878
  [4,]  0.006661 -0.004343  0.019361   0.008173  0.012397  0.003224
  [5,]  0.233006  0.374838 -0.186420  -0.202417  0.144289  0.212071
         beta_X1   beta_X2   beta_X3   beta_X5
  [1,] -9.789127  7.095061 -9.530008 -1.687615
  [2,]  0.713819 -0.222984  0.743146  0.133985
  [3,]  0.746198 -1.591630 -0.923712 -0.042441
  [4,]  0.000453  0.015945  0.012541  0.019188
  [5,]  0.273256 -0.233121 -0.270428  0.122062

  Hypothesis test:
                  wald value Pr(>wald)
  (Intercept)       15.40109  0.118109
  log(totalReads)   11.04187  0.354265
  treatment       2549.23829  0.000000
  age               16.42846  0.088007
```

```
  gender            10.72122  0.379646

  Distribution: Generalized Dirichlet Multinomial
  Log-likelihood: -4289.281
  BIC: 8843.479
  AIC: 8678.563
  Iterations: 46
```

NegMN regression

There are two variants of NegMN regression, depending on whether to link over-dispersion parameter to covariates. The default setting `regBeta=FALSE` instructs `MGLMreg` to fit the NegMN regression with all observations sharing the same over-dispersion parameter value. There are $pd + 1$ parameters.

```
R> system.time(
+    negmnreg2 <- MGLMreg(formula = cbind(X1, X2, X3, X4, X5, X6) ~
+                           log(totalReads) + treatment + age + gender,
+                         data = rnaseq, dist = "NegMN", regBeta = FALSE)
+  )

     user  system elapsed
    0.196   0.004   0.201

R> print(negmnreg2)

  Call: MGLMreg(formula = cbind(X1, X2, X3, X4, X5, X6) ~ log(totalReads) +
      treatment + age + gender, data = rnaseq, dist = "NegMN",
      regBeta = FALSE)

  Coefficients:
  $alpha
                          X1         X2         X3         X4
  (Intercept)     -13.587385 -13.521818 -22.380101 -14.131348
  log(totalReads)   0.907716   0.850412   1.242922   0.915258
  treatment        -0.753113  -0.773507   2.014641   0.675195
  age               0.002583  -0.000938   0.002916  -0.002141
  gender           -0.060696   0.007022  -0.069502   0.012499
                          X5         X6
  (Intercept)     -14.507698 -18.526425
  log(totalReads)   0.899918   1.020360
  treatment        -0.638296  -0.730410
  age              -0.000824   0.008766
  gender           -0.083681  -0.093397

  $phi
      phi
  31.6062


  Hypothesis test:
                   wald value    Pr(>wald)
  (Intercept)       385.75540 3.223413e-80
  log(totalReads)   368.08485 2.017976e-76
  treatment       18377.52958 0.000000e+00
  age                79.70906 4.103065e-15
  gender             54.84662 4.978098e-10

  Distribution: Negative Multinomial
  Log-likelihood: -8746.689
  BIC: 17657.63
  AIC: 17555.38
  Iterations: 35
```

`regBeta=TRUE` instructs `MGLMreg` to link over-dispersion parameter to covariates and there are $p(d + 1)$ regression coefficients. Small likelihood, larger AIC/BIC and slow convergence reflects the lack of fit of this model.

```
R> system.time(
+      negmnreg <- MGLMreg(formula = cbind(X1, X2, X3, X4, X5, X6) ~
+                            log(totalReads) + treatment + age + gender,
+                          data = rnaseq, dist = "NegMN", regBeta = TRUE)
+  )

     user  system elapsed
    9.866   0.023   9.898

R> print(negmnreg)

  Call: MGLMreg(formula = cbind(X1, X2, X3, X4, X5, X6) ~ log(totalReads) +
      treatment + age + gender, data = rnaseq, dist = "NegMN",
      regBeta = TRUE)

  Coefficients:
                          X1         X2         X3         X4
  (Intercept)     -17.648355 -17.582555 -26.462109 -18.204282
  log(totalReads)   1.192057   1.134742   1.528500   1.200309
  treatment        -0.877324  -0.897715   1.890375   0.550944
  age              -0.013397  -0.016918  -0.013072  -0.018127
  gender           -0.101456  -0.033730  -0.110253  -0.028252
                          X5         X6        phi
  (Intercept)     -18.569609 -22.587821  7.543157
  log(totalReads)   1.184315   1.304725 -0.285737
  treatment        -0.762503  -0.854624  0.125412
  age              -0.016804  -0.007213  0.015871
  gender           -0.124433  -0.134173  0.041254

  Hypothesis test:
                   wald value    Pr(>wald)
  (Intercept)       291.48017 3.987668e-59
  log(totalReads)   371.39226 3.232075e-76
  treatment       18377.18774 0.000000e+00
  age                81.70350 6.187095e-15
  gender             54.79955 1.633654e-09

  Distribution: Negative Multinomial
  Log-likelihood: -8745.162
  BIC: 17675.77
  AIC: 17560.32
  Iterations: 140
```

Sparse regression

The function `MGLMsparsereg` performs regularized estimation. Similar to `MGLMreg`, the inputs of the sparse regression function include `formula`, `data`, `dist`, and the convergence controlling arguments. The function also requires the tuning parameter `lambda`, and the penalty type argument `penalty`. The outputs include the coefficient estimates, log-likelihood, AIC, BIC, degrees of freedom, and the number of iterations.

We simulate 100 data points from a 5-variate DM model using 10 covariates. Only three of them (1, 3, and 5) have non-zero effects. For each 5-variate data point, batch size, or the total number of objects that are put into 5 categories, is drawn from $Bin(200, 0.8)$.

```
R> dist <- "DM"
R> n <- 100
R> p <- 10
R> d <- 5
R> set.seed(118)
R> m <- rbinom(n, 200, 0.8)
R> X <- matrix(rnorm(n * p), n, p)
R> alpha <- matrix(0, p, d)
R> alpha[c(1, 3, 5), ] <- 1
R> Alpha <- exp(X %*% alpha)
```

```
R> Y <- rdirmn(size = m, alpha = Alpha)
R> length(m)

  [1] 100

R> head(m)

  [1] 167 160 162 159 156 157

R> head(Y)

      [,1] [,2] [,3] [,4] [,5]
  [1,]   24    7  112   15    9
  [2,]   34   33   31   38   24
  [3,]    0    0    0    0  162
  [4,]    7   17   84   29   22
  [5,]    0   33    0  123    0
  [6,]    0    0    3  154    0

R> head(rowSums(Y))

  [1] 167 160 162 159 156 157
```

We demonstrate the regularized estimation by group, nuclear norm, and element-wise penalization.

Variable selection by group penalty

With input `lambda=Inf`, `MGLMsparsereg` returns $\lambda_{\max}$, the maximum value the tuning parameter can take such that not all regression coefficient estimates are 0.

```
R> pen <- "group"
R> ngridpt <- 30
R> spmodelfit <- MGLMsparsereg(formula = Y ~ 0 + X, dist = dist,
+                        lambda = Inf, penalty = pen)
R> maxlambda <- maxlambda(spmodelfit)
R> lambdas <- exp(seq(from = log(maxlambda), to = log(maxlambda / nrow(Y)),
+                     length.out = ngridpt))
```

Tuning is performed on 30 grid points. The left panel of Figure 1 displays the BIC trace along the solution path.

```
R> BICs <- rep(0, ngridpt)
R> AICs <- rep(0, ngridpt)
R> LogLs <- rep(0, ngridpt)
R> Dofs <- rep(0, ngridpt)
R> ptm <- proc.time()
R> for (j in 1:ngridpt) {
+    if (j == 1) {
+      B0 <- matrix(0, p, ncol(coef(spmodelfit)))
+    }
+    else B0 <- B_hat
+    select.fit <- MGLMsparsereg(formula = Y ~ 0 + X, dist = dist,
+                                lambda = lambdas[j], penalty = pen, init = B0)
+    B_hat <- coef(select.fit)
+    BICs[j] <- BIC(select.fit)
+    LogLs[j] <- logLik(select.fit)
+    AICs[j] <- AIC(select.fit)
+    Dofs[j] <- dof(select.fit)
+  }
R> proc.time() - ptm

     user  system elapsed
    4.469   0.026   4.500

R> pen <- "group"
R> ngridpt <- 30
```

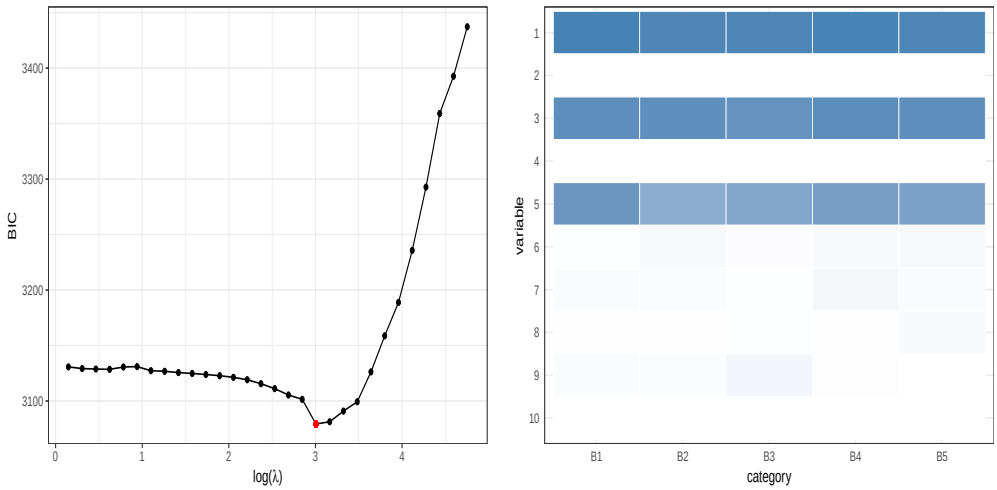


Figure 1: Variable selection by the group penalty. Left: BIC trace. Right: Regularized estimate $\widehat{\boldsymbol{B}}(\lambda)$ at the optimal λ displayed in gray scale.

```
R> spmodelfit <- MGLMsparsereg(formula = Y ~ 0 + X, dist = dist,
+                       lambda = Inf, penalty = pen)
R> maxlambda <- maxlambda(spmodelfit)
R> lambdas <- exp(seq(from = log(maxlambda), to = log(maxlambda / nrow(Y)),
+                     length.out = ngridpt))
```

The right panel of Figure 1 displays the regularized estimate $\widehat{\boldsymbol{B}}(\lambda)$ at the tuning parameter value with minimal BIC.

```
R> chosen.lambda <- lambdas[which.min(BICs)]
R> select <- MGLMsparsereg(formula = Y ~ 0 + X, dist = dist,
+                          lambda = chosen.lambda, penalty = pen)
```

Alternatively, the function `MGLMtune` automates the tuning procedure and reports the regularized estimate at the optimal tuning parameter value according to BIC.

```
R> selectTune <- MGLMtune(Y ~ 0 + X, dist = dist, penalty = pen, ngridpt = 30,
+                         display = FALSE)
```

The option `ngridpt` sets the number of grid points n_{grid} and the sequence of tuning parameters is equally spaced on the log scale within the interval $[\lambda_{\max}/n_{\text{grid}}, \lambda_{\max}]$.

Low rank regression by nuclear norm regularization

Nuclear norm regularization is invoked by setting `penalty="nuclear"`.

```
R> system.time (
+    select <- MGLMtune(Y ~ 0 + X, dist = "DM", penalty = "nuclear",
+                       ngridpt = 30, display = FALSE))

    user  system elapsed
   4.475   0.037   4.528
```

Figure 2 displays the BIC trace plot and the regularized estimate at optimal λ from the same data by the nuclear norm penalty.

Select by entries

```
R> system.time (
+    select <- MGLMtune(Y ~ 0 + X, dist = "DM", penalty = "sweep", ngridpt = 30,
+                       display = FALSE))
```

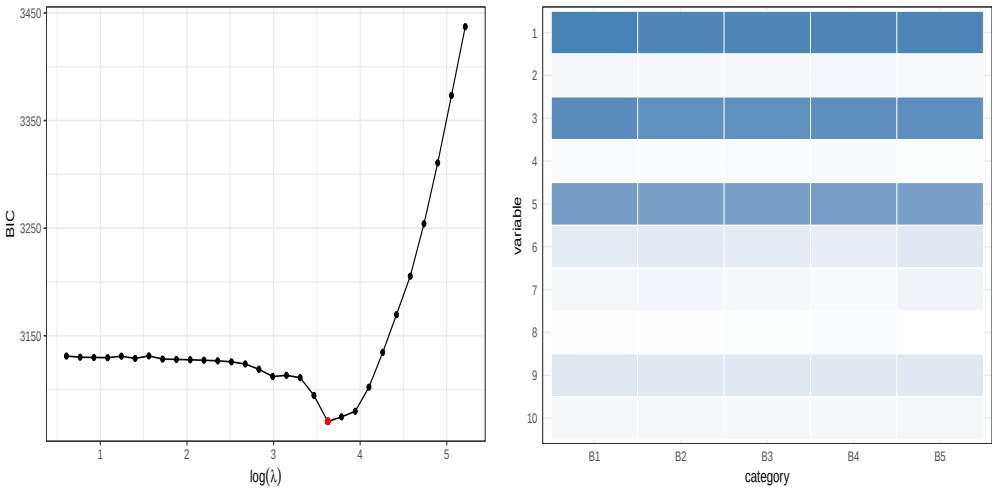


Figure 2: Low rank regression by the nuclear norm penalty. Left: BIC trace. Right: Regularized estimate $\widehat{B}(\lambda)$ at the optimal λ displayed in gray scale.

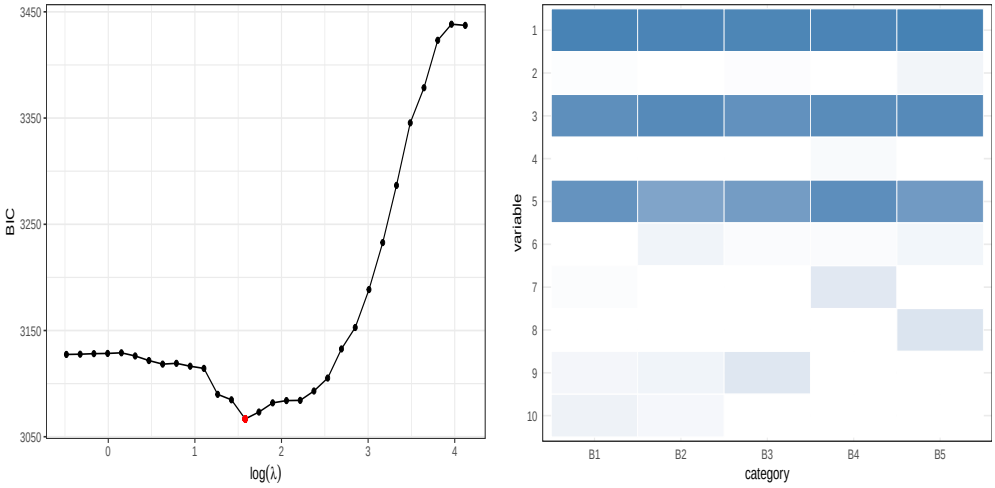


Figure 3: Select by parameter matrix entries with L1 norm penalty. Left: BIC trace. Right: Regularized estimate $\widehat{B}(\lambda)$ at the optimal λ displayed in gray scale.

```
   user  system elapsed
  4.448   0.038   4.504
```

Figure 3 displays the BIC trace plot and the regularized estimate at optimal λ from the same data by the element-wise lasso penalty.

Discussion

This article introduces the **MGLM** package for analysis of multivariate categorical data. Distribution fitting, regression, sparse regression, and random number generation are implemented in a simple and unified framework. It timely responds to the current challenge of multivariate categorical data analysis arising from modern technology such as RNA-seq. The R package is available on CRAN.

There are several possible extensions that would be useful to the package. Some of them include:

1. Alternative parameterization. Some models in Table 1 admits alternative parameterization. For

instance, the DM distribution can be re-parameterized as

$$\theta = \frac{1}{\sum_j \alpha_j},\ p_j = \frac{\alpha_j}{\sum_{j'} \alpha_{j'}},\ j = 1, \ldots, d.$$

in terms of the proportion vector $\boldsymbol{p} = (p_1, \ldots, p_d) \in \Delta_d$ and the overdispersion parameter $\theta > 0$. Appropriate inverse link function can be

$$p_j = \frac{e^{\mathbf{x}^T \boldsymbol{\beta}_j}}{1 + \sum_{j'=1}^{d-1} e^{\mathbf{x}^T \boldsymbol{\beta}_{j'}}}, \quad j = 1, \ldots, d-1,$$
$$\theta = e^{\mathbf{x}^T \boldsymbol{\beta}_d}.$$

The multinomial-logit model can be treated as a special case with $\boldsymbol{\beta}_d = \mathbf{0}$. Zhou and Lange (2010) discuss an algorithm for distribution fitting using this parameterization. Corresponding estimation algorithm for the regression model needs to be devised and implemented. Similar re-parameterization applies to the GDM model.

2. Alternative link function. Current version only implements the `log` link function for positive distribution parameter and `logit` link for the probability vector. Inclusion of alternative link functions such as `probit` and `cloglog` would expand the flexibility of the models.
3. Ordinal categorical responses. Multinomial-logit model can be adapted to ordinal categorical responses (Agresti, 2002, Chapter 7). Parallel developments and implementation for the DM, NegMN, and GDM models are worth considering.
4. Parameter constraints. Current version does not allow constraints among regression parameters. `MGLMreg` calls `glm.fit` functions to fit weighted Poisson regressions in each iteration; we may call functions from **glmc** package (Chaudhuri et al., 2006) instead to incorporate parameter constraints.
5. The `xij` argument (Yee, 2015, Chapter 14). Current version assumes the same set of covariates for all categories. Allowing covariates to take different values for each category can be useful, e.g., for discrete choice modeling in econometrics. The corresponding algorithm and implementation are worth exploring.

The **MGLM** package for R is continually being developed.

Acknowledgments

The work is partially supported by NSF grants DMS-1127914, DMS-1310319 and NIH grants HG006139, GM105785 and GM53275.

Bibliography

A. Agresti. *Categorical Data Analysis*. Wiley Series in Probability and Statistics. John Wiley & Sons, New York, 2nd edition, 2002. ISBN 0-471-36093-7. [p73, 88]

S. G. Baker. The multinomial-poisson transformation. *Journal of the Royal Statistical Society. Series D (The Statistician)*, 43(4):pp. 495–504, 1994. ISSN 00390526. URL http://www.jstor.org/stable/2348134. [p74]

N. Bouguila. Clustering of count data using generalized dirichlet multinomial distributions. *IEEE Transactions on Knowledge and Data Engineering*, 20(4):462–474, 2008. ISSN 1041-4347. URL http://doi.org/10.1109/TKDE.2007.190726. [p76]

S. Boyd and L. Vandenberghe. *Convex Optimization*. Cambridge University Press, Cambridge, 2004. ISBN 0-521-83378-7. [p74]

S. Chaudhuri, M. S. Handcock, and M. S. Rendall. *Glmc: An R Package for Generalized Linear Models Subject to Constraints*, 2006. URL http://www.stat.washington.edu/handcock/combining. [p88]

R. J. Connor and J. E. Mosimann. Concepts of independence for proportions with a generalization of the Dirichlet distribution. *J. Amer. Statist. Assoc.*, 64:194–206, 1969. ISSN 0162-1459. [p76]

B. Efron and D. V. Hinkley. Assessing the accuracy of the maximum likelihood estimator: Observed versus expected Fisher information. *Biometrika*, 65(3):457–487, 1978. ISSN 0006-3444. URL http://doi.org/10.1093/biomet/65.3.457. [p77]

J. Fan and R. Li. Variable selection via nonconcave penalized likelihood and its oracle properties. *J. Amer. Statist. Assoc.*, 96(456):1348–1360, 2001. ISSN 0162-1459. URL http://doi.org/10.1198/016214501753382273. [p77]

I. E. Frank and J. H. Friedman. A statistical view of some chemometrics regression tools. *Technometrics*, 35(2):109–135, 1993. ISSN 00401706. URL http://doi.org/10.2307/1269656. [p77]

J. H. Friedman, T. Hastie, and R. Tibshirani. Regularization paths for generalized linear models via coordinate descent. *Journal of Statistical Software*, 33(1):1–22, 2010. ISSN 1548-7660. URL http://www.jstatsoft.org/v33/i01. [p73, 77]

R. L. Graham, D. E. Knuth, and O. Patashnik. *Concrete Mathematics*. Addison-Wesley Publishing Company, Reading, MA, 2nd edition, 1994. ISBN 0-201-55802-5. A foundation for computer science. [p75]

J. B. Lang. On the comparison of multinomial and Poisson log-linear models. *J. Roy. Statist. Soc. Ser. B*, 58 (1):253–266, 1996. ISSN 0035-9246. URL http://links.jstor.org/sici?sici=0035-9246(1996)58:1<253:OTCOMA>2.0.CO;2-P&origin=MSN. [p74]

K. Lange, D. R. Hunter, and I. Yang. Optimization transfer using surrogate objective functions. *J. Comput. Graph. Statist.*, 9(1):1–59, 2000. ISSN 1061-8600. With discussion, and a rejoinder by Hunter and Lange. [p78]

R. Mazumder, T. Hastie, and R. Tibshirani. Spectral regularization algorithms for learning large incomplete matrices. *Journal of Machine Learning Research*, 11:2287–2322, 2010. [p78]

L. Meier, S. van de Geer, and P. Bühlmann. The group Lasso for logistic regression. *J. R. Stat. Soc. Ser. B Stat. Methodol.*, 70(1):53–71, 2008. ISSN 1369-7412. [p77]

S. R. Paul, U. Balasooriya, and T. Banerjee. Fisher information matrix of the Dirichlet-multinomial distribution. *Biom. J.*, 47(2):230–236, 2005. ISSN 0323-3847. [p78]

S. Powers, T. Hastie, and R. Tibshirani. Nuclear penalized multinomial regression with an application to predicting at-bat outcomes in baseball. *To appear in special edition "Statistical Modelling for Sports Analytics", Statistical Modelling*. [p78]

W. Sun. *Isoform: A Set of Tools for RNA Isoform Study Using RNA-Seq Data*, 2014. URL http://www.bios.unc.edu/~weisun/software/isoform.htm. R package version 0.99.2. [p78]

W. Sun, Y. Liu, J. J. Crowley, and et al. IsoDOT detects differential RNA-isoform expression/usage with respect to a categorical or continuous covariate with high sensitivity and specificity. *J. Amer. Statist. Assoc.*, 110(511):975–986, 2015. ISSN 0162-1459. URL http://doi.org/10.1080/01621459.2015.1040880. [p78]

R. Tibshirani. Regression shrinkage and selection via the lasso. *J. Roy. Statist. Soc. Ser. B*, 58(1):267–288, 1996. ISSN 0035-9246. URL http://links.jstor.org/sici?sici=0035-9246(1996)58:1<267:RSASVT>2.0.CO;2-G&origin=MSN. [p77]

T. Tvedebrink. Overdispersion in allelic counts and θ-correction in forensic genetics. *Theoretical Population Biology*, 78(3):200–210, 2010. ISSN 0040-5809. URL http://doi.org/10.1016/j.tpb.2010.07.002. [p73]

W. N. Venables and B. D. Ripley. *Modern Applied Statistics with S*. Springer-Verlag, New York, 4th edition, 2002. ISBN 0-387-95457-0. URL http://www.stats.ox.ac.uk/pub/MASS4. [p73]

T. W. Yee. The VGAM package for categorical data analysis. *Journal of Statistical Software*, 32(10):1–34, 2010. ISSN 1548-7660. URL http://www.jstatsoft.org/v32/i10. [p73]

T. W. Yee. *Vector Generalized Linear and Additive Models: With an Implementation in R*. Springer-Verlag, New York, USA, 2015. [p73, 88]

T. W. Yee. *VGAM: Vector Generalized Linear and Additive Models*, 2017. URL https://CRAN.R-project.org/package=VGAM. R package version 1.0-4. [p73]

M. Yuan and Y. Lin. Model selection and estimation in regression with grouped variables. *J. R. Stat. Soc. Ser. B Stat. Methodol.*, 68(1):49–67, 2006. ISSN 1369-7412. URL http://doi.org/10.1111/j.1467-9868.2005.00532.x. [p77]

C.-H. Zhang. Nearly unbiased variable selection under minimax concave penalty. *Ann. Statist.*, 38(2): 894–942, 2010. ISSN 0090-5364. URL http://doi.org/10.1214/09-AOS729. [p77]

Y. Zhang, H. Zhou, J. Zhou, and W. Sun. Regression models for multivariate count data. *J. Comput. Graph. Statist.*, 26(1):1–13, 2017. ISSN 1061-8600. URL http://doi.org/10.1080/10618600.2016.1154063. [p78]

H. Zhou and K. L. Lange. MM algorithms for some discrete multivariate distributions. *Journal of Computational and Graphical Statistics*, 19:645–665, 2010. [p76, 78, 88]

H. Zhou and L. Li. Regularized matrix regressions. *Journal of Royal Statistical Society, Series B*, 76(2): 463–483, 2014. [p78]

H. Zhou and Y. Zhang. EM vs MM: a case study. *Computational Statistics & Data Analysis*, 56:3909–3920, 2012. [p78]

H. Zhou, K. L. Lange, and M. A. Suchard. Graphical processing units and high-dimensional optimization. *Statistical Science*, 25:311–324, 2010. [p78]

H. Zou and T. Hastie. Regularization and variable selection via the elastic net. *J. R. Stat. Soc. Ser. B Stat. Methodol.*, 67(2):301–320, 2005. ISSN 1369-7412. URL http://doi.org/10.1111/j.1467-9868.2005.00503.x. [p77]

Juhyun Kim
Department of Biostatistics
University of California, Los Angeles
juhkim111@ucla.edu

Yiwen Zhang
KPMG LLP
Irvine, CA
zhangyiwen1015@gmail.com

Joshua Day
Department of Statistics
North Carolina State University
jtday2@ncsu.edu

Hua Zhou
Department of Biostatistics
University of California, Los Angeles
huazhou@ucla.edu

ArCo: An R package to Estimate Artificial Counterfactuals

by Yuri R. Fonseca, Ricardo P. Masini, Marcelo C. Medeiros and Gabriel F. R. Vasconcelos

Abstract In this paper we introduce the **ArCo** package for R which consists of a set of functions to implement the the Artificial Counterfactual (ArCo) methodology to estimate causal effects of an intervention (treatment) on aggregated data and when a control group is not necessarily available. The ArCo method is a two-step procedure, where in the first stage a counterfactual is estimated from a large panel of time series from a pool of untreated peers. In the second-stage, the average treatment effect over the post-intervention sample is computed. Standard inferential procedures are available. The package is illustrated with both simulated and real datasets.

Introduction

In this paper an R package to conduct counterfactual analysis by the Artificial Counterfactual (ArCo) method is introduced. The ArCo methodology is a flexible and easy-to-implement method to estimate the causal effects of an intervention on a single treated unit and when a control group is not readily available. The procedure consists of two steps where, in the first stage, a counterfactual is estimated on the basis of a large-dimensional set of variables from a pool of untreated units ("donors pool"). In the second stage, the average intervention effect on a vector of variables from the treated unit is estimated. The method is robust to the presence of confounding effects, such as a global shock and can be seen as an extension of the Synthetic Control (SC) approach of Abadie and Gardeazabal (2003) and Abadie et al. (2011) and the panel factor (PF) model method put forward by Hsiao et al. (2012).

Causality and counterfactual analysis are topics of major importance in Social Sciences, Economics, Medicine, Psychology, etc. Routinely, causal statements with respect to a given intervention (or treatment) depend on the construction of counterfactuals, which are estimated from the outcomes of a similar group of individuals not affected by the intervention. Nevertheless, reaching final cause-and-effect conclusions is a very challenging task as a consequence of the difficulties in finding reliable sources of exogenous variation. In microeconometrics there have been major advances in the literature and the estimation of treatment effects is now part of the current toolbox of applied researchers. On the other hand, when there is not a natural control group and there is a single treated unit, which is usually the case when handling, for example, macro (aggregated) data, the econometric tools have evolved at a slower pace. However, in the recent past, some authors have proposed new techniques inspired partially by the developments in microeconometrics that are able, under some assumptions, to estimate counterfactuals with aggregate data. The ArCo method fits into this exciting literature. See, for example, Athey and Imbens (2016a) or Athey and Imbens (2016b) for recent reviews.

Overview

All the econometric analysis conducted with the ArCo method is on the time dimension and it consists of the following key ingredients: the definition of the **intervention** and the **intervention time**, the **units** of interest (treated and untreated units) and the **observed variables** for each unit. Units are, for example, municipalities, states, regions, countries, products, firms, stores, etc. The intervention occurs in one of these units starting in a given point in time and does not affect the other units. Interventions could be war out-breaks, terrorist attacks, policy changes, enforcement of new laws, natural disasters, new prices, among many others. Each unit is described by at least one variable. For example, if we are dealing with countries the variables may be GDP and inflation and for companies they could be profit and revenue. For products the variables could be prices and sold quantities. All variables are observed along a period of time (days, months, years, for example).

The ArCo method is a two-step procedure. In the first step, the data before the occurrence of the intervention is used to estimate a multivariate time-series regression model relating the variables in the treated unit (dependent variables) with only the variables belonging to the untreated peers (explanatory variables). In the second step, the counterfactual is constructed by extrapolating the estimated model with data **after the intervention**. The estimated effect is the time-series average of the difference between the actual data and the counterfactual. It is important to stress that the model is conditioned on data from the untreated peers only and the validity of the method is based on two key assumptions:

1. The peers are not affected by the intervention;

2. The data is trend-stationary.

The first assumption is common in the literature and has been also assumed by Abadie and Gardeazabal (2003) and Hsiao et al. (2012). On the other hand, the trend-stationarity assumption has been dangerously ignored in the literature. Carvalho et al. (2016a) investigate the consequences of applying the ArCo, SC or PF methods when the data are integrated processes of order 1. They find that without a cointegrating relationship (spurious case) the estimator of the effects of the intervention diverges, resulting in the rejection of the null hypothesis of no effect regardless of its existence with probability approaching 1. For the case when one cointegrating relationship exists, the estimator is $\sqrt{T}$-consistent for the intervention effect albeit with a non-standard distribution. Nevertheless, even in this case, the test of no intervention effect is extremely oversized if nonstationarity is ignored. When there is a drift in the data generating processes, the estimator for both cases (cointegrated and spurious) either diverges or is not well defined asymptotically. The authors thus strongly recommend the use of first-differenced data to avoid spurious results in case of integrated data. Furthermore, heterogeneous, possibly nonlinear, **deterministic time trends** are allowed among units. See also Ferman and Pinto (2016) for a similar discussion.

High-dimensionality is relevant when the number of parameters to be estimated is large compared to the actual sample size. This can happen either when the number of peers and/or the number of variables for each peer is large or when the sample size is too small. In Carvalho et al. (2016b) the authors consider a linear model estimated by the *Least Absolute Selection and Shrinkage Operator* (LASSO) proposed by Tibshirani (1996). However, in the R package we leave the choice of the conditioning model to the user. LASSO regressions (and extensions), regression trees and random forests, boosted trees, neural networks, splines, factor models, are some possible examples of models to be estimated in the first step. The results in Carvalho et al. (2016b) are derived under asymptotic limits on the time dimension (T). However, the authors allow the number of peers (n) and the number of observed variables for each peer to grow as a function of T.

A Glimpse on the Literature and Comparison with Other Methods

The ArCo approach nests the PF method proposed by Hsiao et al. (2012) and can been seen as a generalization of the SC approach in the same lines as discussed by Doudchenko and Imbens (2016) and Ferman and Pinto (2016). It is also better suited than Difference-in-Differences (DiD) estimators for comparative case studies when there is a single treated unit and no similar control group is available, even after the inclusion of many control variables. Furthermore, the ArCo approach relaxes the stringent parallel trend restriction of the DiD methods.

Recently, Gobillon and Magnac (2016) generalize DiD estimators by considering a correctly specified linear panel model with strictly exogenous regressors and interactive fixed effects represented as a number of common factors with heterogeneous loadings. Their theoretical results rely on double asymptotics when both T and n go to infinity. The authors allow the common confounding factors to have nonlinear deterministic trends. The ArCo method differs from Gobillon and Magnac (2016) in several directions. First, as mentioned before, the model is not assumed to be correctly specified and there is no need to estimate the common factors. Consistent estimation of factors needs that both the time-series and the cross-section dimensions diverge to infinity and can be severely biased in small samples. The ArCo methodology requires only the time-series dimension to diverge. Furthermore, the regressors do not need to be strictly exogenous which is an unrealistic assumption in most applications with aggregate data. Heterogeneous nonlinear trends are also allowed but there is no need to estimate them (either explicitly or via common factors). Finally, as in the DiD case, the ArCo does not either require the number of treated units to grow or to have a reliable control group (after conditioning on covariates).

Although, both the ArCo and the SC methods construct a counterfactual as a function of observed variables from a pool of peers, the two approaches have important differences. First, the SC method relies on a convex combination of peers to construct the counterfactual which, as pointed out by Ferman and Pinto (2016), biases the estimator. The ArCo solution is a general, possibly nonlinear, function. Even in the case of linearity, the method does not impose any restriction on the parameters. For example, the restriction that weights in the SC methods are all positive seems a bit too strong. The SC method also requires an unrealistic identification assumption about the (perfect-)fit of the model in the pre-intervention period. Furthermore, the weights in the SC method are usually estimated using time averages of the observed variables for each peer. Therefore, all the time-series dynamics is removed and the weights are determined in a pure cross-sectional setting. In addition, the SC method was designed to evaluate the effects of the intervention on a single variable. In order to evaluate the effects in a vector of variables, the method has to be applied several times. The ArCo methodology can be directly applied to a vector of variables of interest. In addition, there is no formal inferential procedure for hypothesis testing in the SC method, whereas in the ArCo methodology, a simple,

uniformly valid and standard test can be applied[1]. Finally, as discussed in Ferman et al. (2016), the SC method does not provide any guidance on how to select the variables which determine the optimal weights[2].

Framework

This section aims to describe the mathematical notation and the key definitions of the ArCo methodology in a way that is compatible with the **ArCo** (Fonseca et al., 2017) package. For further details on statistical properties and theoretical results, see Carvalho et al. (2016b) and Carvalho et al. (2016a). Everything concerning the technique used to estimate the first-step model is left in a very general way as the **ArCo** package was developed to accept many different classes of models. Note, however, that the theory and the inference was developed for LASSO linear regressions.

- **units:** Indexed by $i = 1, \ldots, n$. Units are, for example, municipalities, states, regions, countries, products, firms, stores, etc. The treatment occurs in one of these units and do not affect the others.
- **variables:** For each unit i, $i = 1, \ldots, n$, and for every time period t, $t = 1, \ldots, T$, we observe $q_i \geq 1$ variables. If the units are, for example, firms, the variables may be sales, income, profit, etc. We will refer to these variables as $\boldsymbol{z}_{it} = (z_{it}^1, \ldots, z_{it}^{q_i})$.
- **intervention:** The intervention took place only in the treated unit at time $T_0 = \lambda_0 T$ with $\lambda_0 \in (0, 1)$.

Assume, without loss of generality, that the treated unit is the first one ($i = 1$). Furthermore, let $\boldsymbol{z}_{1t}^{(0)}$ and $\boldsymbol{z}_{1t}^{(1)}$ be the outcomes of the first unit under treatment and without treatment, respectively. Normally, we do not observe both outcomes simultaneously. Instead, we observe:

$$\boldsymbol{z}_{1t} = D_t \boldsymbol{z}_{1t}^{(0)} + (1 - D_t)\boldsymbol{z}_{1t}^{(1)}, \tag{1}$$

where D_t assumes value 1 if the unit in under treatment at time t and 0 otherwise.

The goal is to test the hypothesis on the effects of the intervention being statistically significant for $t \geq T_0$. The interventions are considered in the form

$$\boldsymbol{y}_t^{(1)} = \begin{cases} \boldsymbol{y}_{1t}^{(0)}, & t = 1, \ldots, T_0 - 1 \\ \boldsymbol{y}_{1t}^{(0)} + \boldsymbol{\delta}_t, & t = T_0, \ldots, T, \end{cases} \tag{2}$$

where $\boldsymbol{y}_t^{(j)} = \boldsymbol{h}(\boldsymbol{z}_{1t}^j) \in \mathbb{R}^q$ for $j \in \{0, 1\}$, $\boldsymbol{h}(\cdot)$ is a measurable function of $\boldsymbol{z}_{1t}$ and $\{\boldsymbol{\delta}_t\}_{t=T_0}^T$ is a deterministic sequence. The function $\boldsymbol{h}(\cdot)$ is very general and allows interventions on the mean, variance, covariance, etc. The ArCo method is concerned with the following hypothesis:

$$\mathcal{H}_0 : \boldsymbol{\Delta}_T = \frac{1}{T - T_0 + 1} \sum_{t=T_0}^{T} \boldsymbol{\delta}_t = 0, \tag{3}$$

where $\boldsymbol{\Delta}_t$ is the average treatment effect over the treatment period.

We do not observe $\boldsymbol{y}_t^{(0)}$ for $t \geq T_0$. This quantity is the counterfactual, i.e., what would $\boldsymbol{y}_t$ have been in the absence of intervention. In order to proceed to the first step estimation of the ArCo, let $\boldsymbol{z}_{0t} = (\boldsymbol{z}_{2t}', \ldots, \boldsymbol{z}_{nt}')'$ and $\boldsymbol{Z}_{0t} = (\boldsymbol{z}_{0t}', \ldots, \boldsymbol{z}_{0t-p}')'$ be the collection of all untreated units up to an arbitrary lag $p \geq 0$. The dimension of $\boldsymbol{Z}_{0t}$ depends on the number of units, the number of lags and the number of variables per unit.

Consider the following model for $\boldsymbol{y}_t^{(0)}$

$$\boldsymbol{y}_t^{(0)} = \mathcal{M}_t + \boldsymbol{v}_t, \tag{4}$$

where $\mathbb{E}(\boldsymbol{v}_t) = \boldsymbol{0}$ and $\mathcal{M}_t = \mathcal{M}(\boldsymbol{Z}_{0t})$. Note that $\mathcal{M}$ is a measurable mapping in a sense that it does not need to be an explicit function. For example, one could assume a regression tree or a random forest structure for $\mathcal{M}_t$.

The first step of the ArCo method consists on estimating (4) using the first $T_0 - 1$ observations, given that for $t < T_0$ we have $\boldsymbol{y}_t = \boldsymbol{y}_t^{(0)}$. Set $T_1 = T_0 - 1$ and $T_2 = T - T_0 + 1$, then one can estimate

[1] The theoretical results in Carvalho et al. (2016b) were derived for the case of LASSO regression only but can be easily extended to other classes of models.

[2] Doudchenko and Imbens (2016) advocate the use of shrinkage methods to estimate the pre-intervention model but no theory is provided.

$\widehat{\mathcal{M}}_{t,T_1} = \widehat{\mathcal{M}}_{T_1}(\mathbf{Z}_{0t})$ and use it to construct the counterfactual

$$\widehat{\boldsymbol{y}}_t^{(0)} = \begin{cases} \boldsymbol{y}_t^{(0)}, & t = 1, \ldots, T_0 - 1 \\ \widehat{\mathcal{M}}_{t,T_1}, & t = T_0, \ldots, T. \end{cases} \tag{5}$$

Finally, the ArCo estimator is defined as

$$\widehat{\boldsymbol{\Delta}}_T = \frac{1}{T - T_0 + 1} \sum_{t=T_0}^{T} \widehat{\boldsymbol{\delta}}_t, \tag{6}$$

where $\widehat{\boldsymbol{\delta}}_t = \boldsymbol{y}_t - \widehat{\boldsymbol{y}}_t^{(0)}$ and $t = T_0, \ldots, T$.

As mentioned in the Introduction, the ArCo methodology is a two-step estimator where the first step consists on the estimation of $\mathcal{M}$ on the pre-intervention sample and in the second step we estimate $\widehat{\boldsymbol{\Delta}}$, which is the average intervention impact.

Hypothesis Testing

Carvalho et al. (2016b) shows the conditions and results for the asymptotic normality of $\widehat{\boldsymbol{\Delta}}_T$ and also, how to consistently estimate its covariance matrix, given by $\boldsymbol{\Omega}_T$. Therefore, one can obtain a confidence interval for a chosen significance level α given by

$$\mathcal{I}_{j,\alpha} = \left[\widehat{\Delta}_{j,T} \pm \frac{\widehat{w}_j}{\sqrt{T}} \Phi^{-1}(1 - \alpha/2) \right], \tag{7}$$

for each $j = 1, \ldots, q$, where $\widehat{w}_j = \sqrt{\widehat{\Omega}_{jj}}$ and $\Phi^{-1}(\cdot)$ is the quantile function of a standard normal distribution.

One can also establish a hypothesis test given that $W_t = T\widehat{\boldsymbol{\Delta}}_T' \widehat{\boldsymbol{\Omega}}_T^{-1} \widehat{\boldsymbol{\Delta}}$ has a chi-square distribution with q degrees of freedom.

The estimation of $\boldsymbol{\Omega}$ may be a challenge not because of the ArCo itself, but because of the robust estimator we may choose. The **ArCo** package allows the user to choose between the methods below:

- Covariance matrix assuming the errors are independent and identically distributed;
- prewhitening using VAR models as in Andrews and Monahan (1992). The lag of the VAR may be chosen by the user or though information criterion;
- Newey and West (1987) (NW) covariance matrix with quadratic spectral, truncated, Bartlett, Parzen or Turkey-Hanning kernels; and
- the combination of NW and prewhitening where the NW covariance matrix is calculated on the residuals of the VAR.

Unknown Intervention Time

In many cases it is reasonable to assume that the intervention time is unknown. For example, although some new policy has started at a known time, its effects may have been anticipated due to rational expectations. Regardless the source of the uncertainty, what we need is to estimate λ_0 by adjusting the ArCo estimator to be a function of λ, $\widehat{\boldsymbol{\Delta}}_T(\lambda)$.

Set $\Lambda = (\underline{\lambda}, \overline{\lambda})$ as a bound for λ_0 to avoid finite sample bias close to the boundaries and define $||\cdot||_p$ as the ℓ_p norm, then

$$\widehat{\lambda}_{0,p} = \underset{\lambda \in \Lambda}{\arg\max}\, J_{T,p}(\lambda), \quad J_{T,p} = ||\widehat{\boldsymbol{\Delta}}_T(\lambda)||_p. \tag{8}$$

Thus, $\widehat{\lambda}_{0,p}$ is the one that maximizes the ℓ_p norm of the ArCo estimator. If $\boldsymbol{\Delta} \neq 0$ then $\widehat{\lambda}_{0,p} = \lambda_0 + o_p(1)$. On the other hand, if $\boldsymbol{\Delta} = \mathbf{0}$ then $\widehat{\lambda}_{0,p}$ converges in probability to any $\lambda \in \Lambda$ with equal probability.

Asymptotic Theory and Sample Size

The inferential procedures for the ArCo method are asymptotic. However, simulation experiments show that the method works well in small samples (Carvalho et al., 2016b). It is intuitive to think that the only sample size that matters is the one before the intervention as it is the one used to estimate the

model. However, for a given sample size T, it is desirable to have the intervention as close as possible to $0.5T$ because the average treatment effect, $\boldsymbol{\Delta}$, also depends on the period after the intervention. In practice, if the number of variables and units is not very big, 40 to 50 observations should be enough as we show in the Basque example when we compare the ArCo with the Synthetic Control approach.

The first step of the ArCo is a time-series estimation. Therefore, all the data must have the same periodicity (year, quarters, months, etc). The Synthetic Control kills the time-series dimension of the control variables because it uses only their average (or other statistic such as median). This is an important difference between the methods: the ArCo uses all the time-series information but it requires more data than the Synthetic Control because all series must be complete (without missing values) and in the same periodicity.

Practical Example: the fitArCo Function

The **ArCo** package has two datasets that were created using the data generation process (DGP) from the second simulation example in Carvalho et al. (2016b) and another dataset from the same article's empirical example. In this section we walk through the **ArCo** package using these datasets. All examples presented here are entirely reproducible using the same codes and datasets.

The DGP is given by:

$$z_{it}^{(0)} = 0.5\boldsymbol{A_i} z_{it-1}^{(0)} + \varepsilon_{i,t},\ i = 1,\ldots,n,\ t = 1,\ldots,T, \tag{9}$$

where the error has a factor structure $\varepsilon_{i,t} = \boldsymbol{\Lambda}_i \boldsymbol{f}_t + \boldsymbol{\eta}_{it}$ with $\boldsymbol{f}_t = [1, (t/T), v_t]$, $z_{it} \in \mathbb{R}^q$, $\boldsymbol{A_i}(q \times q)$ is a diagonal matrix with elements between -1 and 1, $\{v_t\}$ is a sequence of independent and normally (NID) distributed random variables with zero mean and unit variance, $\{\boldsymbol{\eta}_{it}\}$ is a sequence of NID random vectors with zero mean and covariance matrix $(0.2)^2 I_{nq}$. We set $T = 100$.

Simple example for one variable

The two generated datasets are `data.q1` (one variable and 20 units) and `data.q2` (two variables and 6 units). We will start with `data.q1`, which can be loaded with the following code:

```
> library(ArCo)
> data(data.q1)
```

The `data.q1` is a matrix with 100 observations of 20 columns. In the ArCo context each column is a unit, therefore, we have one variable from each of the 20 units from the data. The intervention took place at the first unit at $T_0 = 51$ by adding a constant of 0.628, which is one standard deviation from the treated unit before the intervention. The series in question is plotted in Figure 1 with a vertical line at $t = T_0$.

```
> plot(data.q1[,1], type="l")
> abline(v=51, col=2, lty=2)
```

The main function of the **ArCo** package is the function `fitArCo`, which estimates the ArCo and provides the relevant statistics and outputs. The data must be supplied in a list of matrices, each matrix should have T observations of one single variable for all units. Therefore, if one has q variables, the list must have q matrices. Since the `data.q1` has only one variable, we will supply the function with a list with only one matrix.

The user can choose any model to estimate the first step of the ArCo as long as it is possible to write a compatible estimation function (argument `fn`) and forecasting function (argument `p.fn`). The estimation function must receive a matrix of explanatory variables $\boldsymbol{X}$ and the response variable $\boldsymbol{y}$ in a vector, and it should return the estimated model; the forecasting function receives the output from the estimation function and the argument `newdata`[3]. For the purposes of this example we will write a `fn` and a `p.fn` that uses simple Ordinary Least Squares (OLS). The OLS estimation is the default model of the `fitArCo` function in case the user does not supply any functions.

```
> fn=function(X,y){
+   return(lm(y~X))
+ }
> p.fn=function(model,newdata){
+   b=coef(model)
```

[3]The forecasting function is similar to most `predict` methods in R, which receives a model object and the new data used on the prediction.

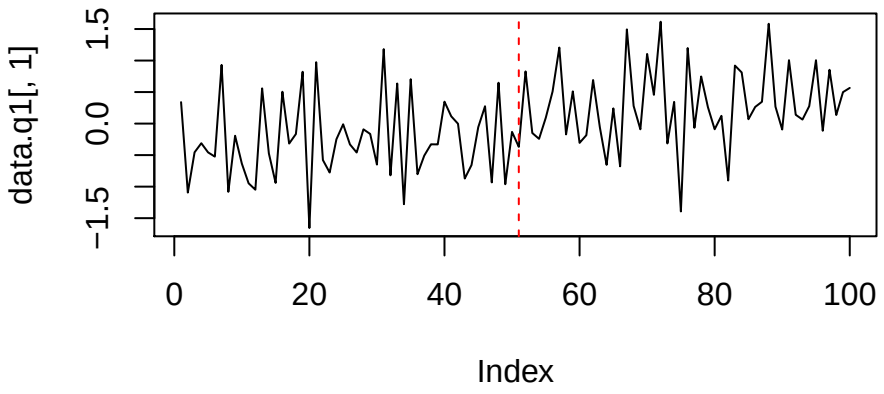


Figure 1: plot of data.q1 unit 1

```
+   return(cbind(1,newdata) %*% b)
+ }
```

The ArCo can now be estimated:

```
> ArCo=fitArCo(data = list(data.q1), fn = fn, p.fn = p.fn, treated.unit = 1, t0 = 51,
VCOV.type = "nw", VCOV.lag = 3, prewhitening.kernel = TRUE)
> plot(ArCo,display.fitted=TRUE)
> ArCo$delta
                 LB     delta        UB
Variable1 0.4115779 0.5174889 0.6233999
```

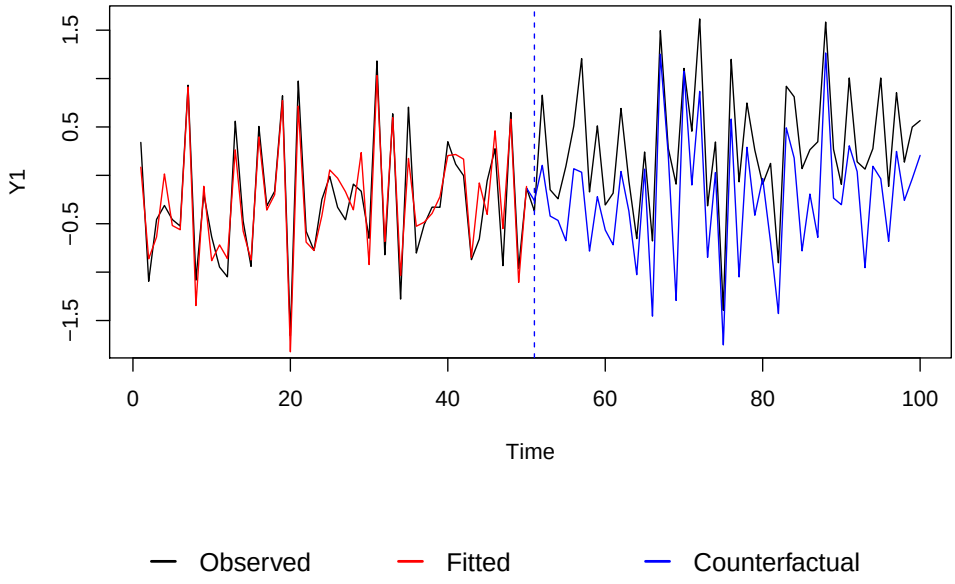


Figure 2: plot of the counterfactual from data.q1 unit 1

The `fitArCo` generates an object of class fitArCo with S3 method for plot[4] (Figure 2). Additionally, the user can chose from several types of robust covariance matrix estimations for $\boldsymbol{\Delta}_T$ such as Newey and West (1987) with prewhitening options from Andrews and Monahan (1992). The argument `lag` allows the inclusion of lags in the model and the argument `Xreg` accounts for exogenous controls. Finally, the `$delta` returns the $\boldsymbol{\Delta}_T$ and its confidence bands for a significance level α chosen by the user and `$p.value` returns the ArCo p-value.

Example for Two Variables and Bootstrap Inference

In this section we show the ArCo with two variables and some additional features from the fitArCo function such as bootstrap estimation of the couterfactual confidence bands, which have the advantage that it can be calculated for any model chosen by the user.

If the argument `boot.cf` is set to TRUE the function will automatically perform block bootstrap replications of the counterfactual, the number of replications and the block length are controlled through the arguments `R` and `l` respectively. The bootstrap procedure uses the package **boot** (Canty and Ripley, 2015) and is described as follows:

1. For $t < T_0$ prepare the data in a way that the vector $\boldsymbol{y}_1$ contains the response variable and a matrix $\boldsymbol{X}_1$ contains all regressors;
2. sample $T_0 - 1 - lag$ observations from $\{\boldsymbol{y}_1', \boldsymbol{X}_1\}$ using blocks of length l generating the bootstrap sample $\{\boldsymbol{y}_1', \boldsymbol{X}_1\}^r$;
3. estimate the model on $\{\boldsymbol{y}_1', \boldsymbol{X}_1\}^r$;
4. for $t \geq T_0$, compute the forecast with the model estimated on step 3 using $\boldsymbol{X}_2$ as the new data;
5. repeat steps 2-4 R times to obtain the counterfactual bootstrap distribution.

The `data.q2` dateset has two variables for 6 units. The number of observations is 100 and the treatment of adding up one standard deviation (0.84 for the first variable and 0.511 for the second variable) took place at $T_0 = 51$ on the first unit.

The following example estimates the ArCo using LASSO with the **glmnet** (Friedman et al., 2010). The function `fn` is `cv.glmnet` the function `p.fn` is `predict`[5]. The function `cv.glmnet` chooses the regularization parameter λ using cross-validation.

```
> data(data.q2)
> library(glmnet)
> set.seed(123)
> ArCo2 = fitArCo(data = data.q2, fn = cv.glmnet, p.fn = predict, treated.unit = 1,
t0 = 51, VCOV.type = "nw", VCOV.lag = 3, prewhitening.kernel = TRUE,boot.cf = TRUE,
R=200,  l=3)
> plot(ArCo2, display.fitted=TRUE ,confidence.bands = TRUE, alpha = 0.05)
> ArCo2$delta
                 LB     delta        UB
Variable1 0.6801909 0.7710389 0.8618870
Variable2 0.4225233 0.5071188 0.5917144
```

Figure 3 shows the counterfactual with 95% confidence bands. The red line shows if the chosen model has an acceptable performance in-sample. Moreover, the `$delta` shows that the treatment was different from zero on both variables and the point value of $\widehat{\boldsymbol{\Delta}}_T$ is close the value used to generate the data (0.84 and 0.511 for variables 1 and 2 respectively).

Practical Advice on the Use of the fitArCo Function

A potential issue for the practitioner is the choice of the function used to estimate the model in the first step of the ArCo (`fn`) method and the corresponding prediction function (`p.fn`). The default is to estimate the ArCo using OLS because it is a simple procedure and require no extra packages. However, the most recommended model approach is to use LASSO to estimate a linear model due to its good properties in high-dimensional settings and, more importantly, because most the asymptotic theory for the ArCo method was developed for LASSO (high-dimensions) and OLS (low-dimensions). Nevertheless, the user is free to use any other models such as Random Forest, Boosted Trees, Neural

[4]The plot.fitArCo function will automatically display legends if the argument display.fitted is set to TRUE. For those who use RStudio, watch out for the figure margins, especially in cases where $q > 1$ and the user wished to display multiple plots. If necessary use `x11()` to plot in a new window.

[5]Both functions are already on the format required by the fitArCo.

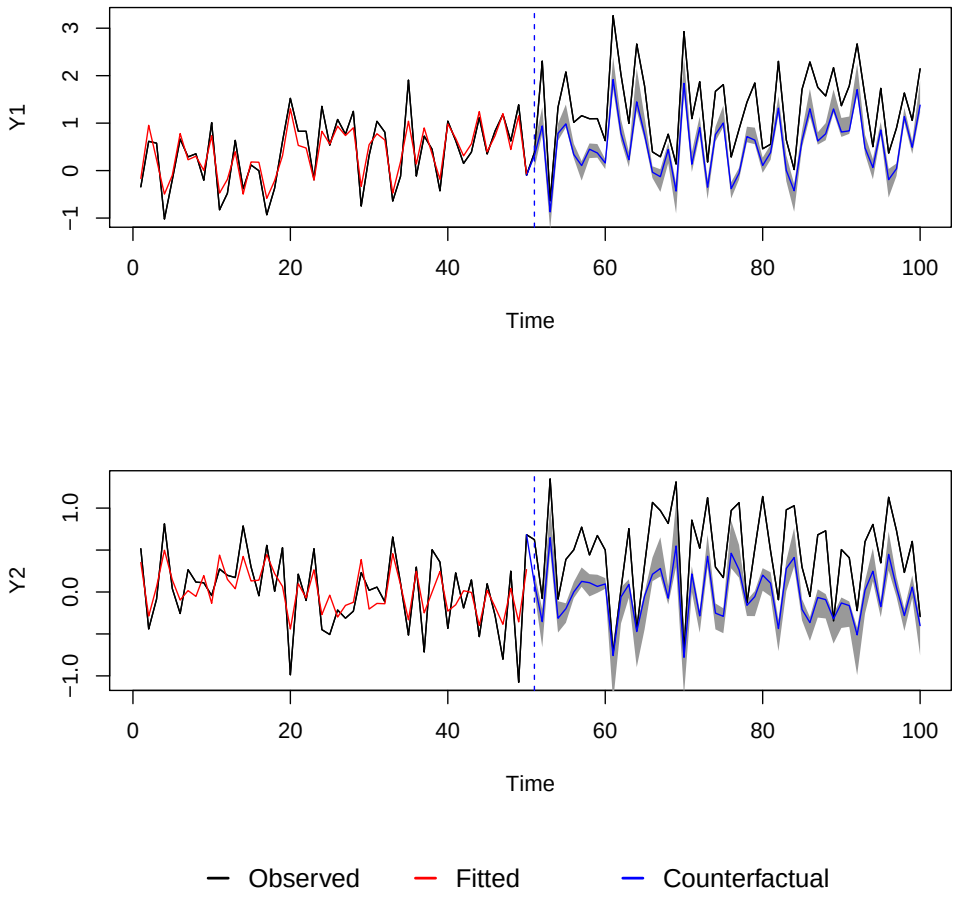


Figure 3: plot of the counterfactual from data.q2 unit 1, variables 1 and 2, with bootstrap confidence bands.

Networks, Factor Models, etc. In these cases, the user should be aware that the inference on the average treatment effect may not be valid and should only be used as an approximation. We recommend doing some simulation tests under the null hypothesis (no treatment) and under the alternative hypothesis (treatment different from zero) if the user chooses to use an estimation approach different from LASSO and OLS[6].

The `fitArCo` accepts extra arguments to be passed to the `fn` function. This is not particularly useful for LASSO or OLS, but if the user chooses models that require tuning (Boosted Trees, Random Forest, etc) extra arguments may be required.

The `fitArCo` function returns all the inference regarding the average treatment effect (confidence intervals and p-values). We included several built-in methods in case a robust covariance matrix is required. We recommend the user to test the first-stage residuals for autocorrelation and heteroskedasticity. If these features are present in the residuals the most common solution is to use Newey-West (`VCOV.type="nw"`) with Quadratic Spectral kernel (default). The optimal lag for the covariance matrix is calculated automatically but the user may force it through the argument `VCOV.lag`.

The entire list of what the function `fitArCo` function returns is:

- **cf:** $T_2 \times q$ matrix of the estimated counterfactual.
- **fitted.values:** $(T_1 - lag) \times q$ matrix with the fitted values for the pre-treatment period.
- **model:** A list with q estimated models, one for each variable. Each model is the exact output of the `fn` function.
- **delta:** The delta statistics for each variable and its confidence interval.
- **p.value:** The ArCo p-value.
- **data:** The data used.
- **t0:** The intervention period used.
- **treated.unit:** The treated unit used.
- **boot.cf:** A list with the bootstrap result (boot.cf=TRUE) or logical FALSE (boot.cf=FALSE). In the first case, each element in the list refers to one bootstrap replication of the counterfactual, i.e. the list length is R.
- **call:** The matched call.

The estimate_t0 Function

We mentioned earlier that in many cases the exact intervention period might be unknown, but it can be estimated using the maximization of the norm of $\mathbf{\Delta}_T(\lambda)$. This is precisely what the function `estimate_t0` does. The example below uses the dataset `data.q2` and LASSO to estimate the model as in the previous example. The arguments `start` and `end` determine the range of λ where the function should search for λ_0 and consequently T_0. The value of T_0 returned by the function was 53, which is very close to the 51 used to generate the data. The user can also see the value of the norm of $\mathbf{\Delta}_T$ correspondent to the estimated T_0.

```
> library(glmnet)
> set.seed(123)
> data("data.q2")
> t0=estimate_t0(data = data.q2, fn = cv.glmnet, p.fn = predict, treated.unit = 1,
start = 0.4, end = 0.9)
> t0$t0
[1] 53
> t0$delta.norm
[1] 0.9209421
```

The natural step that follows is to use the estimated T_0 on the `fitArCo` function. The results will obviously be very similar to the ones presented on the previous section because the estimated T_0 is nearly the same as the value used to generate the data.

[6]OLS should be used only for small problems with considerably more observations than variables in the period before the intervention.

Application to Real Data

This sections shows the **ArCo** package working on two different examples with real data. The first example is from Carvalho et al. (2016b) and it shows the effects on inflation of an tax evasion program on the Brazilian metropolitan area of São Paulo. The second example uses data from the package **Synth** (Abadie et al., 2011) to compare the ArCo with the Synthetic Control (Abadie and Gardeazabal, 2003; Abadie et al., 2010).

So far we have used OLS and cross-validation LASSO to make the examples as simple as possible. However, there are some advantages on selecting the LASSO regularization parameter using information criterion such as the Bayesian Information Criterion (BIC), which allow us to consistently estimate the degrees of freedom of the model (Zou et al., 2007). Moreover, choosing by information criterion is faster and avoids further complications created by cross-validation on time-series. For the next examples we will define the functions `fn` and `p.fn` to to select the model using the BIC as follows:

```
> fn=function (x, y){
+   n=length(y)
+   model = glmnet(x = x, y = y)
+   coef = coef(model)
+   df = model$df
+   yhat=cbind(1,x)%*%coef
+   residuals = (y- yhat)
+   mse = colMeans(residuals^2)
+   nvar = df + 1
+   bic = n*log(mse)+nvar*log(n)
+   selected= which(bic == min(bic))
+   return(coef[,selected])
+ }
>
> p.fn=function(model,newdata){
+   return(cbind(1,newdata) %*% model)
+ }
```

The Effects of an Anti Tax Evasion Program

In October 2007 the government of the Brazilian state of São Paulo launched an anti tax evasion program called *Nota Fiscal Paulista* (NFP). It consists of a refund from a state tax on the circulation of products and services. The NFP acts as an incentive for the consumer to ask for electronic sales receipt. The receipts give the consumer the right to participate on monthly lotteries promoted by the government and the tax rebate if his tax identifier number (CPF) is included in the receipt.

The NFP program received substantial support from the population. In January 2008, 413 thousand people were already registered in the program while in October 2013 this number of participants were more than 15 million.

The problem arises if we assume that there was some degree of tax evasion occurring before the intervention, and once the consumers start demanding the receipts, the sellers were forced to pay more tax and evasion became more difficult. If the sellers have some degree of market power they might increase prices in order to accommodate the extra tax. If these assumptions are true one would expect an upward movement on prices due to an increase on marginal costs. Therefore, the example presented in this section tries to investigate whether the NFP had and impact on consumers price in São Paulo.

The first sector to have the NFP was restaurants, therefore the Food Away from Home component (FAH) of the Consumer Prices Index (CPI) is a good measure to contemplate the entire program from the start. The sample is available on the **ArCo** package with the name of `inflationNFP`. It consists of nine Brazilian metropolitan areas including São Paulo from January 1995 to September 2009. Each metropolitan area has the interpretation of an ArCo unit. The variables are the FAH component of the CPI inflation and the monthly GDP growth for all metropolitan areas. The treatment happened at $T_0 = 34$, therefore, we have 33 observations to estimate the first step of the ArCo and 23 observations to calculate the counterfactual.

```
> library(glmnet)
> data("inflationNFP")
> t0=34
> ArCoNFP=fitArCo(inflationNFP,fn,p.fn,1,t0,VCOV.type = "nw",VCOV.lag = 2,
prewhitening.kernel = TRUE)
```

```
> plot(ArCoNFP,plot=1,display.fitted = TRUE)
> ArCoNFP$delta
                          LB         delta          UB
inflationFAH  0.226033437  0.4226199609 0.619206484
GDP          -0.004699182 -0.0006055599 0.003488062
> ArCoNFP$p.value
       Joint inflationFAH          GDP
1.333664e-04 2.514289e-05 7.718676e-01
```

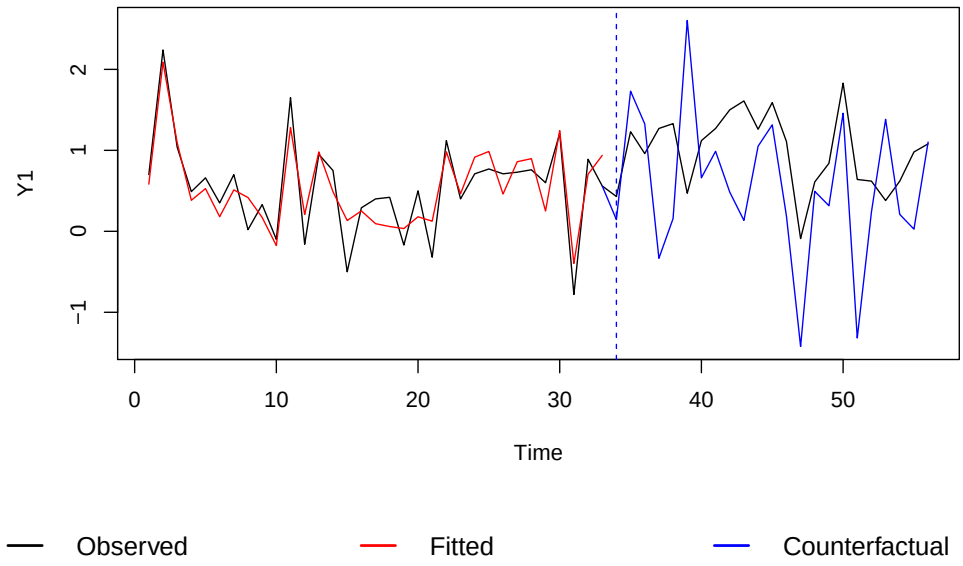


Figure 4: plot of the counterfactual of the CPI Inflation (FAH) from São Paulo, Brazil.

The `$delta` shows that in fact there was a significant impact on inflation caused by the NFP. The impact on the GPD was statistically zero. The `$p.value` relates to an overall effect considering all variables, and it also shows that the effects of the NFP were, in general, different from zero. Figure 4 shows that the LASSO had a good fit in sample, but not too good for us to be concerned with over-fitting. In nearly all periods the estimated counterfactual was below the realized value of the inflation after the NFP was introduced.

It is also interesting to see what happened with the CPI. The code below accumulates the realized inflation and the counterfactual. Figure 5 shows that the accumulated increase on the CPI (FAH) might have been significantly smaller without the NFP.

```
> FAHsp=inflationNFP$inflationFAH[,1]
> real=cumprod(1+FAHsp/100)
> cf=cumprod(1+c(FAHsp[1:(t0-1)],ArCoNFP$cf[,1])/100)
> fitted=cumprod(1+fitted(ArCoNFP)[,1]/100)
>
> plot(real,type="l",ylab="Y1",xlab="time")
> lines(c(rep(NA,t0-1),tail(cf,length(real)-t0+1)),col=4)
> lines(fitted,col=2)
> abline(v=t0,col=4,lty=2)
> legend("topleft",legend=c("Observed","Fitted","Counterfactual"),col=c(1,2,4),
lty=1,lwd=1,cex=1,seg.len = 1,bty="n")
```

A Comparison with Synthetic Control

The package **Synth** provides a user friendly implementation of the SC method and some datasets to test it. In this section we compare the SC and the ArCo by reproducing the exact examples in the `synth`

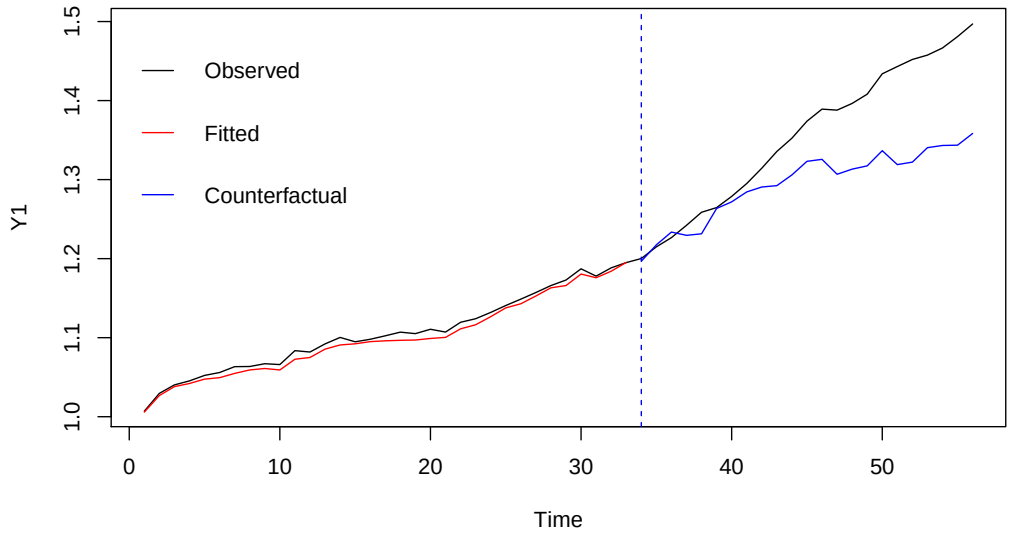


Figure 5: plot of the counterfactual of the CPI (FAH) from São Paulo, Brazil.

function from the **Synth** package. The first example is with generate data, the exact code is reproduced below. The variable of interest is Y and there are three controls. We can only use the control X_2 in the ArCo because X_1 and X_3 do not have enough observations. A crucial difference between the ArCo and the SC is that the second kills the time dimension, and therefore, require less data. The ArCo uses the entire temporal information and needs all the time-series to be complete. Therefore, we dropped the variables with missing values in the ArCo estimation for the next two examples. There is no built-in NA treatment in the **ArCo** package because it is as particular feature of each time-series problem. If the user is dealing with data in different periodicities, for example, monthly and quarterly, he/she could repeat the quarterly data in the months with NA or collapse the monthly data to quarterly data.

```
> # = Exact code reproduced from the first example in the ?synth documentation = #
> library(Synth)
> data("synth.data")
> dataprep.out<-
+   dataprep(
+     foo = synth.data,
+     predictors = c("X1", "X2", "X3"),
+     predictors.op = "mean",
+     dependent = "Y",
+     unit.variable = "unit.num",
+     time.variable = "year",
+     special.predictors = list(
+       list("Y", 1991, "mean"),
+       list("Y", 1985, "mean"),
+       list("Y", 1980, "mean")
+     ),
+     treatment.identifier = 7,
+     controls.identifier = c(29, 2, 13, 17, 32, 38),
+     time.predictors.prior = c(1984:1989),
+     time.optimize.ssr = c(1984:1990),
+     unit.names.variable = "name",
+     time.plot = 1984:1996
+   )
>
> t0 = Sys.time()
> synth.out <- synth(dataprep.out)
```

```
X1, X0, Z1, Z0 all come directly from dataprep object.


****************
 searching for synthetic control unit


****************
****************
****************

MSPE (LOSS V): 4.399697

solution.v:
 1e-10 0.00358062 0.2382334 0.3691776 1.218e-07 0.3890082

solution.w:
 0.2349748 0.007323657 0.02010757 0.1897415 0.4884164 0.05943599

> Sys.time()-t0
Time difference of 14.36551 secs
```

The synth.data is a panel with variables for the time and the units. We can transform it into a list using the function panel_to_ArCo_list. The code below shows the structure of the data, its transformation into list and runs the ArCo using LASSO selecting the best model using the BIC.

```
> head(synth.data)
  unit.num year                     name     Y        X1   X2   X3
1        2 1980 control.region.northeast 131.8        NA 21.5   NA
2        2 1981 control.region.northeast 128.7 0.2534060 24.1   NA
3        2 1982 control.region.northeast 127.4 0.2512521 23.8   NA
4        2 1983 control.region.northeast 128.0 0.2489048 21.6   NA
5        2 1984 control.region.northeast 123.1 0.2462865 23.9 17.9
6        2 1985 control.region.northeast 125.8 0.2434099 22.9 18.1
>
> # = Adjust the synth.data panel to be compatible with fitArCo = #
> ArCodata=panel_to_ArCo_list(synth.data,"year","unit.num",c("Y","X2"))
> ArCodata=lapply(ArCodata,function(x)x[as.character(1984:1996),])
>
> # = Run the ArCo = #
> t0 = Sys.time()
> ArCo3=fitArCo(ArCodata,fn,p.fn,treated.unit = 2,t0 = 9)
> Sys.time()-t0
Time difference of 0.01013398 secs
> ArCo3$delta[1,]
       LB     delta        UB
 7.816165 16.559440 25.302714
> ArCo3$p.value
       Joint            Y           X2
0.0006722134 0.0002055471 0.0233194437
>
> # = Plot the ArCo and the SC = #
> synthcf=dataprep.out$Y0plot%*%synth.out$solution.w
> plot(dataprep.out$Y1plot,type="l",ylim=c(80,180),ylab="Y",xlab="Time")
> lines(synthcf,col=2)
> abline(v=8,col=4,lty=2)
> lines(c(fitted(ArCo3)[,1], ArCo3$cf[,1]),col=4)
> legend("topleft",legend=c("Y","Synth","ArCo"),col=c(1,2,4),cex = 1,
+ seg.len = 1,bty = "n",lty=1)
```

The ArCo took 0.01 seconds to run by LASSO, while the SC took 14 seconds. Figure 6 shows that the two methods had similar results. We used 1992 as the first treated observation (T_0) in the ArCo.The in-sample and out-of-sample are plotted in the same line. Note that we had to drop two variables in the ArCo because of missing values.

The second example in the **Synth** package is to estimate the counterfactual of the GDP in the

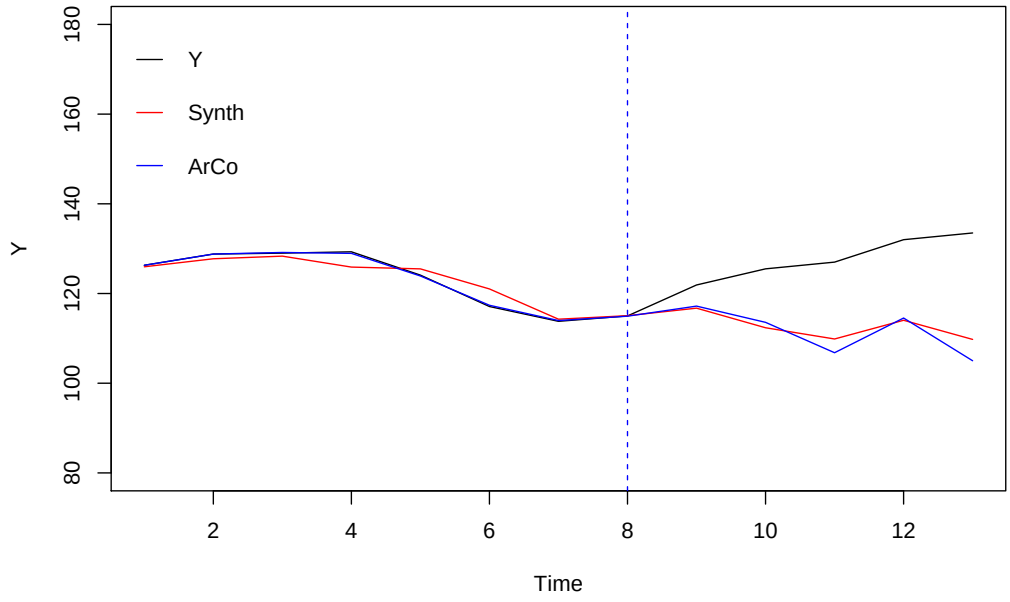


Figure 6: ArCo and Synthetic Control using the data from the first example from the synth function

Basque using other Spanish regions as controls. The main assumption is that the terrorism in the 70s had a negative impact on GDP. The dataset has GDP, investment and several other variables but we can only use the first two because of missing observations. Most of the other variables are related to sector production and education. For more details use ?basque.

```
> # = Exact code reproduced from the second example in the ?synth documentation = #
> # = Comments and printed outputs were omitted = #
> data("basque")
> dataprep.out <-
+   dataprep(
+     foo = basque
+     ,predictors= c("school.illit",
+                    "school.prim",
+                    "school.med",
+                    "school.high",
+                    "school.post.high"
+                    ,"invest"
+     )
+     ,predictors.op = c("mean")
+     ,dependent     = c("gdpcap")
+     ,unit.variable = c("regionno")
+     ,time.variable = c("year")
+     ,special.predictors = list(
+       list("gdpcap",1960:1969,c("mean")),
+       list("sec.agriculture",seq(1961,1969,2),c("mean")),
+       list("sec.energy",seq(1961,1969,2),c("mean")),
+       list("sec.industry",seq(1961,1969,2),c("mean")),
+       list("sec.construction",seq(1961,1969,2),c("mean")),
+       list("sec.services.venta",seq(1961,1969,2),c("mean")),
+       list("sec.services.nonventa",seq(1961,1969,2),c("mean")),
+       list("popdens",1969,c("mean")))
+     ,treatment.identifier  = 17
+     ,controls.identifier   = c(2:16,18)
+     ,time.predictors.prior = c(1964:1969)
+     ,time.optimize.ssr     = c(1960:1969)
+     ,unit.names.variable   = c("regionname")
```

```
+     ,time.plot            = c(1955:1997)
+   )
>
> dataprep.out$X1["school.high",] <-
+   dataprep.out$X1["school.high",] +
+   dataprep.out$X1["school.post.high",]
> dataprep.out$X1                 <-
+   as.matrix(dataprep.out$X1[
+     -which(rownames(dataprep.out$X1)=="school.post.high"),])
> dataprep.out$X0["school.high",] <-
+   dataprep.out$X0["school.high",] +
+   dataprep.out$X0["school.post.high",]
> dataprep.out$X0                 <-
+   dataprep.out$X0[
+     -which(rownames(dataprep.out$X0)=="school.post.high"),]
>
> lowest  <- which(rownames(dataprep.out$X0)=="school.illit")
> highest <- which(rownames(dataprep.out$X0)=="school.high")
>
> dataprep.out$X1[lowest:highest,] <-
+   (100 * dataprep.out$X1[lowest:highest,]) /
+   sum(dataprep.out$X1[lowest:highest,])
> dataprep.out$X0[lowest:highest,] <-
+   100 * scale(dataprep.out$X0[lowest:highest,],
+               center=FALSE,
+               scale=colSums(dataprep.out$X0[lowest:highest,])
+   )
>
> t0 = Sys.time()
> synth.out <- synth(data.prep.obj = dataprep.out)
> Sys.time()-t0
Time difference of 5.340531 secs
```

Since the GDP is non-stationary, it was first differentiated to estimate the ArCo. The code below prepares the data, which also comes in a panel, estimates the ArCo using LASSO and the BIC to select the best model and rebuild the GDP on its original level to compare with the SC. Since we do not know exactly when in the 70s the effects on the GDP begun, we also estimated T_0 to be used in the ArCo.

```
> # = Adjust the panel to fitArCo = #
> leveldata=panel_to_ArCo_list(basque,"year","regionno",c("gdpcap","invest"))
> # = GDP should be differentiated (delta(log)) = #
> diffdata=leveldata
> diffdata$gdpcap=diff(log(diffdata$gdpcap),1)
> diffdata$gdpcap=rbind(NA,diffdata$gdpcap)
> # = adjust sample size = #
> diffdata=lapply(diffdata,function(x)x[-c(1:10,42,43),])
>
> # = Estimate intervention period = #
> t0=Sys.time()
> t0.4=estimate_t0(diffdata,fn,p.fn,treated.unit =  17)
> # = Estimate ArCo = #
> ArCo4=fitArCo(diffdata,fn,p.fn,17,t0.4$t0)
> Sys.time()-t0
Time difference of 0.1583807 secs
> ArCo4$delta
                LB        delta          UB
gdpcap -0.01903297 -0.007195488 0.004641989
invest  0.61542806  1.789351051 2.963274038
> ArCo4$p.value
      Joint      gdpcap      invest
0.009720640 0.233505642 0.002812938
>
> # = Rebuild lvl counterfactual GDP = #
> y0=leveldata$gdpcap[10,17] # = First observation
> ArCocf=y0*exp(cumsum(c(fitted(ArCo4)[,1],ArCo4$cf[,1])))
```

```
> y.lvl=leveldata$gdpcap[as.character(1965:1995),17]
>
> # = Extract synth counterfactual and plot = #
> synthcf=(dataprep.out$Y0plot%*%synth.out$solution.w)[as.character(1965:1995),]
> xax=rownames(diffdata$gdpcap)
> plot(xax, y.lvl,type="l",ylab="Y",xlab="Time",ylim=c(0,12),lwd=3)
> lines(xax,ArCocf,col=4)
> lines(xax,synthcf,col=2)
> abline(v=xax[t0.4$t0],col=4,lty=2)
> legend("topleft",legend=c("Y","Synth","ArCo"),col=c(1,2,4),lty=1,
+ lwd=c(3,1,1),cex=1,seg.len = 1,bty="n")
```

The `$delta` showed that the intervention was not significant for the differentiated GDP, however, Figure 7 shows that there was a considerable impact on its level. The difference estimated by the ArCo is even bigger than the one estimated by the SC. It also seems that the ArCo counterfactual follows the trend before the intervention while the SC simply shifts the observed trend after the intervention.

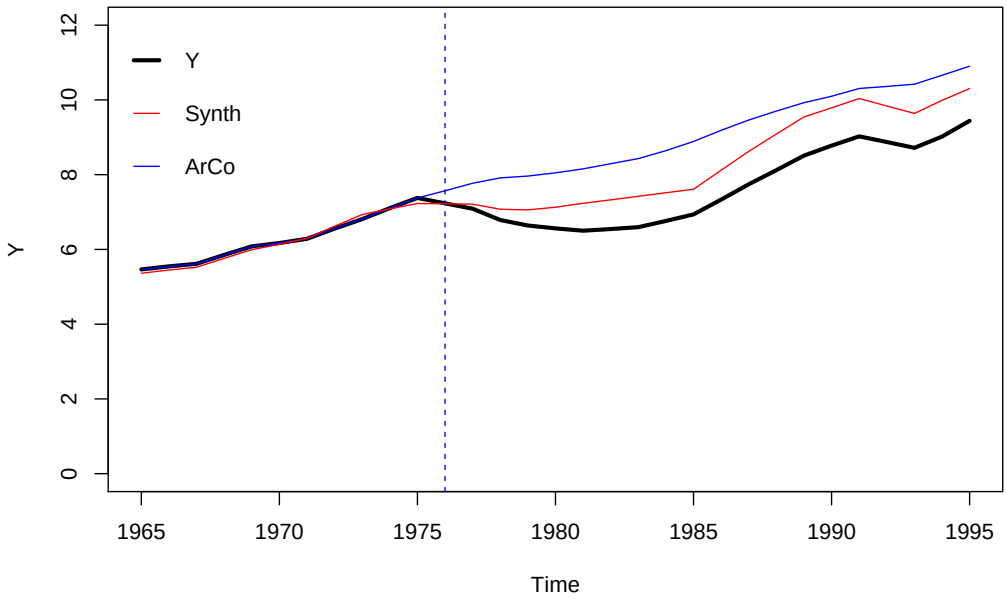


Figure 7: ArCo and Synthetic Control using the data from the Basque example from the synth function

Final Remarks

In this paper we have introduced the **ArCo** package which consists in a set of R functions implementing the ArCo method proposed by Carvalho et al. (2016b), which is a generalization of the Synthetic Control of Abadie and Gardeazabal (2003) and Abadie et al. (2010) and the panel approach put forward by Hsiao et al. (2012).

The ArCo method is a two-step methodology where in the first step a counterfactual is estimated based on a large, possibly high-dimensional, panel of time series from untreated peers. The package is very flexible and the user can choose any model of her preference such as, for example, LASSO regressions, random forests, boosted trees, neural networks, among others. In the second stage the average treatment effect is computed and a test for the null of no intervention effect is provided.

We believe that this R package will serve as an useful addition to the econometrics toolbox to conduct counterfactual analysis in comparative case studies.

Acknowledgements

We would like to thank an anonymous referee for very helpful and detailed comments.

Bibliography

A. Abadie and J. Gardeazabal. The economic costs of conflict: A case study of the Basque country. *American Economic Review*, 93:113–132, 2003. [p91, 92, 100, 106]

A. Abadie, A. Diamond, and J. Hainmueller. Synthetic control methods for comparative case studies: Estimating the effect of California's tobacco control program. *Journal of the American statistical Association*, 105:493–505, 2010. [p100, 106]

A. Abadie, A. Diamond, and J. Hainmueller. Synth: An R package for synthetic control methods in comparative case studies. *Journal of Statistical Software*, 42(13):1–17, 2011. [p91, 100]

D. W. K. Andrews and C. J. Monahan. An improved heteroskedasticity and autocorrelation consistent covariance matrix estimator. *Econometrica*, pages 953–966, 1992. [p94, 97]

S. Athey and G. W. Imbens. The state of applied econometrics - causality and policy evaluation. Technical report, arXiv:1607.00699v1, 2016a. [p91]

S. Athey and G. W. Imbens. The econometrics of randomized experiments. 1607.00698, arXiv, 2016b. [p91]

A. Canty and B. D. Ripley. *boot: Bootstrap R (S-Plus) Functions*, 2015. URL https://cran.r-project.org/web/packages/boot/. R package version 1.3-17. [p97]

C. V. Carvalho, R. P. Masini, and M. C. Medeiros. The perils of counterfactual analysis with integrated processes. Working paper, Pontifical Catholic University of Rio de Janeiro, 2016a. Available at SSRN: https://ssrn.com/abstract=2894065 or http://dx.doi.org/10.2139/ssrn.2894065. [p92, 93]

C. V. Carvalho, R. P. Masini, and M. C. Medeiros. ArCo: An artificial counterfactual approach for high-dimensional panel time-series data. Working paper, Pontifical Catholic University of Rio de Janeiro, 2016b. Available at SSRN: https://ssrn.com/abstract=2823687 or http://dx.doi.org/10.2139/ssrn.2823687. [p92, 93, 94, 95, 100, 106]

N. Doudchenko and G. W. Imbens. Balancing, regression, difference-in-differences and synthetic control methods: A synthesis. 22791, NBER, 2016. [p92, 93]

B. Ferman and C. Pinto. Revisiting the synthetic control estimator. Working paper, São Paulo School of Economics - FGV, 2016. [p92]

B. Ferman, C. Pinto, and V. Possebom. Cherry picking with synthetic controls. Working paper, São Paulo School of Economics - FGV, 2016. [p93]

Y. R. Fonseca, M. C. Medeiros, R. P. Masini, and G. F. R. Vasconcelos. *ArCo: Artificial Counterfactual Package*, 2017. URL https://CRAN.R-project.org/package=ArCo. R package version 0.1-2. [p93]

J. Friedman, T. Hastie, and R. Tibshirani. Regularization paths for generalized linear models via coordinate descent. *Journal of Statistical Software*, 33:1–22, 2010. [p97]

L. Gobillon and T. Magnac. Regional policy evaluation: Interactive fixed effects and synthetic controls. *Review of Economics and Statistics*, 98:535–551, 2016. [p92]

C. Hsiao, H. S. Ching, and S. K. Wan. A panel data approach for program evaluation: Measuring the benefits of political and economic integration of Hong Kong with mainland China. *Journal of Applied Econometrics*, 27:705–740, 2012. [p91, 92, 106]

W. Newey and K. West. A simple, positive semi-definite, heteroskedasticity and autocorrelation consistent covariance matrix. *Econometrica*, 55:703–08, 1987. [p94, 97]

R. Tibshirani. Regression shrinkage and selection via the LASSO. *Journal of the Royal Statistical Society. Series B (Methodological)*, pages 267–288, 1996. [p92]

H. Zou, T. Hastie, and R. Tibshirani. On the degrees of freedom of the Lasso. *The Annals of Statistics*, 35:2173–2192, 2007. [p100]

Yuri R. Fonseca
Department of Materials Science, Military Institute of Engineering (IME)
Praça Gen Tibúrcio 80, Praia Vermelha, Rio de Janeiro, RJ
Brazil
yuriresendefonseca@outlook.com

Ricardo P. Masini
São Paulo School of Economics - FGV-SP
Address Rua Itapeva 474, Bela Vista, São Paulo, SP
Brazil
ricardo.masini@fgv.br

Marcelo C. Medeiros
Department of Economics, Pontifical Catholic University of Rio de Janeiro (PUC-Rio)
Rua Marquês de São Vicente 225, Gávea, Rio de Janeiro, RJ
Brazil
mcm@econ.puc-rio.br

Gabriel F. R. Vasconcelos
Department of Electrical Engineering, Pontifical Catholic University of Rio de Janeiro (PUC-Rio)
Rua Marquês de São Vicente 225, Gávea, Rio de Janeiro, RJ
Brazil
gabrielrvsc@yahoo.com.br

PanJen: An R package for Ranking Transformations in a Linear Regression

by Cathrine Ulla Jensen and Toke Emil Panduro

Abstract **PanJen** is an R-package for ranking transformations in linear regressions. It provides users with the ability to explore the relationship between a dependent variable and its independent variables. The package offers an easy and data-driven way to choose a functional form in multiple linear regression models by comparing a range of parametric transformations. The parametric functional forms are benchmarked against each other and a non-parametric transformation. The package allows users to generate plots that show the relation between a covariate and the dependent variable. Furthermore, **PanJen** will enable users to specify specific functional transformations, driven by a priori and theory-based hypotheses. The package supplies both model fits and plots that allow users to make informed choices on the functional forms in their regression. We show that the ranking in PanJen outperforms the Box-Tidwell transformation, especially in the presence of inefficiency, heteroscedasticity or endogeneity.

Introduction

A model that fits data well but is unrelated to theory can only describe correlations. In contrast, a model with both a good fit and a theoretically sound foundation can give insights into hypotheses on causality. The functional form in a regression model describes the relationship between a dependent variable and its covariates. There are numerous examples of researchers who have neglected functional form relationships and applied the default linear relationship between variables in their regression models (Box, 1976; Breiman and others, 2001; Berk, 2004; Angrist and Pischke, 2010). From a superficial point of view, these models may provide efficient parameter estimates with narrow standard errors, high t-values, and significance. A strong non-linear relationship between a dependent variable and a covariate will often offer reasonable test statistics with a default linear functional form specification. However, positive test statistics are not the same as proof of a linear relationship. Even more important is the fact that a misspecified functional form can lead to an incorrect interpretation and prediction of the relationship between the dependent variable and a given covariate. The specification should be driven by theory with an a priori hypothesis of the relationship between the dependent and covariates.

PanJen was developed over several years of applied research on property value models. Here, the sales price is estimated as a function of its characteristics, such as the size of the living space, the number of rooms, and access to shopping. However, the package is applicable to most cases in which the relationship between a continuous dependent variable and its covariates is explored. In the applied econometrics literature, this task has commonly been solved using power transformations in initial analyses (Palmquist, 2006).

Two examples of power-transformations are *Box-Cox* and *Box-Tidwell* (Box and Tidwell, 1962; Box, 1976). The power-transformations are easy to use; the ability of these two power transformations to detect functional forms has been studied extensively in the academic literature, e.g., Kowalski and Colwell (1986); Brennan et al. (1984); Clark (1984), and they are still used in applied studies (Cohen et al., 2013; Farooq et al., 2010; Link, 2014; Joshi et al., 2017; Benson et al., 1998; Troy and Grove, 2008).

The popularity of these power-transformations is surprising given that their shortcomings are well described in the literature (Levin et al., 1993; Wooldridge, 1992a). While power transformations perform well in many circumstances, they do not perform well in the presence of omitted variables, inefficiency, heteroscedasticity and endogeneity. Furthermore, a transformation can be challenging to interpret, does not necessarily relate back to a theory-driven hypothesis and does not detect whether the relationship changes across the distribution of the dependent variable.

Another approach to the functional form issue is to abandon the parametric model and approach the challenge from a non- or semi-parametric angle. In the academic literature, a number of alternatives have been proposed and used, such as non-parametric or semi-parametric methods (Anglin and Gençay, 1996; Gençay, 1996; Clapp and Giaccotto, 2002; Bin, 2004; Geniaux and Napoléone, 2008). Non- or semi-parametric models provide data-driven approaches to establish the relationship between a dependent variable and covariates. A non-parametric model can be attractive, because the functional form is revealed by the data instead of being be predefined by the researcher. However, the gained flexibility of non-parametric analysis comes at the cost of more difficult interpretation of the estimates, which is perhaps why parametric models are often used in applied work.

In non-parametric models, the relationship is fitted to the sample to the extent that the estimated relationship is at risk of being over-fitted. In other words, the estimated effect captures random error or

noise in combination with the underlying relationship in the population (Wood, 2006). Additionally, a central critique concerning this approach is that the results from a non-parametric model are difficult to generalize or extend outside of the sample (McMillen and Redfearn, 2010). Even so, a non-parametric model holds great potential in exploratory analysis.

We are certainly not the first researchers to consider the possibility of utilizing the apparent advantages of the non-parametric modelling to explore functional form relationships in parametric modelling. The literature on using a non-parametric model to test different parametric specifications is large (González-Manteiga and Crujeiras, 2013). The primary approach in the existing literature has been to test a parametric version of a model against a non-parametric version (Wooldridge, 1992b; Horowitz and Härdle, 1994; Zheng, 1996; Li et al., 2016). However, to the best of our knowledge, none of these tests have been widely adopted in the empirical literature. **PanJen**is developed to allow applied researchers to utilize non-parametric estimation to identify a better parametric functional form. The package offers a test based on well-established measures and is provided on a well-established software platform. We believe that very few empirical researchers know about the existing tests, and the few that do perceive them as too complicated due to their non-parametric basis. **PanJen** offers the user a transformation-ranking based on the parametric transformation that captures most of the variance of the dependent variable. The ranking includes a non-parametric specification that can detect if the relationship between the dependent variable and covariate is non-stationary. In contrast to existing tests, semi-parametric transformations are only included as a benchmark rather than as an incremental part of the test. The engine in the *PanJen ranking* is the well-known Akaike Information Criterion (*AIC*) and Bayesian Information Criterion (*BIC*) measures. These model-fit measures are already widely applied in the empirical literature and should not be a hindrance for the empirical researcher. With the *PanJen ranking*, we hope to introduce an approach that will make applied researchers explicitly consider functional form in their parametric models.

In the next section, we briefly describe the main idea behind the ranking in the package (*PanJen ranking*. In the following section, we describe the workhorse behind it, the Generalized Additive Model (*GAM*). In section 4, we explain how the *PanJen ranking* works. In section 5, we illustrate how to use the package by using a real example from our research. Section 6 offers a comparison between the *PanJen ranking* and the *Box-Tidwell transformation*. We simulated 10,000 datasets and recovered the functional form of one variable in a model with different impediments to show the merits of **PanJen** relative to a conventional approach. In section 6, we conclude the paper with a short discussion of when and how the researcher should use the package with an emphasis on the risk of pre-test bias.

The main idea of the *PanJen ranking*

PanJen is built on the idea that the choice of a functional form can be guided by model fit. In the *PanJen ranking*, a given number of models that vary only in the transformation of one covariate are estimated. One of these transformations is a so-called function that for now we will simply note makes this one model semi-parametric. All the estimated models are then ranked according to their BIC). The BIC provides a relative goodness-of-fit measure that accounts for the complexity of the model. More formally, the *PanJen ranking* estimates a model $Y = \beta_0 + X\beta_k + g(x)\beta_l + \varepsilon$ where Y is the dependent variable, ε is an i.i.d. error-term, X is a vector of k of covariates, β_k is the corresponding vector of parameter estimates, $g(x)$ represents a set of functional form transformations among the set:

$$g(x)\beta_l = \left\{ \frac{1}{x^2}\beta_1, \frac{1}{x}\beta_1, \frac{1}{\sqrt{x}}\beta_1, log(x)\beta_1, \sqrt{x}\beta_1, x\beta_1, x^2\beta_1, x\beta_1 + x^2\beta_2, f(x), 0 \right\}$$

where β_l the corresponding l parameter estimates for the parametric transformations. In the last two transformations, there is no parameter estimate for $g(x)$, because $f(x)$ is the non-parametric smoothing and 0 leaves out the explanatory variable.

The ranked BIC-values show how each transformation performs relative to the others. The semi-parametric transformation allows the user to assess how well parametric transformations perform relative to a flexible semi-parametric function. If the data generation process does not resemble any of the parametric transformations, the smoothing function will still capture the relationship. The BIC scores are supplemented by the closely related AIC. In practice, both the AIC and the BIC penalize model complexity, although the penalty term in BIC is larger than in AIC (Burnham and Anderson, 2004). The smoothing function is highly flexible, but the flexibility comes at a cost. Therefore, it is not necessarily ranked the highest since both AIC and BIC penalize the model complexity. There is no objective and transparent way to choose between the measures. The right measure depends on the user's a priori theory of the data generation process. If the users assume that one of their models perfectly fits the underlying data generation process, then the BIC is the right measure. If they instead

assume that the underlying data generation process is extremely complex and none of the possible models will be able to perfectly capture it, then AIC is the right measure (Aho et al., 2014).

The *PanJen ranking* is supported by a plot function that graphically outlines the relationship between the dependent variable and covariate. The plot is created by predicting the dependent variable using the median for all independent variables other than the one in question. The variable in question varies across a scale from the 5th quantile to the 95th quantile of the actual distribution in the dataset. The plot shows the user how each transformation captures the relationship across the distribution of the dependent variable. If the smoothing far outperforms all parametric transformations, the reason may be that the relationship changes across the distribution and the proposed simple parametric transformation does not capture the relationship between the dependent and independent variable. The plot will reveal this.

A semi-parametric model for benchmark

We estimate the parametric transformations using the Generalized Linear Model (*GLM*) and the semiparametric using GAM. GAM is a special case of the Generalized Linear Model (GLM) in which it is possible to include one or more so-called smoothing functions. A smoothing function is a non-parametric way to include a continuous covariate in a parametric model and make it semi-parametric.

The GAM can be written as follows:

$$Y_i = X_i\beta + f_1(x_{1i}) + \epsilon_i \tag{1}$$

Y_i is the dependent variable of observation i. It is distributed as an exponential family distribution, e.g.the normal, the gamma or chi-square distribution. Xi is a matrix of covariates that are parametrically related to the dependent variable. β is the corresponding vector of the parameter estimate, and f_i is a smoothing function of covariate x_{1i}.

The GAM provides a flexible specification of a covariate by only specifying it as a smoothed function. By entering a variable with a smoothing function, the researcher does not specify a functional form, but instead lets the data speak. The smoothing function comprises the sum of k thin plate regression spline bases $b_h(\bullet)$ multiplied by their coefficients. It is estimated as follows: $f = \sum_{h=1}^{k} \beta_h b_h(x_1)$. The non-parametric component of the model is fitted with a penalty on *wiggliness* (how flexible the smoothing is). The penalty, θ, is determined from the data using generalized cross-validation or related techniques. The penalty directly enters the objective function through an additional term capturing wiggliness in the smoothing function, i.e.,

$$\left\| Y_i - \widehat{Y}_i \right\|^2 + \theta \int f''(x_1)^2 dx_1 \tag{2}$$

Here, $\widehat{Y}$ is the fitted dependent variable, and the second derivatives of the smoothing function describe its wiggliness. We estimate the GAM using the mgcv R-package **mgcv** (Wood, 2017). For a thorough introduction to GAM, please see (Wood, 2006).

Using the package

We illustrate the use of **PanJen** using a hedonic house pricing model. The central idea is that the sale price of a home is a function of its characteristics, understood as both the characteristics of the home itself and its surroundings. The latter poses a problem in the empirical application of the hedonic method because observations can be correlated through space. A very flexible solution to this problem is to use the GAM framework to smooth over the x-y coordinates, thus allowing one to non-parametrically control for spatial correlations. von Graevenitz and Panduro (2015) illustrated the relationship between smoothing over space and classic spatial econometrics with weight-matrices and fixed spatial effects. They also showed that smoothing is a better alternative when the researcher does not know the underlying spatial data generation process. For recent applications, please see Rajapaksa et al. (2017) or Schäfer et al. (2017).

In our example here, we solely focus on the structural characteristics. These characteristics are measured by a range of variables. The researcher does not a priori know how the characteristics of the house are related to its price. For example, we expect the price to increase with the size of the home, but we do not know if that relationship is linear. It could be that going from 2 to 3 bedrooms is different than going from 7 to 8 bedrooms, i.e., we want to know if we should take account of marginally increasing or decreasing price-relationships. **PanJen** was developed to answer this type

of question by finding the functional form relationship between the home price and different home characteristics.

An example: the implicit price for living area

Names	Description
lprice	log transformed price in 1000 EUR
area	living area in square meters
age	build year
bathrooms	number of bathrooms
lake_SLD	distance to nearest lake in meters
highways	distance to nearest highway in meters
big_roads	distance to nearest large road in meters
railways	distance to nearest railway in meters
nature_SLD	distance to nearest nature area in meters

Table 1: Continous variables

The package features a dataset called 'hvidovre'. It includes 901 single detached homes sold between 2007 and 2010 within the Danish municipality of *Hvidovre*. The dataset was compiled from different Danish databases as a part of a larger hedonic study on households' willingness to pay for different urban and recreational services (Lundhede et al., 2013). We have 9 continuous and 7 dummy variables for quality at our disposal. In addition, the dataset includes 3 year dummies to control for price trends. The variables are listed in Tables 1 and 2:

Names	Description
rebuild70	home rebuild in 1970's
rebuild80	home rebuild in 1980's
rebuild90	home rebuild in 1990's
rebuild00	home rebuild in 2000's
brick	Construction made out of brick =1
roof_tile	roof made out of tiles =1
roof_cement	roof made out of cement=1
y7,y8,y9	home sold in 2007, 2008, or 2009

Table 2: Dummy variables

First, we load the package and the dataset:

```
> library(PanJen)
> data("hvidovre")
```

Then, we set up a `formula`-object. We log-transform the prices because this introduces flexibility and is the convention within the hedonic literature (Diewert, 2003). It is possible to test different transformations by simply transforming the variable or test different link-functions by leaving `variable` empty in `fform()`. Ten of the variables are dummy variables where transformations are irrelevant. We include only these in the first regression:

```
> formBase<-formula(lprice ~brick+roof_tile+roof_cemen
+                     + rebuild70+rebuild80+rebuild90+rebuild00+y7+y8+y9)
> summary(gam(formBase, method="GCV.Cp",data=hvidovre))

Family: gaussian
Link function: identity

Formula:
lprice ~ brick + roof_tile + roof_cemen + rebuild70 + rebuild80 +
    rebuild90 + rebuild00 + y7 + y8 + y9

Parametric coefficients:
```

```
            Estimate Std. Error t value Pr(>|t|)
(Intercept)  5.61058    0.02909 192.902  < 2e-16 ***
brick        0.11530    0.02660   4.334 1.63e-05 ***
roof_tile    0.06238    0.02328   2.679 0.007511 **
roof_cemen   0.08845    0.02969   2.979 0.002969 **
rebuild70    0.08357    0.03158   2.646 0.008285 **
rebuild80    0.14506    0.04382   3.310 0.000970 ***
rebuild90    0.14718    0.05356   2.748 0.006122 **
rebuild00    0.21120    0.04275   4.940 9.33e-07 ***
y7           0.14188    0.02653   5.347 1.14e-07 ***
y8           0.09193    0.02847   3.229 0.001286 **
y9          -0.09633    0.02784  -3.460 0.000566 ***
---
Signif.\ codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1


R-sq.(adj) =  0.146   Deviance explained = 15.5%
GCV = 0.089003  Scale est.\ = 0.087916  n = 901
```

This initial model explains nearly 15% of the variation in price. The next characteristic we want to control for is the size of the home. In the dataset, the living area in square metres is stored under ‘area’.

We start out by using the default transformations supplied by the **PanJen** function `fform()`. This function ranks the fit of nine predefined transformations and a smoothing. The mandatory inputs are the name of the dataset, the model formula and the new variable we wish to test using the *PanJen ranking*:

```
> PanJenArea<-fform(hvidovre,"area",formBase,distribution=Gamma(link=log))

             AIC    BIC ranking (BIC)
log(x)    435.07 497.51             1
x^2       435.39 497.84             2
x         437.07 499.52             3
smoothing 436.82 501.42             4
x+x^2     443.96 506.41             5
sqr(x)    444.00 506.44             6
1/x       445.66 508.11             7
base      511.35 568.99             8
[1] "Smoothing is a semi-parametric and data-driven transformation,
 please see Wood (2006) for an elaboration"
```

The results are ranked according to their BIC. Strictly according to this ranking, we should log-transform the area. This implies approximately that a % change in living area results in a % change in price. Given the respondent variable has been log-transformed previous to the model-fitting, any interpretation at the the original scale should be done with care , see e.g. (Barrera-Gómez et al., 2015).

The differences in the score for the four lowest BIC are small, and it might be a matter of differences in the tails of the distribution. This can be checked by plotting the predicted price against the area. The function `plotff()` generates a plot with the predicted price against the area from the 5th to the 95th percentiles with all other covariates variables at their median value:

```
>  plotff(PanJenArea)
```

The black line is the smoothing function. The log-squared and the linear specification closely follow this line. In conclusion, the implicit price for the living area is positive and slightly marginally declining. You can specify your own transformations using `choose.fform()`. In the following, we test three transformations: ‘$area$’, ‘$\log(area)$’ and ‘$area^2$’. We start out by defining a list of transformations:

```
> fxlist = list(linear = function(x) x,sqr = function(x) x^2,log=function(x) log(x))

> PanJenAreaC <- choose.fform(data=hvidovre, variable="area", base_form=formBase,
+ functionList=fxlist)

             AIC    BIC ranking (BIC)
log       290.04 352.49             1
linear    291.70 354.15             2
```

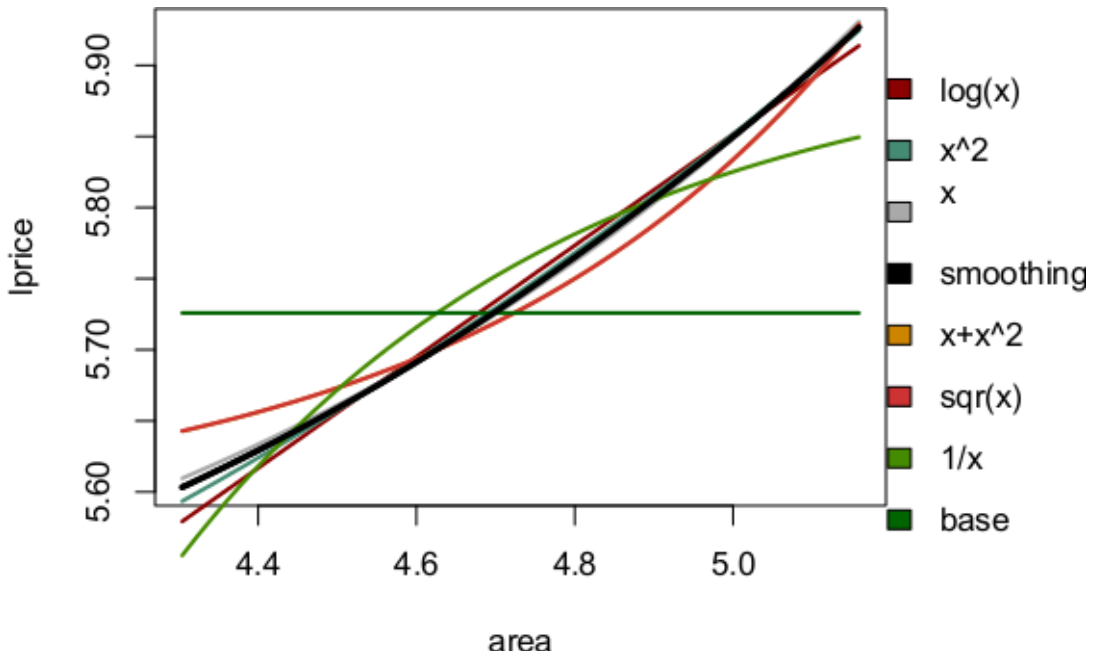


Figure 1: Plot generated by plotff

```
smoothing 291.53 355.91             3
sqr       299.15 361.60             4
base      379.20 436.84             5
[1] "Smoothing is a semi-parametric and data-driven transformation,
 please see Wood (2006) for an elaboration"


> plotff(PanJenAreaC)
```

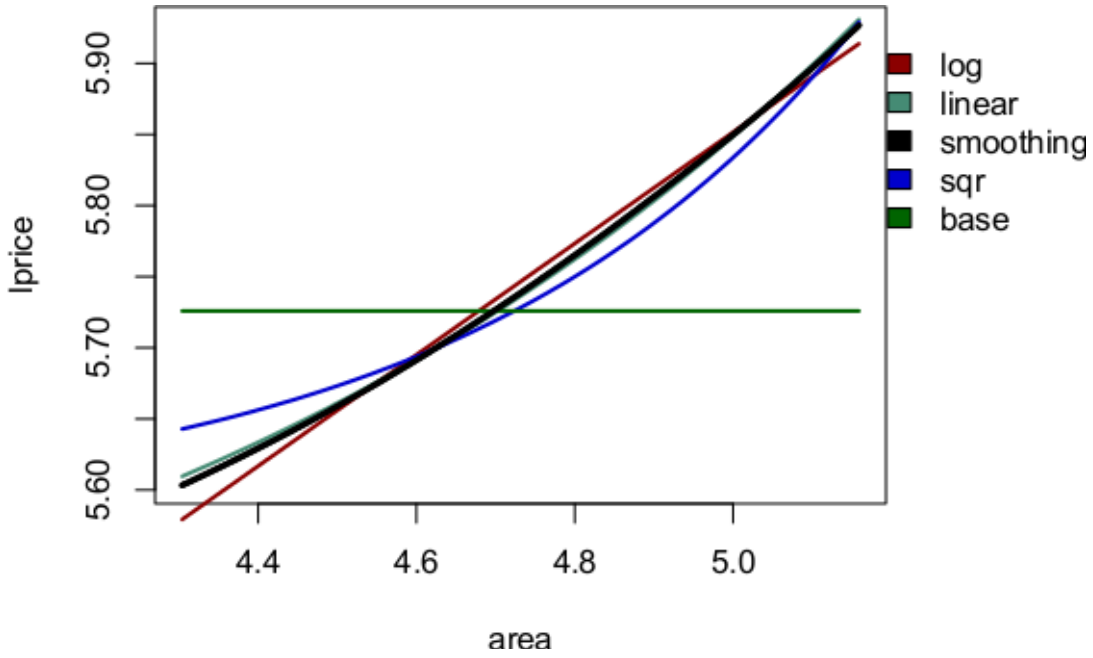


Figure 2: Plot with fewer transformations

We log transform the area and are now able to explain nearly 24% of the variation in price:

```
> hvidovre$larea<-log(hvidovre$area)
```

```
> formArea<-formula(lprice ~brick+rebuild80+rebuild90+rebuild00+y7+y8+y9+larea)

> summary(lm(formArea, data=hvidovre))

Call:
lm(formula = formArea, data = hvidovre)

Residuals:
    Min      1Q  Median      3Q     Max
-3.2109 -0.1000  0.0194  0.1479  0.9255

Coefficients:
            Estimate Std. Error t value             Pr(>|t|)
(Intercept)  3.76482    0.17863  21.076 < 0.0000000000000002 ***
brick        0.08695    0.02477   3.510             0.000471 ***
rebuild80    0.07485    0.04224   1.772             0.076714 .
rebuild90    0.11285    0.05084   2.220             0.026690 *
rebuild00    0.12213    0.04126   2.960             0.003159 **
y7           0.14800    0.02517   5.880        0.00000000579 ***
y8           0.09012    0.02704   3.333             0.000894 ***
y9          -0.10620    0.02645  -4.015        0.00006453604 ***
larea        0.40308    0.03793  10.627 < 0.0000000000000002 ***
---
Signif.\ codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 0.2818 on 892 degrees of freedom
Multiple R-squared:  0.2352,        Adjusted R-squared:  0.2284
F-statistic:  34.3 on 8 and 892 DF,  p-value: < 0.00000000000000022
```

A changing relationship

The first transformation was rather straightforward because the relation between area and price is stable across the price-distribution. The age of the home is a characteristic with which this is not the case. The reason is that the building changes over the years. As the home gets older, there is not a direct link between the age and state of the home. Where a newly built home is likely to be built with modern standards and tastes in mind, an old house can instead be charming and authentic. At the same time, in general, houses built during periods of building booms, which in Denmark were in the 1960s, are of lesser quality than those built in the 1950s or 1970s. We can show this by running `fform()` and plotting the results.

```
> PanJenAge<-fform(data=hvidovre,variable="age",base_form=formArea)

             AIC    BIC ranking (BIC)
base      285.82 333.85           1.5
1/x       285.82 333.85           1.5
log(x)    286.72 339.55           3.0
x         286.72 339.56           4.5
x^2       286.72 339.56           4.5
smoothing 274.92 359.94           6.0
x+x^2     359.69 407.73           7.0
sqr(x)    359.72 407.76           8.0


[1] "Smoothing is a semi-parametric and data-driven transformation,
 please see Wood (2006) for an elaboration"
> plotff(PanJenAge)
```

Based on BIC, the best parametric transformation is $1/x$. However, omitting the variable altogether, as described by the "base", works equally well.

The plot shows the relationship between age and price from the 5th to 95th percentile of the age

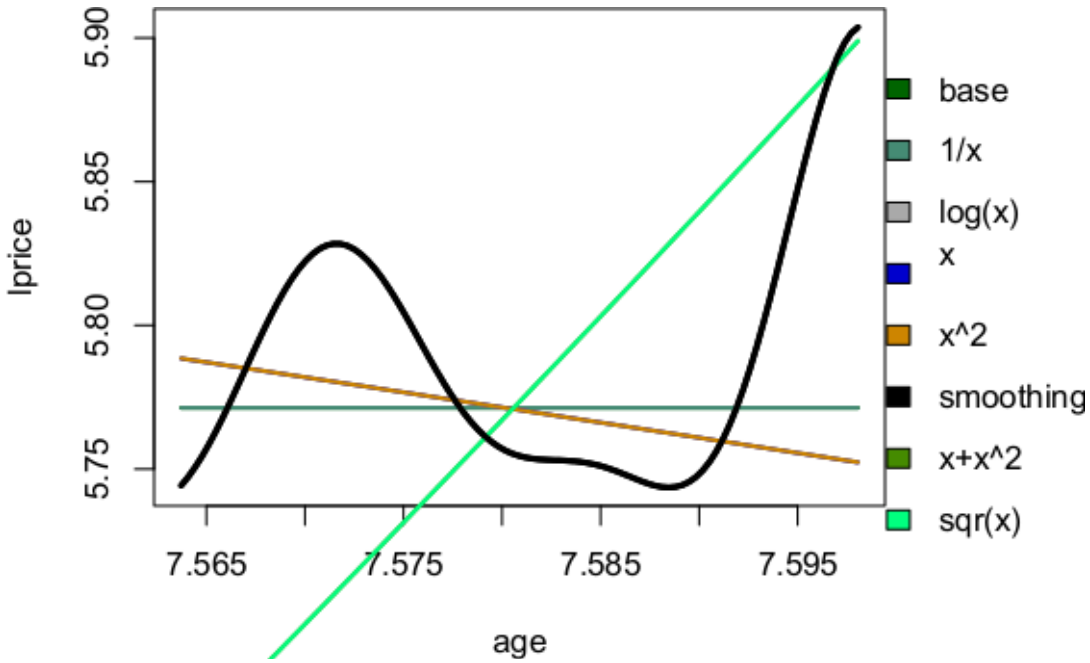


Figure 3: The relation between price and age

distribution. Given the plot, it is difficult to think of a parametric relationship that will capture this relationship. If the age of the home is somehow related to the research question, the best solution might be to use the smoothing function. If age is nothing more than a control variable, one could perhaps resort to interval dummies similar to the year dummies in the model. As a part of the final test of the model, it would be worthwhile to test to what degree the variable of interest is robust to the way age enters the pricing function. In our setting, what should be noted is that the complexity of the relation between age and price would have gone unnoticed if we had compared only the parametric transformations.

Interacting with other packages:

When you run `choose.fform()` or `fform()`, all generated models and datasets are stored in a new *list of list*-object. Within the list '`models`', all estimated models are stored as '`gam`', '`glm`' and '`lm`'-objects. This means that all objects used to create the fform output are easily available. The plotting function in **PanJen** is simple, but perhaps not enough when the researcher needs to produce plots for a third party. Here, we show how to use this to make a plot using base R, but we could just as well have used the **tlm** if we wanted a more detailed plot (Barrera-Gómez and Basagaña, 2017).

For example, you can create a new plot of just one transformation using `predict()` from **mgcv** and the base R plot. Here, we choose to look at '`log(area)`':

```
## The name of the models
> names(PanJenArea$models)
(1) "model_log(x)"    "model_x^0.5"     "model_smoothing" "model_x"
(5) "model_x+x^2"     "model_1/x"       "model_x^2"       "model_1/x^2"
(9) "model_base"


## getting the variable names used in the log model transformation
> namesVariables<-all.vars(formula(PanJenArea$models[["model_log(x)"]]))[1:11]

## creating a prediction dataframe with median values
> pred_frame<-data.frame(matrix(rep(sapply(hvidovre[namesVariables],median)
+ ,each=100),nrow=100))

## giving the prediction dataframe variable names
> names(pred_frame)<-namesVariables
```

```
## Finding the 0.05 quantile and the 0.95 quantile of the area variable
> min05<-as.numeric(quantile(hvidovre$area,0.05))
> max95<-as.numeric(quantile(hvidovre$area,0.95))

## Create prediction scale from  0.05 quantile to the 0.95 percentile
> pred_frame$area<-seq(min05,max95,length.out=100)

## predicting  lprice using the prediction dataframe
> pred_frame$var<-log(pred_frame$area)
> pred_frame$lprice=predict(PanJenArea$models[["model_log(x)"]],
+ newdata=pred_frame, type="response")

## Defining limits for plot
> limx=c(min(pred_frame$area),max(pred_frame$area))
> limy=c(min(pred_frame$lprice),max(pred_frame$lprice))


## Start plot
> plot(pred_frame$lprice~pred_frame$area, data=pred_frame, type="l",sub="",
> xlab="area",ylab="log(price)", lwd=3, col="black", xlim=limx, ylim=limy,
+ main="Price of living area")

## create legend
> legend(80,limy[2] , cex=1,lty=1,  "log(x)",  horiz=FALSE)
```

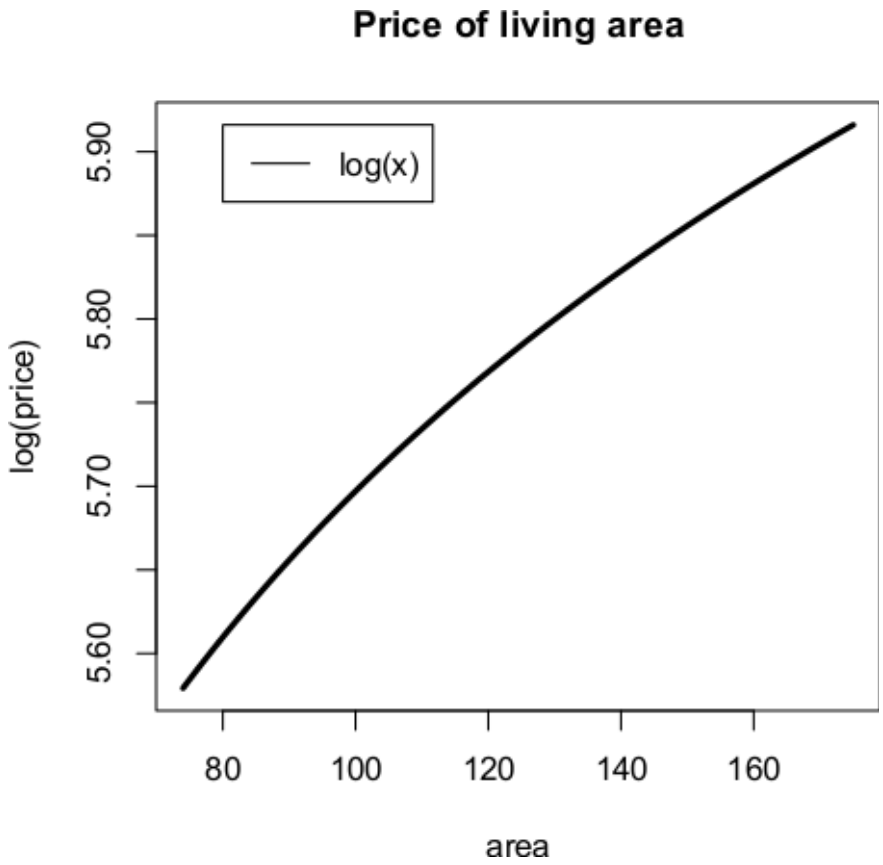


Figure 4: The price of living area

Monte Carlo simulations

We tested the performance of the *PanJen ranking* against the well-known *Box-Tidwell transformation* (Box and Tidwell, 1962). We choose this as a benchmark since it is the most structured choice applied in existing empirical work. Power transformations such as *Box-Cox* and *Box-Tidwell* were suggested in the 1960s by Box and Cox (1964) and Box and Tidwell (1962). The Box-Tidwell transformation identifies the transformation that minimizes non-normality in the error term and linearizes the relationship between the dependent variable and the covariate using a maximum likelihood function. Thus, the researcher can use the test to find the power-transformation with the highest likelihood. This section presents the results from nine Monte Carlo simulations in which the performance of the Box-Tidwell and PanJen is tested.

The simulations are centred on a base model:

$$Y = x_1\beta_1 + x_2\beta_2 + f(x_3) + \varepsilon \tag{3}$$

where x_3 is the variable of interest and x_1 and x_2 are two other covariates. The functional relationship between Y and $f(x_3)$ was then tested using *PanJen ranking* and Box-Tidwell transformations. Table 3 summarizes the results. The fourth and fifth columns show the share of times each method reported the true functional form. In the Box-Tidwell case, the transformation parameter was allowed to vary by up to 0.2 from the correct specification.

Simulation	Simulation description	PanJen	BoxTidwell
Identification	$f(x_3) = x_3^2$	100	97
Identification	$f(x_3) = x_3$	94	77
Identification	$f(x_3) = x_3^{0.5}$	100	100
Efficiency	$f(x_3) = x_3^2$, high variance	99	65
Collinearity	$f(x_3) = x_3^2$, x_2 correlated	100	97
Omitted variable	$f(x_3) = x_3^2$, omitted variable	100	93
Heteroscedasticity	$f(x_3) = x_3^2$, heteroskedastic	98	52
Endogeneity	$f(x_3) = x_3^2$, x_3 is endogenous	100	15
Misspecification	$f(x_3) = x_3^2$, x_2 misspecified	100	97

Table 3: Simulation results - 10.000 simulations

Each of the nine simulations tested the robustness of the methods in relation to different well-known econometric methods. Overall, the PanJen ranking performed acceptably. The method pointed to the correct functional form in 97 to 100% of the cases. The Box-Tidwell transformation performed just as well when the dataset was "well-behaved." It was already well-established in the literature that the method is sensitive to endogeneity, inefficient model estimates, heteroscedasticity and endogeneity, and this is also what we find in our study. In conclusion, PanJen Ranking performs better or just as well as Box-Tidwell.

Conclusion

In this paper, we present the **PanJen** package. We provide a simple and intuitive description of the *PanJen ranking*. Based on a house price dataset, we show how the functions in the package can be applied to determine the relationship between a dependent variable and its covariates. Furthermore, we compare the PanJen ranking method to the Box-Tidwell transformation and show that the PanJen ranking performs just as well as or better than the Box-Tidwell transformations. The Panjen ranking outperforms Box-Tidwell in situations where the model suffers from inefficiency, heteroscedasticity or endogeneity. In some circumstances, the theory provides little or no guidance on the functional relationship between the dependent and covariates in multiple regression models. In such circumstances, **PanJen** can support users in their decision on the functional form of the covariates. If the functional form relationship is more complex than a simple parametric transformation, we suggest considering a semi- or non-parametric model. The package has deliberately been restricted to test one covariate at a time without a silent output option. We want to deter the user from looping over every explanatory variable in search of a fit using the *PanJen ranking*, because this increases the pre-test bias. However, we also recognize that exploratory analysis is part of any empirical application of statistical modelling. Learning is a sequential process, and in many circumstances, we have not properly thought out an a priori hypothesis on which to base our models (Wallace, 1977). People perform exploratory model

estimations and in many cases under-report their approach. Even so, pre-test bias is not a problem caused or exacerbated by *PanJen Ranking*. Regardless of how a researcher performs multiple model estimations, the risk of pre-test bias can be reduced by adopting a sampling approach. The sampling approach can be implemented by dividing data into training and test datasets, where the explorative analysis is conducted on the first. It is our hope that people will use **PanJen** to improve their models by specifying relationships that more accurately fit their data. In doing so, users should consider *PanJen ranking* as a guide and not as a substitute for a priori hypothesis.

Cathrine Ulla Jensen
Department of Food and Resource Economics, Faculty of Science, University of Copenhagen
Rolighedsvej 23, 1958 Frederiksberg Copenhagen
Denmark
cuj@ifro.ku.dk

Toke Emil Panduro
Department of Food and Resource Economics, Faculty of Science, University of Copenhagen
Rolighedsvej 23, 1958 Frederiksberg Copenhagen
Denmark
tepp@ifro.ku.dk

Bibliography

K. Aho, D. Derryberry, and T. Peterson. Model selection for ecologists: The worldviews of aic and bic. *Ecology*, 95(3):631–636, 2014. ISSN 00129658, 19399170. URL http://www.jstor.org/stable/43495189. [p111]

P. M. Anglin and R. Gençay. Semiparametric Estimation of a Hedonic Price Function. *Journal of Applied Econometrics*, 11(6):633–648, 1996. ISSN 0883-7252. URL https://doi.org/10.1002/(sici)1099-1255(199611)11:6<633::aid-jae414>3.0.co;2-t. [p109]

J. D. Angrist and J.-S. Pischke. The credibility revolution in empirical economics: How better research design is taking the con out of econometrics. *The Journal of Economic Perspectives*, 24(2):3–30, 2010. URL https://doi.org/10.1257/jep.24.2.3. [p109]

J. Barrera-Gómez and X. Basagaña. *Tlm: Effects under Linear, Logistic and Poisson Regression Models with Transformed Variables*, 2017. URL https://CRAN.R-project.org/package=tlm. [p116]

J. Barrera-Gómez, X. Basagaña, and Maintainer Jose. Models with transformed variables: Interpretation and software. *Epidemiology*, 26(2):e16–17, 2015. [p113]

E. D. Benson, J. L. Hansen, A. L. Schwartz, and G. T. Smersh. Pricing residential amenities: The value of a view. *The Journal of Real Estate Finance and Economics*, 16(1):55–73, 1998. URL https://doi.org/10.1023/a:100778531592. [p109]

R. A. Berk. *Regression Analysis: A Constructive Critique*, volume 11. Sage, 2004. URL https://doi.org/10.4135/9781483348834. [p109]

O. Bin. A prediction comparison of housing sales prices by parametric versus semi-parametric regressions. *Journal of Housing Economics*, 13(1):68–84, 2004. ISSN 10511377. URL https://doi.org/10.1016/j.jhe.2004.01.001. [p109]

G. Box and P. Tidwell. Transformation of the independent variables. *Technometrics*, 1962. [p109, 118]

G. E. Box. Science and statistics. *Journal of the American Statistical Association*, 71(356):791–799, 1976. URL https://doi.org/10.1080/01621459.1976.10480949. [p109]

G. E. P. Box and D. R. Cox. An analysis of transformations. *Journal of the Royal Statistical Society. Series B (Methodological*, pages 211–252, 1964. ISSN 0035-9246. [p118]

L. Breiman and others. Statistical modeling: The two cultures (with comments and a rejoinder by the author). *Statistical Science*, 16(3):199–231, 2001. [p109]

T. P. Brennan, R. E. Cannaday, and P. F. Colwell. Office rent in the chicago cbd. *Real Estate Economics*, 12(3):243–260, 1984. URL https://doi.org/10.1111/1540-6229.00321. [p109]

K. P. Burnham and D. R. Anderson. Multimodel inference understanding aic and bic in model selection. *Sociological methods & research*, 33(2):261–304, 2004. URL https://doi.org/10.1177/0049124104268644. [p110]

J. Clapp and C. Giaccotto. Evaluating house price forecasts. *Journal of Real Estate Research*, 24(1):26, 2002. URL https://doi.org/10.1177/004209801247197 8. [p109]

J. A. Clark. Estimation of economies of scale in banking using a generalized functional form. *Journal of Money, Credit and Banking*, 16(1):53–68, 1984. [p109]

J. Cohen, P. Cohen, S. G. West, and L. S. Aiken. *Applied Multiple Regression/Correlation Analysis for the Behavioral Sciences*. Routledge, 2013. [p109]

W. E. Diewert. Hedonic regressions. a consumer theory approach. In *Scanner Data and Price Indexes*, pages 317–348. University of Chicago Press, 2003. [p112]

B. Farooq, E. Miller, and M. Haider. Hedonic analysis of office space rent. *Transportation Research Record: Journal of the Transportation Research Board*, (2174):118–127, 2010. URL https://doi.org/10.3141/2174-16. [p109]

G. Geniaux and C. Napoléone. *Semi-Parametric Tools for Spatial Hedonic Models: An Introduction to Mixed Geographically Weighted Regression and Geoadditive Models*, chapter 6, pages 101–127. Springer-Verlag, New York, NY, 2008. ISBN 978-0-387-76815-1. URL https://doi.org/10.1007/978-0-387-76815-1_6. [p109]

R. Gençay. A Statistical Framework for Testing Chaotic Dynamics via Lyapunov Exponents. *Physica D: Nonlinear Phenomena*, 89:261–266, 1996. ISSN 01672789. URL https://doi.org/10.1016/0167-2789(95)00230-8. [p109]

W. González-Manteiga and R. M. Crujeiras. An updated review of goodness-of-fit tests for regression models. *Test*, 22(3):361–411, 2013. URL https://doi.org/10.1007/s11749-013-0327-5. [p110]

J. L. Horowitz and W. Härdle. Testing a parametric model against a semiparametric alternative. *Econometric theory*, 10(05):821–848, 1994. URL https://doi.org/10.1017/s0266466600008872. [p110]

J. Joshi, M. Ali, and R. P. Berrens. Valuing farm access to irrigation in nepal: A hedonic pricing model. *Agricultural Water Management*, 181:35 – 46, 2017. ISSN 0378-3774. URL https://doi.org/10.1016/j.agwat.2016.11.020. [p109]

J. G. Kowalski and P. F. Colwell. Market versus assessed values of industrial land. *Real Estate Economics*, 14(2):361–373, 1986. URL https://doi.org/10.1111/1540-6229.00391. [p109]

A. Levin, R. Davidson, and J. G. MacKinnon. Estimation and Inference in Econometrics. *Journal of the American Statistical Association*, 89(427):1143, 1993. ISSN 01621459. URL https://doi.org/10.2307/2290953. [p109]

H. Li, Q. Li, and R. Liu. Consistent model specification tests based on k-nearest-neighbor estimation method. *Journal of Econometrics*, 194(1):187–202, 2016. URL https://doi.org/10.1016/j.jeconom.2016.03.004. [p110]

H. Link. A cost function approach for measuring the marginal cost of road maintenance. *Journal of Transport Economics and Policy (JTEP)*, 48(1):15–33, 2014. URL https://doi.org/10.1111/j.1467-9787.2010.00664.x. [p109]

T. Lundhede, T. E. Panduro, L. Kummel, A. Staahle, A. Heyman, and B. J. Thorsen. *Værdisætning Af Bykvaliteter-Fra Hovedstad Til Provins: Appendiks*. Institut for Fødevare-og Ressourceøkonomi, Københavns Universitet, 2013. [p112]

D. P. McMillen and C. L. Redfearn. Estimation and Hypothesis Testing for Nonparametric Hedonic House Price Functions. *Journal of Regional Science*, 50(3):712–733, 2010. ISSN 00224146. URL https://doi.org/10.1111/j.1467-9787.2010.00664.x. [p110]

R. B. Palmquist. Property value models. In K. G. Mäler and J. R. Vincent, editors, *Handbook of Environmental Economics*, volume 2, chapter 16, pages 763–819. Elsevier, 1 edition, 2006. URL https://EconPapers.repec.org/RePEc:eee:envchp:2-16. [p109]

D. Rajapaksa, M. Zhu, B. Lee, V.-N. Hoang, C. Wilson, and S. Managi. The impact of flood dynamics on property values. *Land Use Policy*, 69(Supplement C):317 – 325, 2017. ISSN 0264-8377. URL https://doi.org/10.1016/j.landusepol.2017.08.038. [p111]

P. Schäfer, P. Schäfer, J. Hirsch, and J. Hirsch. Do urban tourism hotspots affect berlin housing rents? *International Journal of Housing Markets and Analysis*, 10(2):231–255, 2017. [p111]

A. Troy and J. M. Grove. Property values, parks, and crime: A hedonic analysis in baltimore, {MD}. *Landscape and Urban Planning*, 87(3):233 – 245, 2008. ISSN 0169-2046. URL https://doi.org/10.1016/j.landurbplan.2008.06.005. [p109]

K. von Graevenitz and T. E. Panduro. An alternative to the standard spatial econometric approaches in hedonic house price models. *Land Economics*, 91(2):386–409, 2015. URL https://doi.org/10.3368/le.91.2.386. [p111]

T. D. Wallace. Pretest estimation in regression: A survey. *American Journal of Agricultural Economics*, 59 (3):431–443, 1977. URL https://doi.org/10.2307/1239645. [p118]

S. Wood. *Generalized Additive Models: An Introduction with R*. CRC press, 2006. ISBN 1584884746 (acid-free paper)\r9781584884743. URL https://doi.org/10.1111/j.1541-0420.2007.009053x. [p110, 111]

S. Wood. *Mgcv: Mixed GAM Computation Vehicle with Automatic Smoothness Estimation*, 2017. URL https://CRAN.R-project.org/package=mgcv. [p111]

J. M. Wooldridge. Some Alternatives to the Box-Cox Regression Model. *International Economic Review*, 33(4):p 935–955, 1992a. ISSN 00206598. URL https://doi.org/10.2307/2527151. [p109]

J. M. Wooldridge. A test for functional form against nonparametric alternatives. *Econometric Theory*, 8 (04):452–475, 1992b. URL https://doi.org/10.1017/s0266466600013165. [p110]

J. X. Zheng. A consistent test of functional form via nonparametric estimation techniques. *Journal of Econometrics*, 75(2):263–289, 1996. URL https://doi.org/10.1016/0304-4076(95)01760-7. [p110]

NA

Tackling Uncertainties of Species Distribution Model Projections with Package mopa

by M. Iturbide, J. Bedia, and J.M. Gutiérrez

Abstract Species Distribution Models (SDMs) constitute an important tool to assist decision-making in environmental conservation and planning in the context of climate change. Nevertheless, SDM projections are affected by a wide range of uncertainty factors (related to training data, climate projections and SDM techniques), which limit their potential value and credibility. The new package **mopa** provides tools for designing comprehensive multi-factor SDM ensemble experiments, combining multiple sources of uncertainty (e.g. baseline climate, pseudo-absence realizations, SDM techniques, future projections) and allowing to assess their contribution to the overall spread of the ensemble projection. In addition, **mopa** is seamlessly integrated with the climate4R bundle and allows straightforward retrieval and post-processing of state-of-the-art climate datasets (including observations and climate change projections), thus facilitating the proper analysis of key uncertainty factors related to climate data.

Introduction

Species Distribution Models (SDMs) are statistical tools used for the generation of probabilistic predictions of the presence of biological entities in the geographical space (Guisan and Zimmermann, 2000; Elith and et al, 2006). SDMs operate through the establishment of an empirical link between known presence locations and the physical characteristics of their environment. A particular case is that of *Climate Envelope Models* (CEMs), where appropriate climatic variables are used as predictors to characterize the climatic conditions where a species can potentially live —typically in the form of *bioclimatic* variables (Nix, 1986; Busby, 1991). In the context of climate change, SDMs have become a valuable tool for the vulnerability and impact assessment community, as a means of estimating distribution shifts due to climate variations, a problem of current interest in environmental conservation studies (see e.g.: Araújo et al., 2004; Hamann and Wang, 2006; Jeschke and Strayer, 2008). These studies require suitable climate products to produce models at an adequate spatial resolution and varying geographical extents –up to global–, including historical climate databases (i.e. high resolution gridded observations) and future climate projections for different emission scenarios. However, the intricacy of climate data retrieval and post-processing of the existing climate products (e.g. the global and regional climate change projections available from the Earth System Grid Federation, ESGF, Taylor et al., 2011) has resulted in a wide use of ready-to-use products without considering their limitations for a particular case study (see Bedia et al., 2013). In this paper we fill this gap with the package **mopa** (Species Distribution MOdeling with Pseudo-Absences), which has been developed in the framework of the climate4R bundle for climate data access and post-processing, thus facilitating the use of state-of-the-art global and regional climate data for SDM projections.

Despite the increased use of future SDM projections as a support tool for decision-making in biological conservation, the communication of the inherent uncertainties of these products remains as an ongoing challenge (see, e.g. Araújo et al., 2005; Beaumont et al., 2008; Fronzek et al., 2011). A common approach to tackle different sources of uncertainty is based on producing *ensembles* of future SDM projections that encompass a wide range of variability by considering multiple choices of each of the factors/components involved in the modeling and projection chain (Araújo and New, 2007; Buisson et al., 2010; Bagchi et al., 2013). However, there are important sources of uncertainty that are rarely quantified, yet crucial, in order to assess the credibility of the future distributions, such as the training data (including the baseline climate) used to fit the SDMs characterizing the ecological niche (Mateo et al., 2010; Bedia et al., 2013; Baker et al., 2016), the varying extrapolation ability outside the training period/spatial extent of the different SDM techniques (known as SDM *transferability* in time/space; Bedia et al., 2011; Fronzek et al., 2011), the Global/Regional Climate Model (GCM/RCM) projections and biases (Turco et al., 2013) and others (see e.g.: Falloon et al., 2014, for an overview). Moreover, the *ensemble* approach has also limitations, since it assumes that all SDMs are equally transferable to climate change conditions, thus posing the risk of diluting insightful model signals with noise and error from less useful or defective SDMs forming the ensemble (Thuiller et al., 2004; Peterson et al., 2011).

The package **mopa** here presented has been designed to facilitate the design and analysis of comprehensive multi-factor SDM ensemble experiments, exploring different uncertainty factors such

as presence data sets, pseudo-absence realizations, baseline climate, modeling algorithms, and future climate projections. Moreover, **mopa** provides variance partition tools to assess the contribution of the different factors to the overall uncertainty/spread of the ensemble projection. We illustrate the functionality of the package with the case-study presented in Iturbide et al. (2018), focusing on the impact of the pseudo-absence data in the future distribution of a specific Oak phylogenetic group in Europe resulting from an ensemble of SDM projections considering three factors: 1) different SDMs techniques, 2) different realizations of randomly generated pseudo-absence data and 3) different climate projections produced over Europe from an ensemble of RCMs. The analyses undertaken with **mopa** reveal the sensitivity of SDMs to the pseudo-absence samples, affecting model stability and transferability to new climate conditions, with important implications for the construction of the final ensemble projections. We use and provide publicly available data to guarantee the reproducibility of the results.

mopa and the climate4R bundle for climate data access

The numerous climate databases available (both baselines and future projections) are scattered across many different repositories with various file formats, variable naming conventions, etc. sometimes requiring relatively complex, time-consuming data downloads and error-prone processing steps (e.g. bias correction) prior to SDM development. This is also a major barrier for research reproducibility and data exchange. The climate4R bundle is a set of R packages specifically designed to ease climate data access, analysis and processing in a straightforward manner, tailored to the needs of the impacts and vulnerability assessment community. Further details and references to worked examples and tutorials can be found for instance in Cofino et al. (2017), Bedia et al. (2017) and Frías et al. (2018). With this regard, **mopa** was developed as part of the climate4R ecosystem, so that typical climate data operations for SDM applications and conversion features to the data type handled by **mopa** are provided. Additionally, **mopa** includes a user guide with an end-to-end worked example of climate data retrieval, transformation and SDM development: `help(package = "mopa")`.

The "niche" of mopa within the "SDM ecosystem" in R

The popularity of R and its excellent statistical modeling and spatial analysis support has favored the development of specific, well-established and actively maintained packages for SDM construction and analysis, such as **sdm** (Naimi and Araújo, 2016), **biomod2** (Thuiller et al., 2016), **dismo** (Hijmans et al., 2017) and **SDMTools** (Van der Wal et al., 2014), some of them also implementing pseudo-absence data generation and ensemble building utilities. For instance, both **sdm** and **biomod2** implement methods for building ensemble projections based on model performance in the calibration phase —e.g. by discarding or weighting the obtained results—. On the contrary, **mopa** is oriented towards the design and analysis of multi-factor ensembles of future SDM projections (considering as potential factors the presence data sets, the pseudo-absence realizations, the baseline climate, the modeling algorithms, and the future climate projections). The analysis of the resulting ensemble allows, for instance, assessing the problem of SDM transferability, which can not be properly evaluated during model calibration.

Besides, unlike previously existing packages, **mopa** allows pseudo-absence data generation as an independent step prior to model fitting, thus providing a finer control to the user for the analysis of several alternative methods and specific tuning options. In addition, the novel Three-Step method for pseudo-absence data generation is implemented (TS hereafter, Senay et al., 2013; Iturbide et al., 2015), providing a convenient interface that allows a fine tuning of the technique with simple arguments. Furthermore, **mopa** is also seamlessly integrated with standard R packages for spatial data manipulation like **raster** (Hijmans, 2015) and **sp** (Pebesma and Bivand, 2005; Bivand et al., 2013), allowing their usage at any stage of the modeling process (e.g. for data visualization and post-processing), and also extensibility to other SDM tools available in **sdm**, **biomod2**, ..., also handling the same spatial data classes.

Input data pre-processing

Climate data

SDM predictor variables (in this case-study a number of bioclimatic variables, but not necessarily so) are introduced in the analysis as collections of **raster** objects of the classes `rasterBrick` or `rasterStack`, similarly as other SDM-oriented packages. For instance, here we use a set of present and future bioclimatic variables widely used in SDM applications based on precipitation and temperature climatologies (Busby, 1991), using the function `biovars` of package **dismo**. To this aim, we first exploit the climate4R functionalities to load and post-process observed precipitation and

temperature climatologies from the E-OBS gridded observational dataset (Haylock et al., 2008) and the simulations of 7 Regional Climate Model (RCMs) of the project ENSEMBLES (van der Linden and Mitchell, 2009, http://www.ensembles-eu.org) for the control (20C3M, 1971-2000) and future (A1B, 2071-2100) scenarios, including the application of bias-correction ("delta" method, e.g. Winkler et al., 1997; Zahn and von Storch, 2010).

```
> install.packages("mopa")
> library(mopa)
```

```
> destfile <- tempfile()
> url <- paste0("https://raw.githubusercontent.com/SantanderMetGroup/",
+   "mopa/master/data/biostack.rda")
> download.file(url, destfile)
> load(destfile, verbose = TRUE)
```

Species distribution data

Several impact studies indicate that species should be modeled by treating sub-specific groups of organisms independently (e.g.: distinct genetic linages) due to their differing adaptive responses to changes in their environment (Hernandez et al., 2006; Beierkuhnlein et al., 2011; Serra-Varela et al., 2015). Although this is not always possible, due to the rare availability of information on the distribution of sub-specific groups for most of species, **mopa** has been conceived with this idea in mind, being able to deal with several sets of presences simultaneously. This adds flexibility to the modeling process in order to carry out experiments considering different sub-collections of presences, not only for sub-specific analyses (Iturbide et al., 2015), but also to address the sensitivity of the modeled distributions to different characteristics of the training sample (e.g. the sample size, Hernandez et al., 2006; Mateo et al., 2010). Thus, the `Oak_phylo2` **mopa** dataset contains a named list of length two, containing the geographical coordinates of presence localities for two different Oak phylogenies (H01 and H11, Petit et al., 2002). More details about the source data are provided in the help file of the dataset.

```
> data(Oak_phylo2)
> help(Oak_phylo2)
> presences <- Oak_phylo2$H11
```

Geographic background

The geographic background is often defined as the spatial extent of the area considered in the SDM calibration stage. Here, we refer to the *background* as a regular, geo-referenced grid with a specific size and resolution, in which both the environmental variables and the presence localities are located, so its grid-points are the sampling units. Function `backgroundGrid` provides a simple way of generating a backgroud using a `raster-class` object as reference. It also includes an additional argument (`spatial.subset`) for spatial subsetting, set by a `raster::extent` object or by one or several sets of bounding-box coordinates, providing great flexibility and ease of use for the analysis of SDM spatial aspects. For instance, it allows straightforward exploration of SDM geographical transferability or performing cross-validation experiments based on spatial folds (e.g.: Randin et al., 2006). As a result, when the object `Oak_phylo2` is passed to `backgroundGrid`, two different backgrounds are created by default, each one spatially restricted by its phylogeny distribution (H11 and H01).

```
> bg <- backgroundGrid(raster = biostack$baseline$bio1)
```

A smaller domain than the previous one can be arbitrarily indicated by the user by providing a specific spatial extent:

```
> bg.subdomain <- backgroundGrid(raster = biostack$baseline$bio1,
                 spatial.subset = extent(c(-10, 35, 45, 65)))
```

Similarly, the user might be interested in a background strictly constrained by the bounding box of the actual species localities, by just passing to `spatial.subset` their coordinates:

```
> bg.species <- backgroundGrid(raster = biostack$baseline$bio1,
                 spatial.subset = presences)
```

Thus, the user has flexibility to perform further modifications of the background, so it would be also possible to discard specific areas based on expert knowledge (e.g. Serra-Varela et al., 2015). In this case study, we will retain the full background (`bg`) for further analyses.

Pseudo-absence generation

Most of SDMs require data not only from known presences of the biological entity, but also absence data in order to model the binary response presence/absence as a function of the different environmental variables. While the sampling efforts are typically focused on recording presence localities (atlases, natural history collections, targeted samplings, ...), in most cases there is no explicit information about the absence of the species. Therefore *pseudo-absence* generation is often required for SDM construction, by sampling the background of the study domain. Different methods have been proposed to this aim, whose choice has an important effect on the final SDM results, as highlighted in different previous studies (e.g.: Wisz and Guisan, 2009; Iturbide et al., 2015). However, there is no consensus on the best sampling design for generating pseudo–absences.

Pseudo-absence sampling in **mopa** is performed by the `pseudoAbsence` function. It implements a wide range of methodologies described in the literature (see Iturbide et al., 2015, for an overview and comparison of methods) for maximum user flexibility, but at the same time its arguments have been kept as simple as possible to ease its application (Table 1). Here, three methods are described: random sampling, random sampling with environmental profiling and the three-step method. Their main characteristics are next briefly described. A more extended explanation can be found in (Iturbide et al., 2015) and reference therein.

Argument	Description
`realizations`	Number of realizations of pseudo-absence generation
`exclusion.buffer`	Minimum distance to be kept between presence data and pseudo-absence data
`prevalence`	Proportion of presences against absences
`kmeans`	Performs a k-means clustering of the background to extract the pseudo-absences instead of sampling at random
`varstack`	`RasterStack` of variables for computing the k-means clustering

Table 1: Arguments of function `pseudoAbsences` controlling the parameter values involved in pseudo-absence generation.

Random Sampling (RS). The RS method is the simplest and most frequent way of generating pseudo-absences (Iturbide et al., 2015). In the next example three times more pseudo-absences than presences are generated at random, keeping a 0.249° ($\simeq$ 30 km) exclusion buffer around known presence localities. Ten pseudo-absence realizations are considered:

```
> pa_RS <- pseudoAbsences(xy = presences, background = bg$xy,
                realizations = 10, exclusion.buffer = 0.249,
                prevalence = -0.5)
```

As an alternative to random sampling, a stratified sampling approach can be performed, based on homogeneous environmental conditions. To this aim, a clustering of the environmental space is applied following Senay et al. (2013) by setting argument `kmean` to `TRUE`:

```
> pa_kmeans <- pseudoAbsences(xy = presences, background = bg$xy,
                exclusion.buffer = 0.249,
                prevalence = -0.5,
                kmeans = TRUE, varstack = biostack$baseline)
```

Random Sampling with Environmental Profiling (RSEP). The RSEP method imposes restrictions on the environmental range of the background to be sampled for pseudo-absences. In **mopa** this is done by performing an environmental profiling of the background (function `OCSVMprofiling`) that, following Senay et al. (2013), applies a one-class support vector machine algorithm (OCSVM, implemented in package **e1071**, Meyer et al., 2017) returning a binary (presence/absence) classification of the background gridboxes based solely on the presence information (`bg.profiled$presence` and `bg.profiled$absence` in the example below). Only the predicted absence background is then retained for pseudo-absence generation.

```
> bg.profiled <- OCSVMprofiling(xy = presences, varstack = biostack$baseline,
                background = bg$xy)
```

```
> pa_RSEP <- pseudoAbsences(xy = presences, background = bg.profiled$absence,
                realizations = 10, exclusion.buffer = 0.249,
                prevalence = -0.5)
```

Three-step method (TS). TS is based on imposing restrictions to both the environmental range and the spatial extent of the background from which pseudo-absences are sampled. This method has been shown to outperform other common approaches in terms of resulting SDM robustness (Iturbide et al., 2015). The TS method adds an additional step to the RSEP method, consisting on the partition of the background space (as yielded by RSEP) in multiple bands using different radius from presence localities. In the example below, multiple distance bands with an increasing radius of 30 km between each other are created (argument `by = 0.249`, in degrees). The first one (with the shortest radius from presence localities) is at 30 km from the closest presence point (`start = 0.249`), and the largest one (the longest radius from presences) is set by default to half the length of the diagonal of the background bounding-box (see Iturbide et al., 2015, for more details).

```
>  bg.radius <- backgroundRadius(xy = presences,
                background = bg.profiled$absence,
                start = 0.249, by = 0.249, unit = "decimal degrees")
>  pa_TS <- pseudoAbsences(xy = presences,
                background = bg.radius, realizations = 10,
                exclusion.buffer = 0.249, prevalence = -0.5)
```

A spatial representation of the results yielded by the pseudo-absence methods described is next generated (Fig. 1):

```
> # Generates Fig. 1
> par(mfrow = c(2, 2), mar = c(2, 2, 2, 1.2))
> # Panel 1a (Presence data)
> plot(bg$xy, pch = 18, cex = 0.4, col = "gray", asp = 1)
> points(presences, pch = 18, cex = 0.6, col = "red")
> # Panel 1b (RS method)
> plot(bg$xy, pch = 18, cex = 0.4, col = "gray", asp = 1)
> points(pa_RS$species1$PA01[[1]], pch = 18, col = "darkviolet", cex = .6)
> points(pa_kmeans$species1$PA01[[1]], pch = 18, col = "yellow", cex = .6)
> points(presences, pch = 18, cex = 0.6, col = "red")
> # Panel 1c (RSEP method)
> plot(bg.profiled$absence, pch = 18, cex = 0.4, col = "gray", asp = 1)
> points(bg.profiled$presence, pch = 18, cex = 0.4, col = "aquamarine")
> points(pa_RSEP$species1$PA01[[1]], pch = 18, cex = 0.6, col = "darkviolet")
> points(presences, pch = 18, cex = 0.6, col = "red")
> # Panel 1d (TS method)
> plot(bg.radius[[1]]$km3120, col = "gray", asp = 1, pch = 18, cex = 0.4)
> points(bg.profiled$presence, pch = 18, cex = 0.4, col = "aquamarine")
> for (i in 1:10) {
                l <- (11 - i) * 10
                points(bg.radius[[1]][[l]],
                col = gray.colors(10, start = .9,end = 0.1)[i],
                pch = 18, cex = 0.4)
        }
> points(pa_TS$species1$PA01[[50]], pch = 18, cex = 0.6, col = "darkviolet")
> points(presences, pch = 18, cex = 0.6, col = "red")
```

Thus, **mopa** allows for the generation of a wide range of combinations of environmental restriction criteria (using `OCSVMprofiling`) and spatial extent constraints (using `backgroundRadius`, see Table 2), providing unrivalled functionality for the development and inter-comparison of multiple pseudo-absence setups for SDM refinement and ensemble prediction generation.

SDM fitting and prediction

Model fitting

Once the pseudo-absence dataset(s) chosen by the user is(are) built, the `mopaTrain` function performs SDM fitting. The function is a wrapper for different statistical method implementations commonly

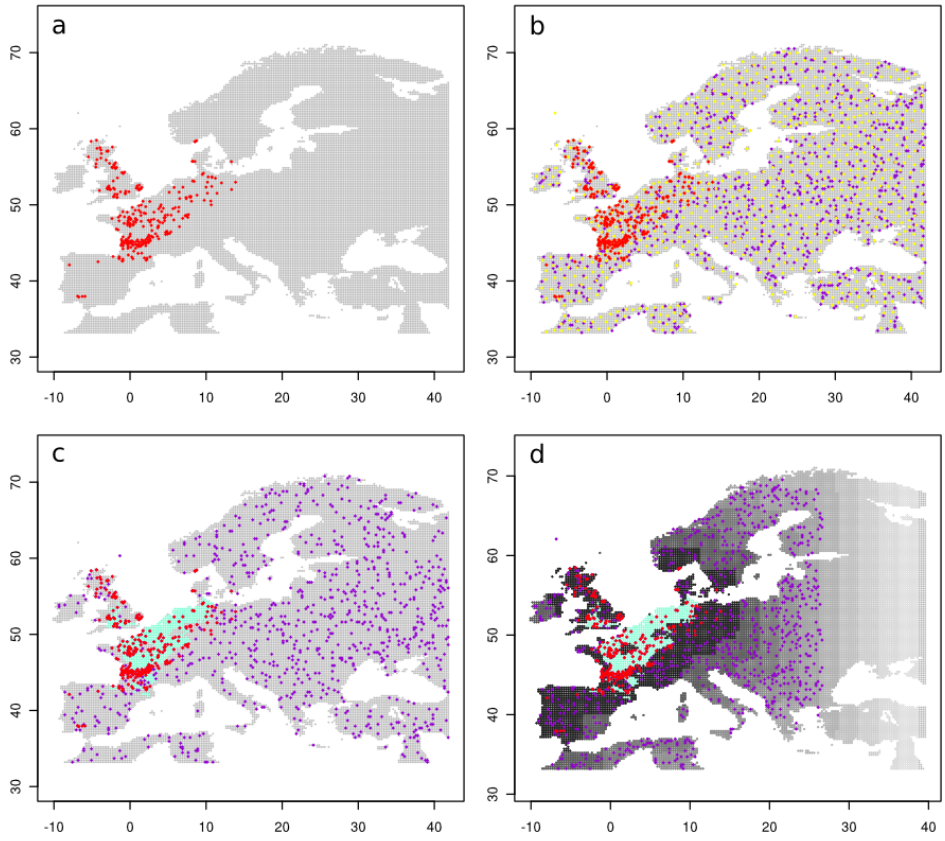


Figure 1: Pseudo-absence dataset maps, as generated by function `pseudoAbsences`. **(a)** Known presence locations of the Oak phylogeny H1 (red points) and initial background for pseudo-absence sampling (grey grid points). **(b)** pseudo-absences generated using the RS method randomly (purple points) and with k-means clustering (yellow points). **(c)** Pseudo-absences generated with the RSEP method (purple), where the turquoise area corresponds to the discarded suitable background space as identified by the OCSVM profiling approach. **(d)** TS approach. Environmentally stratified as RSEP (c), but also spatially stratified background, the different strata (spatial extents) identified by the different gray-scale colors. Pseudo-absences for one of the background extents (3120 km) are depicted as example (purple points).

used in SDM applications (see summary in Table 3). Moreover, `mopaTrain` adds extended functionality for cross-validation for each set of presence/absence data and for each different species contained in the presence dataset, as routinely done in SDM applications (see e.g.: Verbyla and Litvaitis, 1989). In the next line of code, the Oak H1 phylogeny is fitted using a generalized linear model (GLM, Guisan et al., 2002) and multivariate adaptive regression splines (MARS, Friedman, 1991), applying a 10-fold cross validation approach. Moreover, equal weighting of presences and pseudo-absences is indicated with the argument `weighting = TRUE` (see e.g.: Barbet-Massin et al., 2012).

```
> trainRS <- mopaTrain(y = pa_RS, x = biostack$baseline, weighting = TRUE,
                k = 10, algorithm = c("glm", "mars"))
```

The special case of model fitting with TS pseudo-absences

After the generation of TS pseudo-absences, multiple background extents exist as a result of the different distances defined by `backgroundRadius`. It has been noted that the background extent from which pseudo-absences are sampled is an important factor affecting not only model performance, but also its transferability and biological meaning Van der Wal and Shoo (2009). With this regard, Iturbide et al. (2015) propose a selection criterion based on the response of model performance as a function of distance radius, that is generalizable to different SDM characteristics and spatial scales. With this regard, the performance criterion chosen is the Area Under the ROC Curve (AUC), one of the most widely used accuracy measures of binary classification systems (Swets, 1988). Essentially, the method performs a non-linear regression of the AUC obtained by each SDM extent against their background radius, considering three possible asymptotic models (Fig. 2):

OCSVMprofiling	backgroundRadius	**Method**
×	×	No restriction (RS method)
✓	×	Environmental restriction (RSEP method)
✓	✓	Environmental and spatial restriction (TS method)
×	✓	Spatial restriction (Particular case of RS)

Table 2: Combinations of functions `OCSVMprofiling` and `backgroundRadius` for background definition. These are used prior to pseudo-absence data generation with function `pseudoAbsences`, that controls the different sampling methods.

SDM technique	`algorithm` value	**pkg**`::function`	**Reference**
Generalized Linear Model	`"glm"`	**stats**`::glm`	Part of R
Random Forest	`"rf"`	**ranger**`::ranger`	Wright and Ziegler (2017)
Multivariate Adaptive Regression Splines	`"mars"`	**earth**`::earth`	Milborrow (2017)
Maximum Entropy	`"maxent"`	**dismo**`::maxent`	Hijmans et al. (2017)
Support Vector Machine	`"svm"`	**e1071**`::best.svm`	Meyer et al. (2017)
Classification and regression tree (tree)	`"cart.tree"`	**tree**`::tree`	Ripley (2016)
Classification and regression tree (rpart)	`"cart.rpart"`	**rpart**`::rpart`	Therneau et al. (2017)

Table 3: SDM techniques available in **mopa** through the function `mopaTrain`. The corresponding `algorithm` argument values are also indicated.

1. Michaelis-Menten model: $v(x) = \frac{ax}{Km + x}$
2. 2-parameter exponential model: $v(x) = a(1 - e^{-bx})$
3. 3-parameter exponential model: $v(x) = a - be^{-cx}$

, where v and x represent the AUC and the background extent respectively. a is the asymptotic AUC value achieved by the system and $a - b$ is the intercept. Km is the *Michaelis constant* (i.e. the extent at which the AUC is half of a, and c is the coefficient of the point where the curve is most pronounced. The asymptotic model that better fits the AUC response to the different background extents is automatically selected to extract the AUC asymptotical value. The minimum extent at which the AUC lies above the asymptote is retained as the optimal threshold radius, being the corresponding fitted SDM returned. The asymptotic models are fitted internally by `mopaTrain` via the `nls` function from package **stats** always the TS method is used (this is automatically detected by the function). Optionally, a diagram displaying the results is also returned by setting the argument `diagrams=TRUE` (Fig. 2).

```
> # Train TS model and generate Fig. 2
> trainTS <- mopaTrain(y = pa_TS, x = biostack$baseline, weighting = TRUE,
                k = 10, algorithm = c("glm", "mars"), diagrams = TRUE)
```

Model assessment

The object returned by `mopaTrain` is a list of several components generated in the model calibration and evaluation process. Several performance measures are included apart from the AUC, like the True Skill Statistic (TSS) and Cohen's Kappa obtained in the cross-validation, frequently use for the assessment in SDMs (Allouche et al., 2006). These and other ocmponents of the SDM fitted object can be accessed using `extractFromModel`. For, instance, to extract the TSS:

```
> tss.RS <- extractFromModel(models = trainRS, value = "tss")
```

However, and for maximum user flexibility, a matrix containing the observed and predicted probability values for each calibration point is returned, allowing other types of user-tailored model performance assessments.

```
> ObsPred.RS <- extractFromModel(models = trainRS, value = "ObsPred")
```

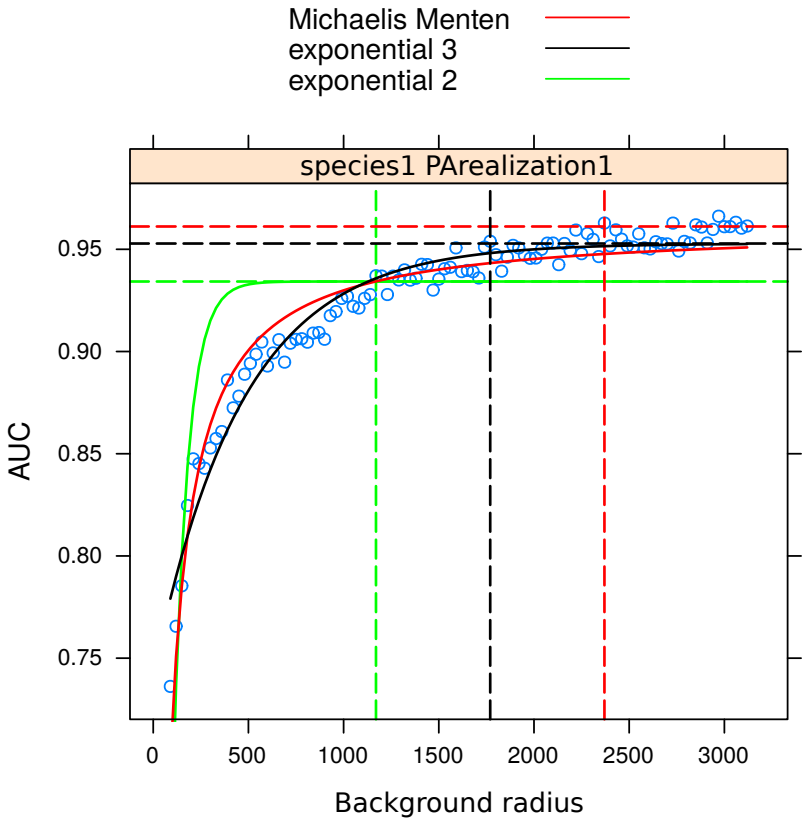


Figure 2: Asymptotic model fitting in SDMs using the TS approach for pseudo-absence generation. The blue points are the AUC values (y-axis) obtained by the SDMs for different background radius extents (x axis). Non-linear fits to the three asymptotic models considered (Michaelis Menten, 2 and 3-parameter exponential). The vertical and horizontal lines indicate the optimal radius and resulting AUC value of the final `mopaTrain` SDM output.

The fitted models are stored in the `"model"` (or `"fold.models"`) component, required for subsequent model prediction.

```
> models.RS <- extractFromModel(models = trainRS, value = "model")
```

Additionally, variable importance may be also estimated. One straightforward possibility is to pass the fitted models to function `varImp` from package **caret** (Kuhn, 2017).

Model predictions

SDM predictions are obtained by passing a new set of predictors (e.g.: future bioclimatic variables) to the generated models. The `model` component corresponds to the models fitted using all available data for model training, while the SDM predictions for the k-cross-validation setup are generated from the component `fold.models` –instead of `model`–. Thus, **mopa** allows handling both the cross-fitted models for flexible model performance assessment and the global model –fitted with all presences and pseudo-absences– for predicting distributions, accomplished through the use of the function `mopaPredict`. In the following example, models corresponding to the RS method are projected to reference climate conditions (`biostack$baseline`) and to 7 future climate projections (`biostack$future`):

```
> ensemble.present <- mopaPredict(models = models.RS,
                newClim = biostack$baseline)
> ensemble.future <- mopaPredict(models = models.RS,
                newClim = biostack$future)
```

Exploring the uncertainty in SDM projections

Projections returned by `mopaPredict` are structured in a nested `list`. Each depth or level in the `list` corresponds to a different component. These are: presence data sets (`SP`), pseudo-absence realizations (`PA`), modeling algorithms (`SDM`), baseline climate (`baseClim`), and the new climate (`newClim`)

used to project models (e.g. future climate projections). The function used to extract components is `extractFromPrediction`. In the next example, projections corresponding to the first pseudo-absence realization (object `rcms_run1`) and to the future climate projection from the MPI RCM (object `runs_rcm1`) are extracted:

```
> rcms_run1 <- extractFromPrediction(ensemble.future, "PA01")
> runs_rcm1 <- extractFromPrediction(ensemble.future, "MPI")
```

Then, the function is again applied to object `runs_rcm1` to extract the SDM results for MPI and GLM. The resulting object is of S4-class `raster*`, thus being straightforward to apply any of the plotting/analysis methods for spatial objects. Here, we use `spplot` from **sp** for output visualization (Fig. 3).

```
> glm_runs_rcm1 <- extractFromPrediction(runs_rcm1, "glm")
> # Generates Fig. 3
> data(wrld)
> spplot(glm_runs_rcm1, layout = c(5, 2), at = seq(0, 1, 0.1),
                col.regions = colorRampPalette(c("white",  "red3")),
                sp.layout= list(wrld, first = FALSE, lwd = 0.5))
```

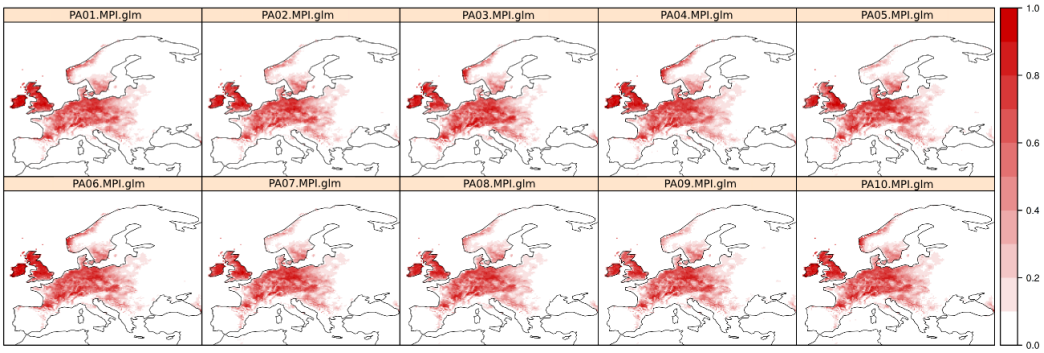


Figure 3: Future species distribution projections (2071-2100) according to the MPI RCM projections, considering 10 different pseudo-absence realizations of the RS method, as stored in the object `glm_runs_rcm1`.

Thus, it is easy to explore the results by inspecting the different components of the `mopaPredict` outputs. For instance, the **raster** package can be particularly useful this aim allowing for a wide variety of map algebra operations through the function `stackApply` over user-defined subsets of SDM projections.

Partition of the uncertainty into components using ANOVA

The relative contribution of each component to the total ensemble spread/variability is implemented in **mopa** using an ANOVA approach, through the function `varianceAnalysis`, following the method in Déqué et al. (2012), also applied by San-Martín et al. (2016). For instance, in this example, the total variance V can be decomposed as the summation of the variance explained by the pseudo-absence realization P, the RCM R and the combination of both PR, so $V = P + R + PR$. Let i be the index of the pseudo-absence realization ($i = 1, \ldots, 10$), j the index of the RCM ($j = 1, \ldots, 7$), and X_{ij} is the response (e.g.: the predicted distribution for the particular realization and climate projection). Then:

$$P = \frac{1}{10}\sum_{i=1}^{10}(X_i - \bar{X})^2 \quad \text{and} \quad R = \frac{1}{7}\sum_{j=1}^{7}(X_j - \bar{X})^2 \tag{1}$$

are the terms resulting from the realization alone (P), and RCM alone (R), and

$$PR = \frac{1}{10}\sum_{i=1}^{10}\frac{1}{7}\sum_{j=1}^{7}(X_{ij} - X_i - X_j + \bar{X})^2 \tag{2}$$

is the interaction term of the realization with the RCM (PR). The following example shows the analysis performed for the pseudo-absence realizations (`component1 = "PA"`) and the climate projections (`component2 = "newClim"`) in GLM projections (`fixed = "glm"`). In order to illustrate thoroughgoing

information on the spread in the projected potential distributions, variance percentage maps are returned together with the maps of the mean and standard deviation. Again, the results can be conveniently visualized with function `spplot` (Figs. 4 and 5).

```
> var.glm <- varianceAnalysis(predictions = ensemble.future,
                component1 = "PA", component2 = "newClim", fixed = c("glm"))
> # Generates Fig. 4
> spplot(var.glm$mean,
                at = seq(0,1,0.1),
                col.regions = colorRampPalette(c("white", "red3")),
                sp.layout= list(wrld, first = FALSE, lwd = 0.5))
> # Generates Fig. 5
> spplot(var.glm$variance,
                col.regions = rev(gray.colors(10, end = 1)),
                at = seq(0, 100, 10),
                sp.layout= list(wrld, first = FALSE, lwd = 0.5))
```

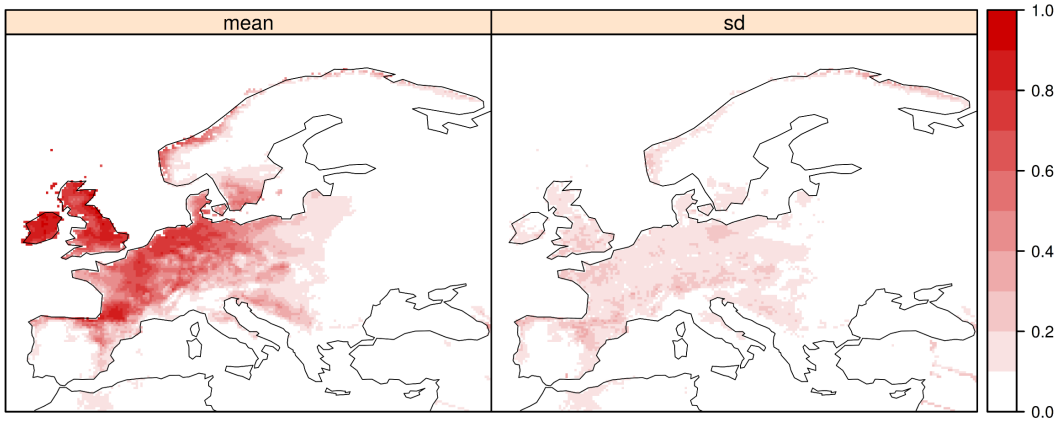


Figure 4: Mean and standard deviation of the SDM ensemble projections (GLM), formed by 7 RCMs × 10 pseudo-absence realizations (RS method, object `var.glm$mean`).

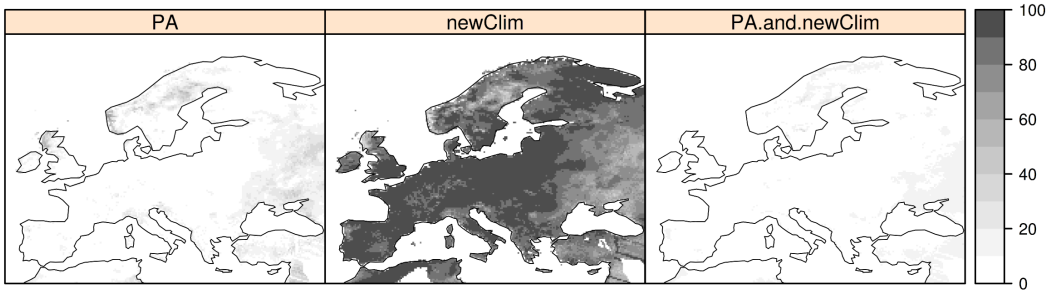


Figure 5: Variance percentage explained by each component: pseudo-absence realization (*PA*), RCM future climate projections (*newClim*) and their joint contribution (*PA.and.newClim*), considering GLM projections (object `var.glm$var`).

Figures 4 and 5 depict the ensemble SDM projections and the variance analysis results, applied to the set of projections that correspond to the 10 pseudo-absence realization and 7 climate projections (10 realizations x 7 RCMs). The mean suitability map and the standard deviation are shown in Figure 4, while Figure 5 are the variance fraction maps (%), depicting the contribution of each component (realization, RCM and realization & RCM) to the overall variance. For instance, the results displayed in Figure 5 unveil that the RCM choice (component `newClim`) is by far the most important factor contributing to the ensemble spread, while pseudo-absence realization has some impact in areas that are outside the current domain of the Oak phylogeny H1 (e.g. Scandinavia).

Similarly, the next lines perform the same analysis, but considering MARS instead of GLM as the statistical modeling technique (Figs. 6 and 7):

```
> var.mars <- varianceAnalysis(predictions = ensemble.future,
                component1 = "PA", component2 = "newClim", fixed = c("mars"))
```

```
> # Generates Fig. 6
>  spplot(var.mars$mean,
                at = seq(0,1,0.1),
                col.regions = colorRampPalette(c("white", "red3")),
                sp.layout= list(wrld, first = FALSE, lwd = 0.5))
> # Generates Fig. 7
> spplot(var.mars$variance,
                at = seq(0, 100, 10),
                col.regions = rev(gray.colors(10, end = 1)),
                sp.layout= list(wrld, first = FALSE, lwd = 0.5))
```

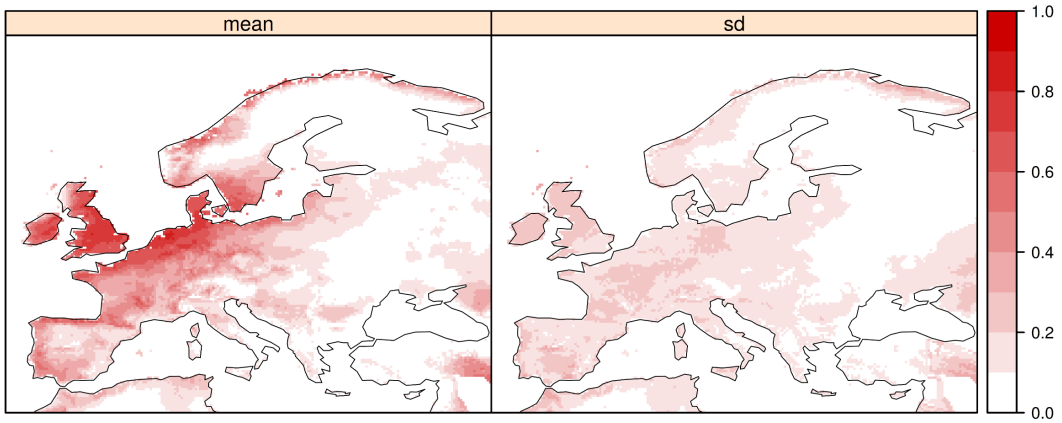


Figure 6: Same as Fig. 4, but considering MARS instead of GLM as statistical modeling technique for SDM production (object `var.mars$mean`).

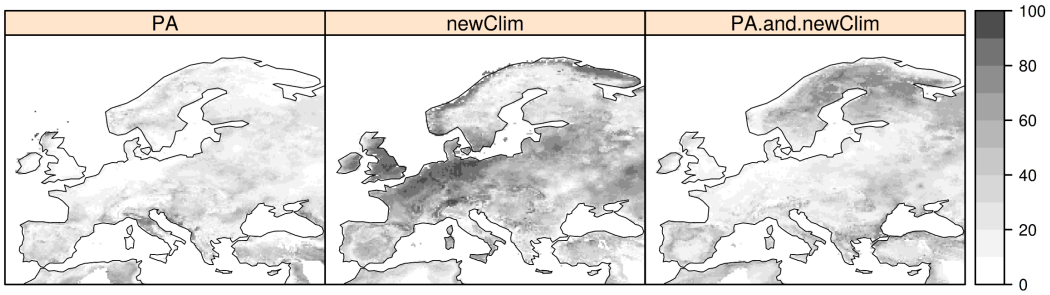


Figure 7: Same as Fig. 5, but considering MARS instead of GLM as the statistical modeling technique for SDM production (object `var.mars$var`).

Unlike GLM, in the case of MARS the ensemble spread (Fig. 6) is greatly affected by the pseudo-absence realization in a wide area of the study domain (Fig. 7), specially in peripheral regions. This is unequivocally diagnosed after applying function `varianceSummary`, which provides a summary of the results, including a graph (Fig. 8) and allowing the comparison of multiple results for a particular uncertainty component. This summary is based on the spatial subsetting of the study area, by specifying the number of subsets with argument `regions`. The output boxplot (Fig. 8) shows the spatial spread of the results (variance proportion explained by a component and the total standard deviation) in each region.

As a result, in Figure 8, we compare GLM and MARS (`var.glm` and `var.mars`) with regard to the variance proportion explained by the RCM choice (`component = 2L`), so that the percentage not explained by it, is associated to the pseudo–absence realization. From this summary, we can confirm a significantly higher sensitivity of MARS to the pseudo–absence sample across all regions.

```
> # Generates Fig. 8
> varianceSummary("glm" = var.glm, "mars" = var.mars,
                component = 2L, regions = c(6, 6), drawBoxplot = TRUE)
```

Alternatively, a `SpatialPolygons` object (package **sp**) can be passed to argument `regions` in order to focus the analysis on specific areas of interest. For illustrative purposes, in this example we use the

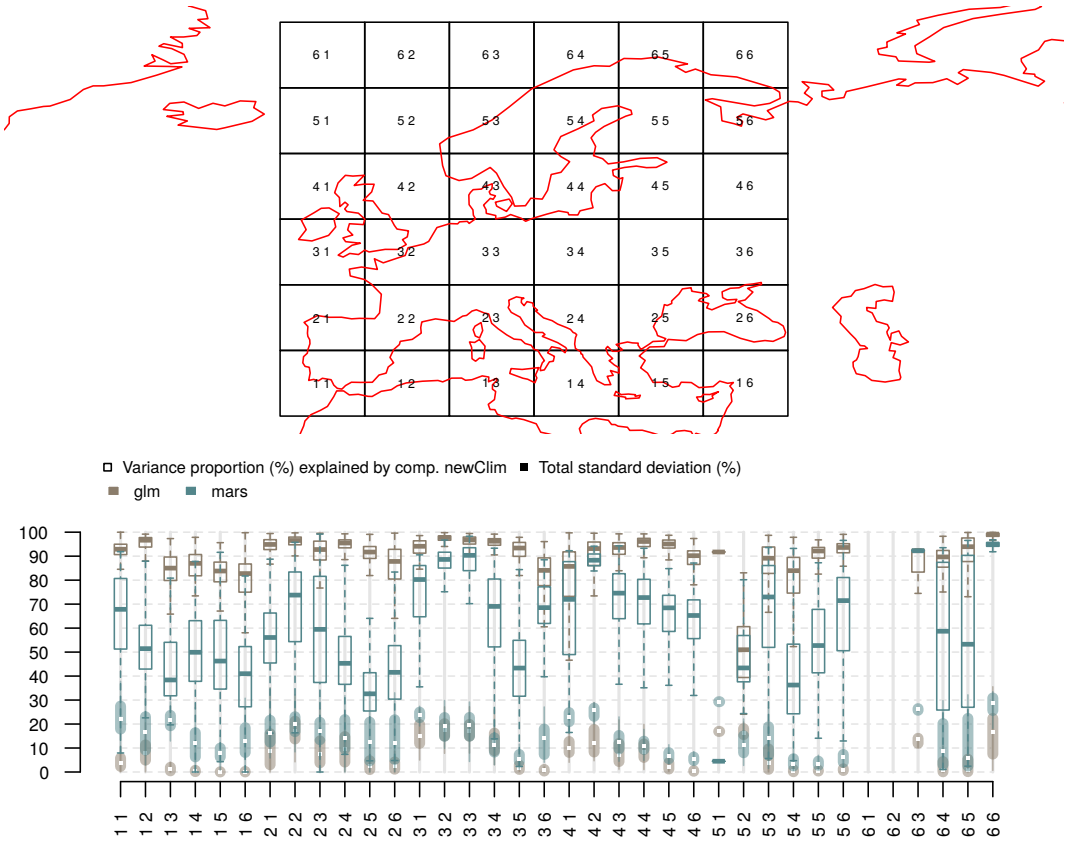


Figure 8: Summary of the variance analysis results generated with function `varianceSummary`, where GLM and MARS techniques (brown and blue respectively) are compared. Boxes account for the spatial spread of the results in each region. Empty boxes show the variance proportion explained by the RCM choice (component `newClim`) and filled boxes show the overall spread, this is, the standard deviation of the predicted probability expressed as a percentage. Thus, empty boxes show how is the total spread (filled boxes) distributed between components (`PA` and `newClim`). The x-axis corresponds to the regions shown in the map at the top.

climatic regions defined in the EU-funded PRUDENCE project (Christensen and Christensen, 2007), which is available at the climate4R package **visualizeR** (Frías et al., 2018):

```
> regiondir <- tempfile()
> download.file(paste0("https://github.com/SantanderMetGroup/",
+   "visualizeR/raw/devel/data/PRUDENCEregions.rda"), destfile = regiondir)
> load(regiondir)
> varianceSummary("glm" = var.glm, "mars" = var.mars,
                component = 2L, drawBoxplot = FALSE, regions = PRUDENCEregions)
```

Additionally, if argument `drawBoxplot` is set as `FALSE`, a simpler graph is obtained displaying the points of the spatial mean. This might be useful when multiple results are being compared in the same graph.

The much higher sensitivity of MARS to the pseudo–absence sample warns about its instability, while GLM reveals much better properties in terms of model stability and transferability. These findings are possible after ANOVA analysis thanks to the utilities included in **mopa**, enabling a flexible experimental setup with a simple user interface. Model transferability is thus not apparent during the SDM calibration stage and is not coupled to model performance (even with the application of the 10-fold cross validation approach), so for instance TSS among realizations was 0.82 for GLM and 0.85 for MARS, and the mean AUC, 0.91 and 0.92 respectively. The uncertainty analysis results are extremely valuable for the construction of an ensemble of SDM projections that minimizes the risk of including unuseful realizations, thus yielding more plausible results.

In the same vein, the contribution of pseudo–absences in front SDM techniques to the overall spread is achieved by adding a new component argument to `varianceAnalysis`, while the RCM projection (MPI in this example) is kept as a fixed factor:

```
> MPI.var <- varianceAnalysis(ensemble.future,
                component1 = "PA", component2 = "SDM", fixed = c("MPI"))
```

In case further uncertainty components are considered for predicting distributions (named in **mopa** as `SP`, `baseClim` and `foldModels`), these could also be analyzed by keeping several fixed factors, each corresponding to a component that is not being analyzed. This is explained in detail in the help document of function "varianceAnalysis".

```
> help(varianceAnalysis)
```

SDM ensemble building

Finally, the ensemble forecast is built. In this particular example, we could discard those MARS projections that we consider are the result of bad transferability, e.g. corresponding to the pseudo-absence realizations that resulted in unrealistic predictions. Let us consider the simplified case where, after a more detailed analysis of the results, we conclude that MARS projections corresponding to pseudo–absence realization 8 along with GLM projections, are valid forecasts, then, as shown in the next example, the definitive ensemble is easily built with function `extractFromPrediction` and the utilities of the **raster** package. Here we calculate and plot the ensemble mean and standard deviation of the final SDM ensemble projections (Fig. 9):

```
> marsEns <- extractFromPrediction(ensemble.future, value = "mars")
> marsEnsPA08 <- extractFromPrediction(marsEns, value = "PA08")
> glmEns <- extractFromPrediction(ensemble.future, value = "glm")

> ensemble.future.def <-  stack(list(glmEns, marsEnsPA08))
> mean.ensemble <- stackApply(ensemble.future.def, fun = mean,
                indices = rep(1, nlayers(ensemble.future.def)))
> sd.ensemble <- stackApply(ensemble.future.def, fun = sd,
                indices = rep(1, nlayers(ensemble.future.def)))
> forecast.future <- stack(mean.ensemble, sd.ensemble)
> names(forecast.future) <- c("ensemble mean", "ensemble sd")
> # Generates Fig. 9
> spplot(forecast.future, at = seq(0,1,0.1),
                col.regions = colorRampPalette(c("white", "red3")),
                sp.layout= list(wrld, first = FALSE, lwd = 0.5))
```

Basically, this is a weighting exercise that favors GLM predictions in front of those of MARS, beyond the performance shown in the calibration phase.

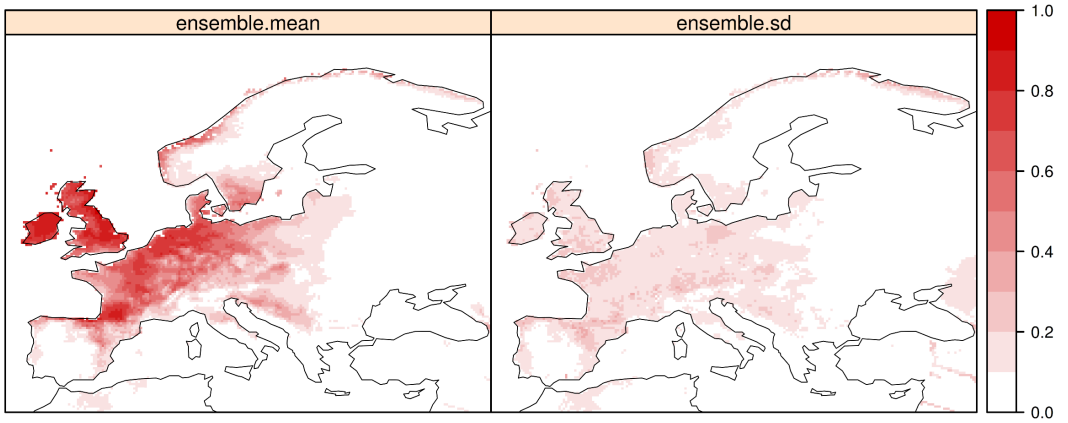


Figure 9: Future ensemble forecast (mean and standard deviation) of the suitability of the oak phylogeny H11 under climate conditions given by 7 different RCMs.

Summary

The impacts of climate change on the biological systems are of current concern worldwide, and future SDMs have become a key tool for the vulnerability and impact assessment community. Thus, the utilities in package **mopa** can help in the SDM production chain since the early stage (climate data retrieval and post-processing) to the ultimate phase in which a final set of models is retained for ensemble generation and map production.

In this case-study, we illustrate the development of a set of SDM projections considering multiple combinations of climate change projections from a set of state-of-the-art RCMs, two popular statistical modeling methods (GLM and MARS) and different pseudo-absence realizations, enabling the identification of those members of the ensemble yielding consistent and plausible future estimates for final SDM building. With this regard, the ability to quantitatively assess the individual contribution of each factor to the overall SDM spread, as implemented in function `varianceAnalysis` proved crucial in the evaluation. While previously existing R packages already provide functionality for SDM building and their assessment during the calibration stage, we have shown that model performance, as evaluated by ordinary cross-validation, is not coupled to model transferabilty into future climate, being therefore this essential feature specific of **mopa**. Other characteristic aspects introduced by the package consist of the novel methods for pseudo-absence generation, and the ability to perform a fine-tuning of these methods prior to model fitting. Furthermore, the inter-operability of **mopa** with other SDM-related R packages enables maximum flexibility and eases the use of R for SDM applications in the framework of complex modeling exercises, for which multiple aspects have a varying contribution to the overall uncertainty.

Acknowledgements

We are grateful to the one anonymous referee for her/his valuable comments. We acknowledge the ENSEMBLES project (GOCE-CT-2003-505539), supported by the European Commission's 6th Framework Program for providing publicly the RCM simulations and observational data used in this study. We are also grateful to Rémy Petit and François Ehrenmann for providing the distribution of Oak phylogenies.

Bibliography

O. Allouche, A. Tsoar, and R. Kadmon. Assessing the accuracy of species distribution models: Prevalence, kappa and the true skill statistic (TSS). *Journal of Applied Ecology*, 43:1223–1232, 2006. URL https://doi.org/10.1111/j.1365-2664.2006.01214.x. [p128]

M. B. Araújo and M. New. Ensemble forecasting of species distributions. *Trends in Ecology & Evolution*, 22:42–47, 2007. URL https://doi.org/10.1016/j.tree.2006.09.010. [p122]

M. B. Araújo, M. Cabeza, W. Thuiller, L. Hannah, and P. H. Williams. Would climate change drive species out of reserves? an assessment of existing reserve-selection methods. *Global Change Biology*, 10:1618–1626, 2004. URL https://doi.org/10.1111/j.1365-2486.2004.00828.x. [p122]

M. B. Araújo, R. J. Whittaker, R. J. Ladle, and M. Erhard. Reducing uncertainty in projections of extinction risk from climate change. *Global Ecology and Biogeography*, 14:529–538, 2005. URL https://doi.org/10.1111/j.1466-822x.2005.00182.x. [p122]

R. Bagchi, M. Crosby, B. Huntley, D. G. Hole, S. H. M. Butchart, Y. Collingham, M. Kalra, J. Rajkumar, A. Rahmani, M. Pandey, H. Gurung, L. T. Trai, N. Van Quang, and S. G. Willis. Evaluating the effectiveness of conservation site networks under climate change: Accounting for uncertainty. *Global Change Biology*, 19:1236–1248, 2013. URL https://doi.org/10.1111/gcb.12123. [p122]

D. J. Baker, A. J. Hartley, S. H. M. Butchart, and S. G. Willis. Choice of baseline climate data impacts projected species' responses to climate change. *Global Change Biology*, 22:991–1003, 2016. URL https://doi.org/10.1111/gcb.13273. [p122]

M. Barbet-Massin, F. Jiguet, C. H. Albert, and W. Thuiller. Selecting pseudo-absences for species distribution models: How, where and how many? *Methods in Ecology and Evolution*, 3:327–338, 2012. URL https://doi.org/10.1111/j.2041-210x.2011.00172.x. [p127]

L. J. Beaumont, L. Hughes, and A. J. Pitman. Why is the choice of future climate scenarios for species distribution modelling important? *Ecology Letters*, 11:1135–1146, 2008. URL https://doi.org/10.1111/j.1461-0248.2008.01231.x. [p122]

J. Bedia, J. Busqué, and J. M. Gutiérrez. Predicting plant species distribution across an alpine rangeland in northern spain: a comparison of probabilistic methods. *Applied Vegetation Science*, 14:415–432, 2011. URL https://doi.org/10.1111/j.1654-109x.2011.01128.x. [p122]

J. Bedia, S. Herrera, and J. M. Gutiérrez. Dangers of using global bioclimatic datasets for ecological niche modeling. limitations for future climate projections. *Global and Planetary Change*, 107:1–12, 2013. URL https://doi.org/10.1016/j.gloplacha.2013.04.005. [p122]

J. Bedia, N. Golding, A. Casanueva, M. Iturbide, C. Buontempo, and J. M. Gutiérrez. Seasonal predictions of Fire Weather Index: Paving the way for their operational applicability in Mediterranean Europe. *Climate Services*, 2017. URL https://doi.org/10.1016/j.cliser.2017.04.001. [p123]

C. Beierkuhnlein, D. Thiel, A. Jentsch, E. Willner, and J. Kreyling. Ecotypes of european grass species respond differently to warming and extreme drought. *Journal of Ecology*, 99:703–713, 2011. URL https://doi.org/10.1111/j.1365-2745.2011.01809.x. [p124]

R. S. Bivand, E. Pebesma, and V. Gomez-Rubio. *Applied Spatial Data Analysis with R, Second Edition*. Springer-Verlag, 2013. URL http://www.asdar-book.org/. [p123]

L. Buisson, W. Thuiller, N. Casajus, S. Lek, and G. Grenouillet. Uncertainty in ensemble forecasting of species distribution. *Global Change Biology*, 16:1145–1157, 2010. URL https://doi.org/10.1111/j.1365-2486.2009.02000.x. [p122]

J. R. Busby. *Nature Conservation: Cost Effective Biological Surveys and Data Analysis*, chapter BIOCLIM - a bioclimatic analysis and prediction system. CSIRO, 1991. [p122, 123]

J. H. Christensen and O. B. Christensen. A summary of the PRUDENCE model projections of changes in European climate by the end of this century. *Climatic Change*, 81(1):7–30, 2007. ISSN 0165-0009, 1573-1480. URL https://doi.org/10.1007/s10584-006-9210-7. [p134]

A. Cofino, J. Bedia, M. Iturbide, M. Vega, S. Herrera, J. Fernández, M. D. Frías, R. Manzanas, and J. M. Gutiérrez. The ECOMS User Data Gateway: Towards Seasonal Forecast Data Provision and Research Reproducibility in the Era of Climate Services. *Climate Services*, in press, 2017. URL https://doi.org/10.1016/j.cliser.2017.07.001. [p123]

M. Déqué, S. Somot, E. Sanchez-Gomez, C. M. Goodess, D. Jacob, G. Lenderink, and O. B. Christensen. The spread amongst ENSEMBLES regional scenarios: Regional climate models, driving general circulation models and interannual variability. *Climate Dynamics*, 38:951–964, 2012. URL https://doi.org/10.1007/s00382-011-1053-x. [p130]

J. Elith and et al. Novel methods improve prediction of species' distributions from occurrence data. *Ecography*, 29:129–151, 2006. URL https://doi.org/10.1111/j.2006.0906-7590.04596.x. [p122]

P. Falloon, A. Challinor, S. Dessai, L. Hoang, J. Johnson, and A.-K. Koehler. Ensembles and uncertainty in climate change impacts. *Frontiers in Environmental Science*, 2:33, 2014. URL https://doi.org/10.3389/fenvs.2014.00033. [p122]

J. H. Friedman. Multivariate adaptive regression splines. *The Annals of Statistics*, 19:1–67, 1991. [p127]

S. Fronzek, T. R. Carter, and M. Luoto. Evaluating sources of uncertainty in modelling the impact of probabilistic climate change on sub-arctic palsa mires. *Natural Hazards and Earth System Sciences*, 11: 2981–2995, 2011. URL https://doi.org/10.5194/nhess-11-2981-2011. [p122]

M. D. Frías, M. Iturbide, R. Manzanas, J. Bedia, J. Fernández, S. Herrera, A. S. Cofiño, and J. M. Gutiérrez. An R package to visualize and communicate uncertainty in seasonal climate prediction. *Environmental Modelling & Software*, 99(Supplement C):101–110, 2018. ISSN 1364-8152. URL https://doi.org/10.1016/j.envsoft.2017.09.008. [p123, 134]

A. Guisan and N. E. Zimmermann. Predictive habitat distribution models in ecology. *Ecological modelling*, 135:147–186, 2000. URL https://doi.org/10.1016/s0304-3800(00)00354-9. [p122]

A. Guisan, T. C. Edwards, and T. Hastie. Generalized linear and generalized additive models in studies of species distributions: Setting the scene. *Ecological Modelling*, 157:89–100, 2002. URL https://doi.org/10.1016/s0304-3800(02)00204-1. [p127]

A. Hamann and T. Wang. Potential effects of climate change on ecosystem and tree species distribution in british columbia. *Ecology*, 87:2773–2786, 2006. URL https://doi.org/10.1890/0012-9658(2006)87[2773:peocco]2.0.co;2. [p122]

M. R. Haylock, N. Hofstra, A. M. G. Klein Tank, E. J. Klok, P. D. Jones, and M. New. A European daily high-resolution gridded data set of surface temperature and precipitation for 1950–2006. *Journal of Geophysical Research*, 113:D20119, 2008. URL https://doi.org/10.1029/2008jd010201. [p124]

P. A. Hernandez, C. H. Graham, L. L. Master, and D. L. Albert. The effect of sample size and species characteristics on performance of different species distribution modeling methods. *Ecography*, 29: 773–785, 2006. URL https://doi.org/10.1111/j.0906-7590.2006.04700.x. [p124]

R. J. Hijmans. *raster: Geographic Data Analysis and Modeling*, 2015. URL https://CRAN.R-project.org/package=raster. R package version 2.4-20. [p123]

R. J. Hijmans, S. Phillips, J. Leathwick, and J. Elith. *dismo: Species Distribution Modeling*, 2017. URL https://CRAN.R-project.org/package=dismo. R package version 1.1-4. [p123, 128]

M. Iturbide, J. Bedia, S. Herrera, O. del Hierro, M. Pinto, and J. M. Gutiérrez. A framework for species distribution modelling with improved pseudo-absence generation. *Ecological Modelling*, 312:166–174, 2015. URL https://doi.org/10.1016/j.ecolmodel.2015.05.018. [p123, 124, 125, 126, 127]

M. Iturbide, J. Bedia, and J. M. Gutiérrez. Background sampling and transferability of species distribution model ensembles under climate change. *Global and Planetary Change*, 2018. URL https://doi.org/10.1016/j.gloplacha.2018.03.008. In press. [p123]

J. M. Jeschke and D. L. Strayer. Usefulness of bioclimatic models for studying climate change and invasive species. In *Year in Ecology and Conservation Biology*, volume 1134 of *Annals of the New York Academy of Sciences*, pages 1–24. Blackwell Publishing, 9600 Garsington RD, Oxford OX4 2DQ, Oxen, England, 2008. [p122]

M. Kuhn. *caret: Classification and Regression Training*, 2017. URL https://CRAN.R-project.org/package=caret. R package version 6.0-76. [p129]

R. G. Mateo, Ángel M. Felicísimo, and J. Muñoz. Effects of the number of presences on reliability and stability of MARS species distribution models: The importance of regional niche variation and ecological heterogeneity. *Journal of Vegetation Science*, 21:908–922, 2010. URL https://doi.org/10.1111/j.1654-1103.2010.01198.x. [p122, 124]

D. Meyer, E. Dimitriadou, K. Hornik, A. Weingessel, and F. Leisch. *e1071: Misc Functions of the Department of Statistics, Probability Theory Group (Formerly: E1071), TU Wien*, 2017. URL https://CRAN.R-project.org/package=e1071. R package version 1.6-8. [p125, 128]

S. Milborrow. *earth: Multivariate Adaptive Regression Splines*, 2017. URL https://CRAN.R-project.org/package=earth. R package version 4.5.0. [p128]

B. Naimi and M. B. Araújo. sdm: a reproducible and extensible R platform for species distribution modelling. *Ecography*, 39:368–375, 2016. URL https://doi.org/10.1111/ecog.01881. [p123]

H. A. Nix. *Atlas of Elapid Snakes of Australia*, chapter A biogeographic analysis of Australian Elaphid snakes. Australian Government Publishing Service, Canberra, Australia, 1986. [p122]

E. J. Pebesma and R. S. Bivand. *Classes and Methods for Spatial Data in R*, 2005. URL https://cran.r-project.org/doc/Rnews/Rnews_2005-2.pdf. [p123]

A. T. Peterson, J. Soberón, R. G. Pearson, R. P. Anderson, E. Martinez-Meyer, M. Nakamura, and M. B. Araujo. *Ecological Niches and Geographic Distributions*. Monographs in population biology. Princeton University, 2011. ISBN 978-0-691-13688-2. [p122]

R. J. Petit, U. M. Csaikl, S. Bordács, K. Burg, E. Coart, J. Cottrell, B. van Dam, J. D. Deans, S. Dumolin-Lapègue, S. Fineschi, R. Finkeldey, A. Gillies, I. Glaz, P. G. Goicoechea, J. S. Jensen, A. O. König, A. J. Lowe, S. F. Madsen, G. Mátyás, R. C. Munro, M. Olalde, M.-H. Pemonge, F. Popescu, D. Slade, H. Tabbener, D. Taurchini, S. G. M. de Vries, B. Ziegenhagen, and A. Kremer. Chloroplast DNA variation in European white oaks: Phylogeography and patterns of diversity based on data from over 2600 populations. *Forest Ecology and Management*, 156:5–26, 2002. URL https://doi.org/10.1016/s0378-1127(01)00645-4. [p124]

C. F. Randin, T. Dirnböck, S. Dullinger, N. E. Zimmermann, M. Zappa, and A. Guisan. Are niche-based species distribution models transferable in space? *Journal of Biogeography*, 33:1689–1703, 2006. URL https://doi.org/10.1111/j.1365-2699.2006.01466.x. [p124]

B. Ripley. *tree: Classification and Regression Trees*, 2016. URL https://CRAN.R-project.org/package=tree. R package version 1.0-37. [p128]

D. San-Martín, R. Manzanas, S. Brands, S. Herrera, and J. M. Gutiérrez. Reassessing Model Uncertainty for Regional Projections of Precipitation with an Ensemble of Statistical Downscaling Methods. *Journal of Climate*, 30:203–223, 2016. URL https://doi.org/10.1175/jcli-d-16-0366.1. [p130]

S. D. Senay, S. P. Worner, and T. Ikeda. Novel three-step pseudo-absence selection technique for improved species distribution modelling. *PLoS ONE*, 8:e71218, 2013. URL https://doi.org/10.1371/journal.pone.0071218. [p123, 125]

M. J. Serra-Varela, D. Grivet, L. Vincenot, O. Broennimann, J. Gonzalo-Jiménez, and N. E. Zimmermann. Does phylogeographical structure relate to climatic niche divergence? a test using maritime pine (Pinus pinaster Ait.). *Global Ecology and Biogeography*, 24:1302–1313, 2015. URL https://doi.org/10.1111/geb.12369. [p124]

J. Swets. Measuring the accuracy of diagnostic systems. *Science*, 240:1285–1293, 1988. URL https://doi.org/10.1126/science.3287615. [p127]

K. E. Taylor, R. J. Stouffer, and G. A. Meehl. An Overview of CMIP5 and the Experiment Design. *Bulletin of the American Meteorological Society*, 93(4):485–498, 2011. URL https://doi.org/10.1175/bams-d-11-00094.1. [p122]

T. Therneau, B. Atkinson, and B. Ripley. *rpart: Recursive Partitioning and Regression Trees*, 2017. URL https://CRAN.R-project.org/package=rpart. R package version 4.1-11. [p128]

W. Thuiller, M. B. Araújo, R. G. Pearson, R. J. Whittaker, L. Brotons, and S. Lavorel. Biodiversity conservation: Uncertainty in predictions of extinction risk. *Nature*, 430:145–148, 2004. URL https://doi.org/10.1038/nature02716. [p122]

W. Thuiller, D. Georges, R. Engler, and F. Breiner. *biomod2: Ensemble Platform for Species Distribution Modeling*, 2016. URL https://CRAN.R-project.org/package=biomod2. R package version 3.3-7. [p123]

M. Turco, A. Sanna, S. Herrera, M.-C. Llasat, and J. M. Gutiérrez. Large biases and inconsistent climate change signals in ENSEMBLES regional projections. *Climatic Change*, 120:859–869, 2013. URL https://doi.org/10.1007/s10584-013-0844-y. [p122]

P. van der Linden and J. F. B. Mitchell. ENSEMBLES: Climate Change and its Impacts: Summary of research and results from the ENSEMBLES project — European Environment Agency (EEA). Technical report, Met Office Hadley Centre, FitzRoy Road, Exeter EX1 3PB, UK., 2009. [p124]

J. Van der Wal and L. P. Shoo. Selecting pseudo-absence data for presence-only distribution modeling: How far should you stray from what you know? *Ecological Modelling*, 220:589–594, 2009. URL https://doi.org/10.1016/j.ecolmodel.2008.11.010. [p127]

J. Van der Wal, L. Falconi, S. Januchowski, L. Shoo, and C. Storlie. *SDMTools: Species Distribution Modelling Tools: Tools for Processing Data Associated with Species Distribution Modelling Exercises*, 2014. URL https://CRAN.R-project.org/package=SDMTools. R package version 1.1-221. [p123]

D. L. Verbyla and J. A. Litvaitis. Resampling methods for evaluating classification accuracy of wildlife habitat models. *Environmental Management*, 13:783–787, 1989. URL https://doi.org/10.1007/bf01868317. [p127]

J. A. Winkler, J. P. Palutikof, J. A. Andresen, and C. M. Goodess. The Simulation of Daily Temperature Time Series from GCM Output. Part II: Sensitivity Analysis of an Empirical Transfer Function Methodology. *Journal of Climate*, 10:2514–2532, 1997. URL https://doi.org/10.1175/1520-0442(1997)010<2514:tsodtt>2.0.co;2. [p124]

M. S. Wisz and A. Guisan. Do pseudo-absence selection strategies influence species distribution models and their predictions? an information-theoretic approach based on simulated data. *BMC Ecology*, 9:8, 2009. URL https://doi.org/10.1186/1472-6785-9-8. [p125]

M. N. Wright and A. Ziegler. ranger: A fast implementation of random forests for high dimensional data in C++ and R. *Journal of Statistical Software*, 77:1–17, 2017. URL https://doi.org/10.18637/jss.v077.i01. [p128]

M. Zahn and H. von Storch. Decreased frequency of North Atlantic polar lows associated with future climate warming. *Nature*, 467:309–312, 2010. URL https://doi.org/10.1038/nature09388. [p124]

M. Iturbide, J.M. Gutiérrez
https://orcid.org/0000-0002-5048-0941
https://orcid.org/0000-0002-2766-6297
Meteorology group. Insituto de Física de Cantabria (IFCA)
CSIC - Universidad de Cantabria. Avda. de los Castros, s/n
39005. Santander. Spain
miturbide@ifca.unican.es

J. Bedia
https://orcid.org/0000-0001-6219-4312
Predictia Intelligent Data Solutions S.L.
http://www.predictia.es/en
Avda. los Castros s/n. Edificio I+D+i. S345
39005. Santander. Spain
Dept. of Applied Mathematics and Computing Science
Universidad de Cantabria. Avda. de los Castros, 44
39005. Santander. Spain

FHDI: An R Package for Fractional Hot Deck Imputation

by Jongho Im, In Ho Cho, and Jae Kwang Kim

Abstract Fractional hot deck imputation (FHDI), proposed by Kalton and Kish (1984) and investigated by Kim and Fuller (2004), is a tool for handling item nonresponse in survey sampling. In FHDI, each missing item is filled with multiple observed values yielding a single completed data set for subsequent analyses. An R package **FHDI** is developed to perform FHDI and also the fully efficient fractional imputation (FEFI) method of (Fuller and Kim, 2005) to impute multivariate missing data with arbitrary missing patterns. FHDI substitutes missing items with a few observed values jointly obtained from a set of donors whereas the FEFI uses all the possible donors. This paper introduces **FHDI** as a tool for implementing the multivariate version of fractional hot deck imputation discussed in Im et al. (2015) as well as FEFI. For variance estimation of FHDI and FEFI, the Jackknife method is implemented, and replicated weights are provided as a part of the output.

Introduction

Incomplete data are common in survey sampling, biomedical, and social sciences. Naive analysis with only complete cases, conducted by removing all the cases with any missing items, is exposed to nonresponse bias unless the missing data mechanism is missing completely at random (Rubin, 1976). Even if the complete cases can be treated as a complete random sample, the complete case analysis is inefficient, as all the partially observed cases are ignored. To incorporate these partial observations, we may consider an imputation technique in which the missing items are filled in with plausible values.

Imputation is often classified into single imputation and repeated imputation on the basis of the number of imputed values per each missing value. Several values are assigned to each missing value in repeated imputation, while a single value is imputed in single imputation. Although single imputation is often preferred in practice, due to its convenience, it does not necessarily preserve the distribution of the original data. Thus, single imputation is inadequate for general purpose estimation.

There are two popular methods in the repeated imputation: multiple imputation and fractional imputation. Multiple imputation, proposed by Rubin (1987, 1996) produces multiply imputed data sets as imputation output. Many multiple imputation methods are already available in R, for example, **mice** (van Buuren and Groothuis-Oudshoorn, 2011), **mi** (Su et al., 2011), **Amelia** (Honaker et al., 2011), and **VIM** (Kowarik and Templ, 2016). Fractional imputation, initially proposed by Kalton and Kish (1984) and extensively discussed in Fay (1996), Kim and Fuller (2004), Durrant and Skinner (2006), and Kim (2011), creates a single completed data set with fractional weights after imputation. The size of the imputed data is always larger than that of the incomplete input data. However, there is no suitable R package for implementing fractional imputation. So far, the only publicly available software for doing fractional imputation is the SAS procedure `SURVEYIMPUTE` (SAS Institue Inc., 2015), which partly covers some fractional imputation methods.

Fractional imputation is yet to be widely used in practice other than survey sampling possibly because it is relatively new and there is more complexity involved in implementing variance estimation compared to multiple imputation. However, fractional imputation has its own advantages. For instance, in the case of using a method-of-moment estimator, fractional imputation provides consistent variance estimation while the multiple imputation variance estimator is inconsistent (Yang and Kim, 2016).

Imputing multivariate missing data is challenging for both multiple imputation and fractional imputation. In practice, a full conditional specification of the joint model can be used to fill in several missing items. One popular method is multiple imputation using chained equations, also named **mice** (van Buuren and Groothuis-Oudshoorn, 2011) as in the name of its R package. This multiple imputation procedure by chained equation (MICE) involves a variable-by-variable approach using chained equations, whereby the imputation model for each missing item is separately specified with the other items as predictors. A linear regression model or predictive mean matching is used for continuous variables, and a logistic regression is used for categorical variables. Once all the conditional distributions are specified, the multiple imputation procedure is repeated until convergence.

However, full conditional specification is subject to model mis-specifications and model compatibility problems (Chen, 2010), and the parameter estimation procedure is sometimes cumbersome. As an alternative, we employ the fractional hot deck imputation (FHDI) proposed by Im et al. (2015), which is a nonparametric imputation approach using a two-phase sampling idea. To apply FHDI in multivariate missing data, Im et al. (2015) first created imputation cells to match donors and recipients

in a nonparametric way, where units with complete data serve as donors and units with at least one missing item serve as recipients. Once the imputation cells are created, multiple donors are assigned to each recipient with the probability proportional to fractional weights, which can be understood as the conditional probability of obtaining the imputed values given the partial observation. After that, the observed values of each donor are jointly imputed to fill the missing parts of the recipient.

The complete R package **FHDI** and entire source codes are available in Im et al. (2018), and can be installed with R 3.4.0 or higher. The package basically includes three main functions with separate purposes. The function `FHDI_CellMake()` converts continuous data into categorical data which can be used as imputation cells. The function `FHDI_CellProb()` is provided to estimate the imputation cell probability using a version of the expectation maximization (EM) algorithm. This function can be used to get the maximum likelihood estimates of the cell probabilities from multivariate, incomplete, categorical data under the missing at random assumption. The function `FHDI_Driver()` performs fractional hot deck imputation, and also provides a set of replicated fractional weights for variance estimation.

FHDI

Fractional hot deck imputation (FHDI), proposed by Kim and Fuller (2004), replaces each missing value with a set of imputed values. Those imputed values are selected at random from values of the donors in the same imputation cell, with the cells constructed to achieve within-cell data homogeneity. Fractional weights are assigned to each imputed value to preserve the original data structure, and a singly imputed data set is obtained as the output of the FHDI.

Im et al. (2015) extended Kim and Fuller (2004)'s idea in two ways. First, the new FHDI does not require imputation cells to be made in advance. Imputation cells are determined as a by-product of the imputation procedure, and are generally created to preserve the most of the correlations among survey items. Second, the new FHDI method is now applied to multivariate missing data with arbitrary missing patterns.

The FHDI of Im et al. (2015) can be understood as an imputation method using two-phase sampling for stratification. In phase one, the imputation cells are determined so that the survey values are homogeneous within cells and each missing unit has at least two possible donors within each imputation cell. The cell probabilities are estimated using the EM by weighting method (Ibrahim, 1990). Once imputation cells are fixed, then the fully efficient fractional imputation (FEFI), named by Fuller and Kim (2005), is implemented by replacing each missing value with all observed values within the imputation cell. The FEFI fractional weights are obtained during this FEFI procedure. However, in general, the size of imputed data from the FEFI procedure can be too large to handle for statistical analysis. To avoid this size issue, we may select $M(\geq 2)$ donors on each missing unit instead of taking all possible donors. In phase two, a set of donors are assigned to each recipient with the probability proportional to their FEFI fractional weights. A vector of missing values is jointly imputed with the observed values of assigned donors.

Basic setup

For a description of the FHDI procedure, suppose that we have a finite population of size N, indexed by $U = \{1, 2, \ldots, N\}$, with two continuous variables y_1 and y_2. Let z_1 and z_2 be discretized values of y_1 and y_2, respectively. Assume that z_1 takes values in $\{1, \ldots, G\}$ and z_2 takes values in $\{1, \ldots, H\}$. Let δ_p $(p = 1, 2)$, a response indicator function of y_p, take a value of one if y_p is observed and zero otherwise. Note that if y_p has a missing value, then z_p is also missing.

The finite population U can be subdivided into $G \times H$ cells based on z_1 and z_2, and we assume a cell mean model on the cells such that

$$y \mid (z_1 = g, z_2 = h) \sim (\mu_{gh}, \Sigma_{gh}), \quad g = 1, \ldots, G, \; h = 1, \ldots, H, \tag{1}$$

where $y = (y_1, y_2)$, $\mu_{gh} = (\mu_{1,gh}, \mu_{2,gh})$ is a vector of cell means and Σ_{gh} is the variance-covariance matrix of y in cell (gh).

Let y_{obs} and y_{mis} be the observed and missing part of y, respectively. We assume that the data are missing at random (MAR) in the sense that

$$P(\delta \mid y) = P(\delta \mid y_{obs}), \tag{2}$$

where $\delta = (\delta_1, \delta_2)$. The MAR condition (2) implies that the imputation model (1) also holds for the respondents.

Let A be the index set of the sample elements selected from the finite population U. Let A_R be the

index set of the respondents who answered both items y_1 and y_2, that is, $A_R = \{j \in A; \delta_{1j}\delta_{2j} = 1\}$. Similarly, define A_M to be the set of the nonrespondents who have at least one missing value. That is, $A_M = \{j \in A; \delta_{1j}\delta_{2j} = 0\}$. Denote n_R and n_M as the size of A_R and A_M, respectively. We assume that A_R is non-empty and there exists enough donors in A_R, that is,

$$n_R \geq n_R^* \tag{3}$$

for some n_R^*. The value of n_R^* is determined by the size of the imputation cells.

Imputation

The FHDI procedure in Im et al. (2015) consists of the following four steps: (i) Cell construction by discretization, (ii) Estimating cell probabilities, (iii) Constructing FEFI fractional weights, and (iv) Imputation. A detailed step-by-step description is provided below.

(Step 1): Cell construction by discretization

We wish to construct imputation cells satisfying (1). The imputation cell variable z can be given in advance, or can be obtained using the estimated sample quantiles. To discuss the latter case, let $\{a_1, \ldots, a_G\}$ be a set of cumulative proportions such that $0 = a_0 < a_1 < \cdots < a_{G-1} < a_G = 1$. We can choose a_i so that each cell contains an equal number of respondents. Let

$$\hat{F}(t) = \frac{\sum_{i \in A} \delta_{1i} w_i I(y_{1i} \leq t)}{\sum_{i \in A} \delta_{1i} w_i} \tag{4}$$

be the estimated distribution function for y_1, where w_i is the sampling weight of unit i; $I(S)$ is an indicator function that takes value of one if S is true and zero otherwise; and $\hat{q}(a_k)$ be the estimated sample quantile corresponding to a_k, defined by $\hat{q}(a_k) = \min\{t; \hat{F}(t) \geq a_k\}$. Once we have the estimated sample quantiles, we can construct z_{1i} from y_{1i}. For instance, $z_{1i} = g$ if $\hat{q}(a_{g-1}) < y_{1i} \leq \hat{q}(a_g)$. If y_{1i} is missing, then z_{1i} has a 'NA' value. Similarly, we can construct z_2 with the range from 1 to H.

From the realized values of z_{1i} and z_{2i} in the sample, we can construct two sets of observed patterns of (z_1, z_2) for A_R and A_M. Let V_R be the set of all observed combinations of z_1 and z_2 in A_R. A size of V_R is $G \times H$ at maximum, but it can be smaller in the realized samples. Similarly we obtain V_M based on the observed parts of nonrespondents. For example, we may have $V_M = \{(\text{NA}, \text{NA}), (\text{NA}, z_2 = 1), (\text{NA}, z_2 = 2), (z_1 = 1, \text{NA}), (z_1 = 2, \text{NA})\}$ in the case of two binary outcomes.

For the proposed FHDI, we need at least two donors for each recipient to capture the variability from imputation. However, an initial discretization may not give enough donors for some recipients. In this case, we apply a cell collapsing procedure to have at least two donors to each recipient. During the cell collapsing, the dimension of the cell variable z can be adjusted to have larger samples within the cells. However, each cell variable z should have at least two distinct values marginally for its validity in the construction of imputation cell. The cell collapsing procedure is designed to be stopped if any cell variable has a single observation. In the left panel of Table 1, we only have one possible donor for recipients with $(z_1 = NA, z_2 = 1)$. In the right panel of Table 2, two cells, $(z_1 = 1, z_2 = 1)$ and $(z_1 = 1, z_2 = 2)$, are merged into a single cell to guarantee a larger size of donors to the recipients. As the result of cell collapsing, recipients who have observed values in z_2 with $z_2 = 1$ or $z_2 = 2$ share the same donors if they have $z_1 = 1$. Since the cell variables z_2 have two distinct values given $z_1 = 2$, the cell collapsing result does not contradict the overall size assumption of $H \geq 2$.

$z_1 \backslash z_2$	1	2
1	0	3
2	1	4

$z_1 \backslash z_2$	1	2
1	3	
2	1	4

Table 1: An illustrative example for the number of donors in cell with $G = 2$ and $H = 2$. Left panel: initial imputation cell; right panel: final cell subdivision after cell collapsing.

(Step 2): Estimating of cell probabilities

Once the imputation cells are finalized from the above discretization, we need to estimate the cell probabilities defined by

$$\pi_{gh} = P(z_1 = g, z_2 = h), \quad g = 1, \ldots, G;\ h = 1, \ldots, H.$$

The initial cell probabilities are obtained using only the respondents in A_R. These initial cell probabilities are updated using the following EM method, modified from the EM by weighting (Ibrahim, 1990):

E-step: Let z_{obs} and z_{mis} be observed and missing part of z, respectively. Then, the conditional probability of $z_{mis} = b^*$ given $z_{obs} = a$, denoted by $\hat{\pi}_{b^*|a}$, is computed using the current t-th estimate of the joint probability, where a is the observed value in z_{obs}, b^* is one possible value for z^*_{mis}, and

$$\hat{\pi}^{(t)}_{b^*|a} = \frac{\hat{P}^{(t)}(z_{i,obs} = a, z_{i,mis} = b^*)}{\sum_b \hat{P}^{(t)}(z_{i,obs} = a, z_{i,mis} = b)}. \tag{5}$$

M-step: Updates the joint probability of a particular combination $z^* = (z_{obs} = a, z^*_{mis} = b^*)$ by

$$\hat{P}^{(t+1)}(z^*) = \left(\sum_{i\in A}^{n} w_i\right)^{-1} \sum_{i\in A}^{n} w_i \hat{\pi}^{(t)}_{b^*|a} I(z_{i,obs} = a). \tag{6}$$

(Step 3): Constructing FEFI fractional weights

The key point of the approach in Im et al. (2015) is to approximate the FEFI by the FHDI method with a smaller size of donors. To achieve this goal, we first need to compute the FEFI fractional weights for all possible donors assigned to each recipient. Let w^*_{ij} be the j-th fractional weights for the recipient i corresponding to donor j given by

$$w^*_{ij} = \hat{\pi}_{z^*_{i,mis}|z_{i,obs}} \frac{w_j I\{(z_{i,obs}, z^*_{i,mis(i)}) = (z_{j,obs(i)}, z_{j,mis(i)})\}}{\sum_{k\in A_R} w_k I\{(z_{i,obs}, z^*_{i,mis(i)}) = (z_{k,obs(i)}, z_{k,mis(i)})\}}, \tag{7}$$

where $z^*_{i,mis}$ is an imputed value for the missing part of recipient i and $(z_{k,obs(i)}, z_{k,mis(i)})$ denotes the values of unit k corresponding to the observed and missing part of recipient i in the sample imputation cell. Here, the FEFI fractional weights are constructed using the cell conditional probability of $z_{i,mis}$ in the missing part given the observed part $z_{i,obs}$. Note that the sum of w^*_{ij} over all j is equal to one by construction.

(Step 4): Imputation

In FEFI, we employed all respondents as donors to each recipient in the same cell, and then assign the FEFI fractional weights to each donor. However, this FEFI may not be attractive in practice due to its huge size. Instead of using all the respondents, we can select just M donors among the FEFI donors with the selection probability proportional to FEFI fractional weights and then assign equal fractional weights. As for donor selection, we used a tailored systematic sampling method given below:

(a) Sort all FEFI donors in y values by the half-ascending half-descending order. That is, for example, $\{1, 2, \ldots, 8\}$ is sorted as follows: 1,3,5,7,8,6,4,2.

(b) Let $k \in A_{Rd}$, where A_{Rd} is a set of the FEFI donors, be the FEFI donors after the sorting algorithm in (a). Let (L_k, U_k) be the interval for the following systematic sampling scheme:

  (b1) $L_1 = 0$. Set $k = 1$.

  (b2) For current k, $U_k = L_k + w^*_{ik}$.

  (b3) Set $k = k + 1$ and $L_k = U_{k-1}$ then go to Step (b2) until $k = n_{Rd}$, where n_{Rd} is a size of A_{Rd}.

(c) Let A_{Mr} be a subset of A_M who has the same values in the observed part and n_{Mr} be the size of A_{Mr}. Let RN be a random number generated from a uniform distribution $U(0, 1)$. For each $l \in A_{Mr}$, we select M donors as follows: for $j = 1, \ldots, M$, if

$$L_k \leq \frac{RN + (l-1)}{n_{Mr}} + (j-1) \leq U_k$$

for some k, then a donor k is selected as the j-th donor for recipient l.

This tailored systematic sampling is designed to select the FEFI donors efficiently within imputation cells. If n_{Rd} is less than M, we select all donors and then assign the FEFI fractional weights instead of assigning equal weights M^{-1}.

Analysis

After imputation, we obtain the imputed data in the size of $n_R + n_M \times M$. On the imputed data, we can conduct statistical analysis such as mean estimation, regression analysis, and so on. The FHDI mean estimator is

$$\bar{y} = \frac{\sum_{i \in A} \sum_{j=1}^{M} w_i w_{ij}^* y_{ij}^*}{\sum_{i \in A} w_i}, \tag{8}$$

where y_{ij}^* and w_{ij}^* are the imputed values and the fractional weights for unit $i \in A_M$, but $y_{ij}^* = y_i$ for all j and $w_{ij}^* = 1$ for $i = j$ and $w_{ij}^* = 0$ for all $i \neq j$ if unit $i \in A_R$.

Similarly, the regression coefficient for the regression of y_1 given y_2 can be written as a classical weighted regression in the form,

$$\hat{\beta} = (X' W X)^{-1} X' W y_1^*, \tag{9}$$

where X is a design matrix including a vector of $y_{2,ij}^*$, and W is a weighting matrix whose diagonal elements are $w_i w_{ij}^*$ and non-diagonal elements are all zeros.

The replication method is considered for variance estimation of the FHDI estimator. For L replicates, the replication variance estimator for the FEFI estimator is

$$\hat{\theta}_{FHDI} = \sum_{k=1}^{L} c_k \left(\hat{\theta}_{FHDI}^{(k)} - \hat{\theta}_{FHDI} \right)^2, \tag{10}$$

where c_k is a replicate factor associated with $\hat{\theta}_{FHDI}^{(k)}$ and $\hat{\theta}_{FHDI}^{(k)}$ is the the k-th replicate estimate obtained using the k-th fractional weights replicate denoted by $w_i^{(k)} \times w_{ij}^{*(k)}$. The current version of the **FHDI** package provides a set of replication fractional weights using a jackknife method. See Im et al. (2015) for details in computation of the replication fractional weights.

Implementation of FHDI

In the **FHDI** package, we have three main functions: (i) `FHDI_CellMake`, (ii) `FHDI_CellProb`, and (iii) `FHDI_Driver`. The function `FHDI_CellMake()` is used to create the imputation cell variable z. The EM algorithm introduced in Section 24.2 is built into the function `FHDI_CellProb()`. The main function `FHDI_Driver()` conducts imputation and variance estimation including cell construction and estimating cell probabilities. We used simulated data to describe these components of the package **FHDI**. All results in this section are obtained from a Microsoft Windows 64 bit operation system. The FEFI results should be the same without reference to the underlying operating system, while the FHDI results using other platforms (e.g., Linux) may be slightly different, as FHDI selects donors using a standard random number library which is generally platform-dependent.

DATA

We have $n = 100$ sample observations for the multivariate data vector $y_i = (y_{1i}, y_{2i}, y_{3i}, y_{4i})$, $i = 1, \ldots, n$, generated from

$$\begin{aligned} Y_1 &= 1 + e_1, \\ Y_2 &= 2 + \rho e_1 + \sqrt{1 - \rho^2} e_2, \\ Y_3 &= Y_1 + e_3, \\ Y_4 &= -1 + 0.5 Y_3 + e_4. \end{aligned}$$

We set $\rho = 0.5$; e_1 and e_2 are generated from a standard normal distribution; e_3 is generated from a standard exponential distribution; and e_4 is generated from a normal distribution $N(0, 3/2)$.

Response indicators are generated from a Bernoulli distribution with different p_k,

$$\delta_k \sim B(p_k),$$

where $(p_1, p_2, p_3, p_4) = (0.6, 0.7, 0.8, 0.9)$. Although the response indicators are generated based on the missing completely at random (MCAR) assumption for simplicity, the FHDI method also holds for other response models based on MAR.

Based on the outcome models and the response models above, an example data are generated using the following R code:

```
n = 100
set.seed(1345)
rho = 0.5
e1 = rnorm(n, 0, 1)
e2 = rnorm(n, 0, 1)
e3 = rgamma(n, 1, 1)
e4 = rnorm(n, 0, sd = sqrt(3/2))

y1 = 1 + e1
y2 = 2 + rho * e1 + sqrt(1 - rho^2) * e2
y3 = y1 + e3
y4 = -1 + 0.5 * y3 + e4

r1 = rbinom(n, 1, p = 0.6)
r2 = rbinom(n, 1, p = 0.7)
r3 = rbinom(n, 1, p = 0.8)
r4 = rbinom(n, 1, p = 0.9)

y1[r1 == 0] = NA
y2[r2 == 0] = NA
y3[r3 == 0] = NA
y4[r4 == 0] = NA
```

Imputation cell

Using the R function `summary()`, we see missing values in all four variables; realized response rates are 0.58, 0.66, 0.82, and 0.89, respectively; and sample means for complete cases are 0.98, 1.93, 1.80, and -0.01, respectively.

```
> daty = cbind(y1, y2, y3, y4)  # data
> datr = cbind(r1, r2, r3, r4)  # response (0:mising cell;  1: observed cell)
> summary(daty)
      y1                y2                y3                y4
Min.   :-1.6701   Min.   :0.02766   Min.   :-1.4818   Min.   :-2.920292
1st Qu.: 0.4369   1st Qu.:1.03796   1st Qu.: 0.9339   1st Qu.:-0.781067
Median : 0.8550   Median :1.79693   Median : 1.7246   Median :-0.121467
Mean   : 0.9821   Mean   :1.93066   Mean   : 1.7955   Mean   :-0.006254
3rd Qu.: 1.6171   3rd Qu.:2.71396   3rd Qu.: 2.5172   3rd Qu.: 0.787863
Max.   : 3.1312   Max.   :5.07103   Max.   : 5.3347   Max.   : 4.351372
NA's   :42        NA's   :34        NA's   :18        NA's   :11
```

We now use the function `FHDI_CellMake()` to convert continuous y variables into discretized variables z. A vector of initial cell dimension k should be given as input information. If the input value k is a single integer, the same number of category is applied to all variables for initial discretization. The current version allows up to 35 distinct categories for each variable. When the variables are specified as the unordered categorical type through the option, the algorithm excludes those variables from the discretization procedure. The details are illustrated in the a ppendix along with the application of mixed data types.

Sample weight and ID are optional for input information. Default values are 1 through n for the ID while 1.0 for the sample weight. Another option is `s_op_merge` which controls randomness in the cell collapsing procedure. There are two possible values for `s_op_merge`: `"fixed"` as default and `"rand"` as optional. During the cell collapsing, randomness occur if there exists multiple adjacent cells for a fixed cell, which is required to be merged into another cell. If `s_op_merge` is set with `"rand"`, then the matrix of discretized values can be different even for the same incomplete data. The last option is `"categorical"` used to denote the data type which has a value 1 if the input variable is unordered and 0 otherwise. If the input vector for the option `"categorical"` is not specified by the users, the algorithm treats all variables as continuous or ordered categorical types.

The main outputs of the function `FHDI_CellMake()` are (a) incomplete data matrix attached with ID and sample weight named by "`data`", (b) matrix for imputation cell variables named by "`cell`", (c) cell pattern matrix for respondents named by "`cell.resp`", and (d) cell pattern matrix for nonrespondents named by "`cell.non.resp`". When missing values are recorded with specific numeric values other than `NA`, the response indicator matrix (i.e., "`datr`") should be inserted with the original data matrix. The indicator matrix "`datr`" has the same dimension as "daty" contains 0 for missing cell locations and 1 otherwise. For instance, the first six rows of "`datr`" of the "`daty`" given above will look like:

```
y1  y2  y3  y4
 1   1   0   1
 0   1   1   1
 1   1   1   0
 0   1   0   1
 1   1   1   1
 1   1   1   1
```

As long as "daty" contains NA at the missing cell locations, automatic detection of missing cell locations takes place. When one wants to define more missing cell locations, one can override the automatic detection by separately defining "datr" and including it in the function argument, for example, 'FHDI_CellMake(daty,datr,k=3)'. We set $k = 3$ and do not give additional input information on weights and ID, then the first output is shown as

```
> cdaty = FHDI_CellMake(daty, k = 3)
> names(cdaty)
[1] "data"          "cell"          "cell.resp"     "cell.non.resp"
[5] "w"             "s_op_merge"
> head(cdaty$data)
     ID WT         y1       y2        y3           y4
[1,]  1  1 1.47963286 2.150860        NA  1.894211796
[2,]  2  1         NA 1.141496 1.6025296 -1.036946859
[3,]  3  1 0.70870936 1.885673 1.2506894           NA
[4,]  4  1         NA 2.753840        NA  1.211049509
[5,]  5  1 0.86273572 2.425549 1.8875492 -0.539284732
[6,]  6  1 0.03460025 1.740481 0.4909525  0.007130484
```

The observed values are converted to ordered categorical values from 1 to k. Here, missing values are presented with a common missing value "0". During the discretization procedure, the initial dimension of k can be down to have larger donors within imputation cells. In our example, the default value $k = 3$ was kept for all variables.

```
> head(cdaty$cell)
     y1 y2 y3 y4
[1,]  3  2  0  3
[2,]  0  1  1  1
[3,]  2  2  2  0
[4,]  0  2  0  3
[5,]  2  3  2  2
[6,]  1  2  1  2

> apply(cdaty$cell, 2, table)
  y1 y2 y3 y4
0 42 34 18 11
1 26 18 27 30
2 13 36 32 28
3 19 12 23 31
```

From the output component named by "cell", we can find out whether the cells are merged during the discretization process. For instance, we have 34 missing values in y_2, and this indicates that 22 $(= 66/3)$ observations are evenly distributed for each category based on the estimated density function. However, the finalized frequencies for three categories are 18, 36, and 12. This means that some initial cells are merged to adjacent cells to obtain enough donors for recipients.

It is important to specify the cell patterns to match the respondents and the nonrespondents. Thus, the cell pattern matrix for respondents and nonrespondents, named by "cell.resp" and "cell.non.resp", are produced as the outputs of the function FHDI_CellMake(). Note that it is possible to have $3^4 = 81$ distinct vectors theoretically, but we only have 10 unique patterns in our toy example.

```
> cdaty$cell.resp
     y1 y2 y3 y4
[1,]  1  1  1  1
[2,]  1  1  2  3
[3,]  1  2  1  2
[4,]  1  2  2  1
```

```
[5,]  2  2  2  3
[6,]  2  3  2  2
[7,]  3  1  3  3
[8,]  3  2  3  2
[9,]  3  2  3  3
[10,]  3  3  3  1
```

Similarly, a set of unique cell patterns for nonrespondents are reported at the fourth output. We have 47 patterns for nonrespondents in this example. Some patterns are presented below

```
> head(cdaty$cell.non.resp)
     y1 y2 y3 y4
[1,]  0  0  0  2
[2,]  0  0  0  3
[3,]  0  0  1  1
[4,]  0  0  1  2
[5,]  0  0  2  1
[6,]  0  0  2  2
```

Note that it is possible to have all zero values as a cell pattern, because we allow to have missing values for all variables. Under the MAR assumption, if there are no additional auxiliary variables, then the cases with missing values in all items can be safely removed from the original data set.

Cell probability estimation

We now estimate the cell probabilities based on the second output obtained using `FHDI_CellMake()`. Since the function `FHDI_CellProb()` is based on the EM algorithm designed to compute cell probabilities for multivariate categorical data, this function can be separately used when we are only interested in obtaining the maximum likelihood estimates for the cell probabilities.

```
> datz = cdaty$cell
> jcp = FHDI_CellProb(datz)
> jcp$cellpr
1111       1123       1212       1221       2223       2322
0.18110421 0.05474648 0.12693514 0.07786676 0.17388579 0.08263912
3133       3232       3233       3331
0.02175015 0.10356376 0.08871434 0.08879425
> sum(jcp$cellpr)
[1] 1
```

Note that the joint cell probabilities are estimated only from observed patterns, not from all possible values. Overall sum of the joint cell probabilities should be equal to one. All discretized values are presented by a single values in the order of data columns. For example, a value of "1111" denotes a vector $(z_1 = 1, z_2 = 1, z_3 = 1, z_4 = 1)$. If the number of categories in a variable is larger than 10, the categories are label with 26 alphabet letters (a-z). If we have a value 'b2c', then it denotes the z vector $(11, 2, 12)$.

Fractional hot deck imputation

We can use the function `FHDI_Driver()` without searching for a suitable imputation cell matrix in advance. In short, `FHDI_Driver()` automatically performs `FHDI_CellMake()` and `FHDI_CellProb()` and then proceeds toward the imputation and/or variance estimation. The main input information is the matrix of original variables, matrix for response indicators, and the imputation method (`"FEFI"`, `"FHDI"`). In the case of '`s_op_imputation="FHDI"`', the imputation size M should be given as an input value. For instance, when $M = 5$, FHDI will randomly select 5 donors from all possible donors. In the case of `"FEFI"`, the imputation is conducted to assign all possible values to each recipient with the FEFI fractional weights.

The output consists of four main parts except for input information: (i) imputation results with fractional weights named by "`fimp.data`", (ii) an imputed data in format of single imputation result named by "`simp.data`", (iii) the FHDI mean estimates with estimated standard error named by "`imp.mean`", and (iv) replication fractional weights for variance estimation named by "`rep.weight`". If variance estimation option is given by '`i_op_variance=0`', then third output and fourth output are not produced as the main output. The default is '`i_op_variance=1`'.

The part of the fractionally imputed data are given below with newly attached variables FID and FWT, where FID denotes donors' local serial index and FWT denotes fractional weights assigned to imputed values:

```
> FEFI = FHDI_Driver(daty, s_op_imputation = "FEFI", i_op_variance = 1, k = 3)
> names(FEFI)
[1] "fimp.data"       "simp.data"       "imp.mean"
[4] "rep.weight"      "M"               "s_op_imputation"
[7] "i_option_merge"

> FEFI$fimp.data[1:20,]
      ID FID WT       FWT          y1       y2       y3         y4
 [1,]  1   1  1 0.5000000  1.47963286 2.150860 2.881646  1.8942118
 [2,]  1   2  1 0.5000000  1.47963286 2.150860 2.493438  1.8942118
 [3,]  2   1  1 0.2000000 -0.09087472 1.141496 1.602530 -1.0369469
 [4,]  2   2  1 0.2000000 -1.67006193 1.141496 1.602530 -1.0369469
 [5,]  2   3  1 0.2000000 -0.39302750 1.141496 1.602530 -1.0369469
 [6,]  2   4  1 0.2000000  0.97612864 1.141496 1.602530 -1.0369469
 [7,]  2   5  1 0.2000000  0.21467221 1.141496 1.602530 -1.0369469
 [8,]  3   1  1 0.1666667  0.70870936 1.885673 1.250689  0.7770526
 [9,]  3   2  1 0.1666667  0.70870936 1.885673 1.250689  1.2839115
[10,]  3   3  1 0.1666667  0.70870936 1.885673 1.250689  0.6309413
[11,]  3   4  1 0.1666667  0.70870936 1.885673 1.250689  0.3232018
[12,]  3   5  1 0.1666667  0.70870936 1.885673 1.250689  0.5848844
[13,]  3   6  1 0.1666667  0.70870936 1.885673 1.250689  1.0342970
[14,]  4   1  1 0.1103616  1.22307261 2.753840 1.923865  1.2110495
[15,]  4   2  1 0.1689153  2.29021618 2.753840 2.881646  1.2110495
[16,]  4   3  1 0.1103616  0.86825894 2.753840 1.086562  1.2110495
[17,]  4   4  1 0.1103616  2.16515160 2.753840 2.311461  1.2110495
[18,]  4   5  1 0.1103616  0.79827971 2.753840 3.040016  1.2110495
[19,]  4   6  1 0.1689153  2.01819696 2.753840 2.493438  1.2110495
[20,]  4   7  1 0.1103616  0.73228949 2.753840 3.422369  1.2110495
```

The singly imputed data has the same size as the original incomplete data in which missing values are filled with a single value, essentially the mean of fractionally imputed values. For example, in the first sample presented at 'daty[1,]' below, we have a missing value for y_3. The imputed values are 2.881646 and 2.493438 presented above, and the weighted mean of two values 2.6875422, considering sample weights and fractional weights, replaces the missing value. This second output can be used as the result for single imputation. However, its uses should be controlled under the limitations of single imputation. The second output is partially presented below

```
> daty[1,]
      y1       y2       y3       y4
1.479633 2.150860       NA 1.894212

> head(FEFI$simp.data)
              y1       y2        y3           y4
[1,]  1.47963286 2.150860 2.6875422  1.894211796
[2,] -0.19263266 1.141496 1.6025296 -1.036946859
[3,]  0.70870936 1.885673 1.2506894  0.772381422
[4,]  1.44470586 2.753840 2.3372705  1.211049509
[5,]  0.86273572 2.425549 1.8875492 -0.539284732
[6,]  0.03460025 1.740481 0.4909525  0.007130484
```

The component "imp.mean" shows the FEFI mean estimates with the standard errors. The first row presents the FEFI mean estimates and the second row presents the standard error of the FEFI mean estimates. In our example, the FEFI mean estimates (standard errors) for variables are approximately $0.90(0.128)$, $1.87(0.121)$, $1.82(0.137)$, and $-0.03(0.130)$, respectively.

```
> FEFI$imp.mean
          [,1]      [,2]      [,3]        [,4]
[1,] 0.9049227 1.8668846 1.8188381 -0.03193875
[2,] 0.1275504 0.1206944 0.1369989  0.12958845
```

The standard errors are computed using the variance formula in (10) with the replicated fractional weights reported in the fourth output. In a random sample, the associate factor c_k has the same value for all replicates. For instance, $c_k = (n-1)/n$ for the jackknife variance estimation. Replication fractional weights for the imputed data are given below:

```
> dim(FEFI$rep.weight)
[1] 330 100
> FEFI$rep.weight[1:13, 1:6]
           [,1]      [,2]      [,3]      [,4]      [,5]      [,6]
 [1,] 0.0000000 0.5050505 0.5050505 0.5050505 0.5050505 0.5050505
 [2,] 0.0000000 0.5050505 0.5050505 0.5050505 0.5050505 0.5050505
 [3,] 0.2020202 0.0000000 0.2020202 0.2020202 0.2020202 0.2020202
 [4,] 0.2020202 0.0000000 0.2020202 0.2020202 0.2020202 0.2020202
 [5,] 0.2020202 0.0000000 0.2020202 0.2020202 0.2020202 0.2020202
 [6,] 0.2020202 0.0000000 0.2020202 0.2020202 0.2020202 0.2020202
 [7,] 0.2020202 0.0000000 0.2020202 0.2020202 0.2020202 0.2020202
 [8,] 0.1683502 0.1683502 0.0000000 0.1683502 0.1683502 0.1683502
 [9,] 0.1683502 0.1683502 0.0000000 0.1683502 0.1683502 0.1683502
[10,] 0.1683502 0.1683502 0.0000000 0.1683502 0.1683502 0.1683502
[11,] 0.1683502 0.1683502 0.0000000 0.1683502 0.1683502 0.1683502
[12,] 0.1683502 0.1683502 0.0000000 0.1683502 0.1683502 0.1683502
[13,] 0.1683502 0.1683502 0.0000000 0.1683502 0.1683502 0.1683502
```

Because a jackknife method is used for variance estimation, a matrix of the replication fractional weights has n_I rows and n columns, where $n_I (> n)$ denotes the size of the imputed data. By its construction, the column sums of the replication fractional weights should be the sum of unit weights, that is, $\sum_{i=1}^{n} w_i^{(k)} = \sum_{i=1}^{n} w_i$ holds for all k.

Instead of the FEFI, we can perform a general FHDI with $M(\geq 2)$. If the imputation method is chosen to be `"FHDI"` and an imputation size M is given, then M donors are assigned to each recipients. If the number of donors for a recipient is smaller than M, all donors are selected and assigned the FEFI fractional weights.

Note that in the recipient with ID=3 (see '`FEFI$fimp.data[1:20,]`' results shown above), there are six donors for FEFI but there are five donors $(M = 5)$ for FHDI (see '`FHDI$fimp.data[1:14,]`' results shown below). Five donors are randomly selected among all possible six donors with the probability proportional to the FEFI fractional weights. This indicates that we have additional randomness due to donor selection in the FHDI method. As shown in '`FHDI$imp.mean`' results below, the FHDI mean estimates (standard errors) are approximately $0.90(0.129)$, $1.87(0.121)$, $1.82(0.137)$, and $-0.04(0.131)$, respectively. Compared to the FEFI estimates, both point and variance estimates are similar to each other. This indicates that the FHDI estimator well approximates the FEFI estimator with smaller size of the imputed data. The results (which can vary slightly between operating systems) are presented below:

```
> FHDI = FHDI_Driver(daty, s_op_imputation="FHDI", M=5, i_op_variance=1, k=3)
> FHDI$fimp.data[1:14,]
      ID FID WT FWT          y1       y2       y3         y4
 [1,]  1   1  1 0.5  1.47963286 2.150860 2.881646  1.8942118
 [2,]  1   2  1 0.5  1.47963286 2.150860 2.493438  1.8942118
 [3,]  2   1  1 0.2 -0.09087472 1.141496 1.602530 -1.0369469
 [4,]  2   2  1 0.2 -1.67006193 1.141496 1.602530 -1.0369469
 [5,]  2   3  1 0.2 -0.39302750 1.141496 1.602530 -1.0369469
 [6,]  2   4  1 0.2  0.97612864 1.141496 1.602530 -1.0369469
 [7,]  2   5  1 0.2  0.21467221 1.141496 1.602530 -1.0369469
 [8,]  3   1  1 0.2  0.70870936 1.885673 1.250689  0.7770526
 [9,]  3   2  1 0.2  0.70870936 1.885673 1.250689  1.2839115
[10,]  3   3  1 0.2  0.70870936 1.885673 1.250689  0.6309413
[11,]  3   4  1 0.2  0.70870936 1.885673 1.250689  0.3232018
[12,]  3   5  1 0.2  0.70870936 1.885673 1.250689  1.0342970
[13,]  4   1  1 0.2  1.22307261 2.753840 1.923865  1.2110495
[14,]  4   2  1 0.2  2.29021618 2.753840 2.881646  1.2110495

> FHDI$imp.mean
[,1]     [,2]      [,3]        [,4]
[1,] 0.9032611 1.865013 1.8204121 -0.03900379
[2,] 0.1291721 0.120509 0.1367363  0.13086824
```

Table 2 presents the standard errors of the three mean estimators. Here, the Naive estimator is just a simple mean estimator computed using only observed values. Since the partially observed values are used in the mean estimation, the two estimators obtained using fractional hot deck imputation produce smaller standard errors compared to the Naive estimator.

Estimator	y_1	y_2	y_3	y_4
Naive	0.135	0.135	0.150	0.138
FEFI	0.128	0.121	0.137	0.130
FHDI	0.129	0.121	0.137	0.131

Table 2: Standard errors of three mean estimators.

It should be noted that the automatically generated "`datz`" can be replaced with a user-defined one. If the imputation cell matrix "`datz`" is separately obtained from `FHDI_CellMake()` or provided by the user, it needs to be specified as an option of the function `FHDI_Driver()`. An example code is given below:

```
> FEFI=FHDI_Driver(daty,datz,s_op_imputation="FEFI",i_op_variance=1,k=3)
```

If survey variables are all categorical, the original data "`daty`" can be directly used for "`datz`." In the case of mixed types including continuous and/or ordered and unordered categorical data, we need two steps to obtain the imputation cell matrix: (i) discretization procedure for the continuous parts, and (ii) combining procedure for the converted data and the unordered categorical data. If an error message is presented due to insufficient of the donors from the function `FHDI_CellMake()`, then the cell collapsing procedure has to be engaged manually for the categorical part. We illustrate an example in Appendix when and how we manually merge categories in practice.

Regression analysis

We now consider a regression analysis using the imputed data. The imputed data can be treated as complete data with assigned fractional weights. Thus, the regression coefficient estimates of y_1 given y_2 can be directly obtained using the `lm()` function. Compared to using only observed samples, the fractional weights are given as input information. The estimates are also obtained using a classical weighted regression estimator presented in Section 24.2. The R codes for obtaining the regression coefficient estimates for the three estimators are given below:

```
> reg.naive = lm(y1 ~ y2, data = as.data.frame(daty))
> reg.naive$coeff
(Intercept)          y2
-0.07425917  0.58798028
> summary(reg.naive)$coeff[,2]
(Intercept)          y2
  0.3050262   0.1424451

> i.daty = as.data.frame(FEFI$fimp.data)
> reg.fefi = lm(y1 ~ y2, data = i.daty, weights = FWT)
> reg.fefi$coeff
(Intercept)          y2
 0.03465671  0.46615949

> i.daty2 = as.data.frame(FHDI$fimp.data)
> reg.fhdi = lm(y1 ~ y2, data = i.daty2, weights = FWT)
> reg.fhdi$coeff
(Intercept)          y2
  0.0227328   0.4721299
```

Note that if we directly use `lm()` on the imputed data, the standard errors are 0.103 and 0.048 for the FEFI estimator and 0.111 and 0.052 for the FHDI estimator, respectively. However, the standard errors using the replication method are nearly 0.251 and 0.094 for both the FEFI estimator and FHDI estimator. Because the variances coming from imputation are not captured with a classical variance estimator of the regression estimator, variance estimation should be conducted using the replication variance estimator formula in (10).

Estimator	Intercept (S.E.)	Slope (S.E.)
True	0	0.5
Naive	-0.074 (0.305)	0.588 (0.142)
FEFI	0.035 (0.103)	0.466 (0.048)
FHDI	0.023 (0.111)	0.472 (0.052)

Table 3: Regression coefficient estimates with standard errors. (S.E.: standard error.)

Table 3 presents the regression coefficient estimates with standard errors for the three estimators. Point estimates of the FEFI and FHDI estimators are much closer to the true values compared to the values of the Naive estimator. Also, two fractional imputation estimators have smaller standard errors than those of the naive estimator. All R codes to obtain these results are given below:

```
> reg.fefi.coef = reg.fefi$coeff
> reg.est = t(apply(FEFI[[4]], 2, function(s) lm(y1 ~ y2, data = i.daty,
    weights = s)$coeff))
> reg.fefi.rep = reg.est - matrix(reg.fefi.coef, n, 2, byrow = TRUE)
> sqrt(apply(reg.fefi.rep^2, 2, sum) * (n - 1)/n)
(Intercept)          y2
 0.25082547  0.09369202
> summary(reg.fefi)$coeff[,2]
(Intercept)          y2
 0.10333038  0.04840829

> reg.fhdi.coef = reg.fhdi$coeff
> reg.est2 = t(apply(FHDI[[4]], 2, function(s) lm(y1 ~ y2, data = i.daty2,
    weights = s)$coeff))
> reg.fhdi.rep = reg.est2 - matrix(reg.fhdi.coef, n, 2, byrow = TRUE)
> sqrt(apply(reg.fhdi.rep^2, 2, sum) * (n - 1)/n)
(Intercept)          y2
 0.25165391  0.09524197
> summary(reg.fhdi)$coeff[,2]
(Intercept)          y2
 0.11149137  0.05230785
```

Conclusion

FHDI is a useful tool for handling item nonresponse. Since FHDI employs a nonparametric estimation of the joint distribution and uses observed values as imputed values, it can be widely accepted in many research and incomplete data analysis. This paper documents the R package **FHDI** to enable R users to perform fractional hot deck imputation as well as fully efficient fractional imputation.

The current version of the package **FHDI** has some limitations. First, the functions within the package are not always applicable for all types of incomplete data. The current imputation algorithms cannot be applied to fill in missing values when there is no fully observed units over all variables. Although the users may assume conditional independence to apply our imputation algorithms, the imputation on the basis of untestable conditional independence assumption may break the original data structure. Second, the unordered categorical data should be handled manually for the insufficient donor cases before the users use `FHDI_Driver()`. However, overcoming this limitation is identical to coming up with a way to implement cell collapsing with no regard to the rationality of categorization. Also, jackknife variance estimation may be not appropriate or efficient for larger or complicated incomplete data. Algorithm-oriented parallel computing methods and substantial issues including variable types and variance estimation options will be targeted in the future update.

Bibliography

H. Y. Chen. Compatibility of conditionally specified models. *Statistics and Probability Letters*, 80:670–677, 2010. [p140]

G. B. Durrant and C. Skinner. Using missing data methods to correct for measurement error in a distribution function. *SM*, 32:25–36, 2006. [p140]

R. E. Fay. Alternative paradigms for the analysis of imputed survey data. *JASA*, 91:490–498, 1996. [p140]

W. A. Fuller and J. K. Kim. Hot deck imputation for the response model. *SM*, 31:139–149, 2005. [p140, 141]

J. Honaker, G. King, and M. Blackwell. Amelia ii: A program for missing data. *Journal of Statistical Software*, 45:1–47, 2011. [p140]

J. G. Ibrahim. Incomplete data in generalized linear models. *JASA*, 85:765–769, 1990. [p141, 143]

J. Im, J. K. Kim, and W. A. Fuller. Two-phase sampling approach to fractional hot deck imputation. In *JSM Proceedings of Survey Research Methodology Section*, pages 1030–1043, Seattle, WA, 2015. asa. [p140, 141, 142, 143, 144]

J. Im, I. H. Cho, and J. K. Kim. *FHDI: Fractional Hot Deck and Fully Efficient Fractional Imputation*, 2018. URL https://CRAN.R-project.org/package=FHDI. [p141]

G. Kalton and L. Kish. Some efficient random imputation methods. *Communications in Statistics-Theory and Methods*, 13:1919–1939, 1984. [p140]

J. K. Kim. Fractional hot deck imputation. *BMK*, 98:119–132, 2011. [p140]

J. K. Kim and W. A. Fuller. Fractional hot deck imputation. *BMK*, 91:559–578, 2004. [p140, 141]

A. Kowarik and M. Templ. Imputation with the R package VIM. *Journal of Statistical Software*, 74:1–16, 2016. [p140]

D. B. Rubin. Inference and missing data. *BMK*, 63:581–592, 1976. [p140]

D. B. Rubin. *Multiple Imputation for Nonresponse in Survey*. John Wiley & Sons, New York, 1987. [p140]

D. B. Rubin. Multiple imputation after 18+ years. *JASA*, 91:473–489, 1996. [p140]

SAS Institue Inc. *SAS/STAT 14.1 User's Guide*. SAS Institue Inc., Cary, NC, 2015. URL http://support.sas.com/documentation/. [p140]

Y. S. Su, A. Gelman, J. Hill, and M. Yajima. Multiple imputation with diagnostics (mice) in r: Opening windows into the black box. *Journal of Statistical Software*, 45:1–31, 2011. [p140]

S. van Buuren and K. Groothuis-Oudshoorn. mice: Multivariate imputation by chained equations in R. *Journal of Statistical Software*, 45:1–67, 2011. [p140]

S. Yang and J. K. Kim. A note on multiple imputation for method of moments estimation. *BMK*, 103: 244–251, 2016. [p140]

Jongho Im
Department of Applied Statistics
Yonsei University
South Korea
ijh38@yonsei.ac.kr

In Ho Cho
Department of Civil, Construction and Environmental Engineering
Iowa State University
United States
icho@iastate.edu

Jae Kwang Kim
Center for Survey Statistics and Methodology
Department of Statistics
Iowa State University
United States
jkim@iastate.edu

Appendix

We here present how R users can handle mixed types, combination of continuous and unordered categorical data, in practice. For illustration purpose, we use the exit poll data collected to predict the 18th South Korean legislative election held in 2008. The exit poll data include data collection channel, the sampled voters' gender, age and candidate. We simply assume the MAR mechanism without any further discussion to guide how the R users handle categorical data or mixed type data in uses of `FHDI_CellMake()`.

Table A.1 gives a summary of the exit poll results in an example district. The data were collected from six different channels, denoted by $1, 2, \ldots, 6$, and the size of exit poll was 2210. Gender was recorded with 1 (Male) and 2 (Female), and age was recorded as a numerical value. There were five parties for the voters' choice, recorded by 1,2,5,6,7. The response rates for the gender, age and candidate were 98%, 98%, and 80%, respectively.

Variable	Type	Size of Support	Observation	Response Rate (%)
Channel	Categorical	6	2210	100
Gender	Categorical	2	2172	98
Age	Numerical	72	2170	98
candidate	Categorical	5	1764	80

Table A.1: A summary of the exit poll results in an example district

For imputation, we first consider the case in which the variables are given as their original types presented in Table A.1. From the input data types, the cell collapsing procedure can be only applied to the variable 'age'. Since three variables, 'channel', 'gender' and 'candidate', require 60 $(= 6 \times 2 \times 5)$ imputation cells, the final size will be $60 \times C_{age}$, where C_{age} is the finally discretized cell size of $'age'$. It is easy to fail to have enough donors within each imputation cell due to sparsity problem. When we type the R codes below,

```
> cell.election=FHDI_CellMake(election,k=3,categorical=c(1,1,0,1))
```

we have the following error message:

The current data set does not have enough donors while there is at least one non-collapsible categorical variable!

To make feasible imputation cells, in addition to 'age', we want to also merge parties to reduce the cell dimensions. One possible approach is to un-specify the variable 'candidate' as the unordered categorical values, and the other approach is to merge candidate manually according to the users' background knowledge of the parties. The results for the first approach are presented below:

```
> cell.election = FHDI_CellMake(election, k = 3,categorical = c(1, 1, 0, 0))
> apply(cell.election$cell, 2, table)
$channel
  1   2   3   4   5   6
396 708 508 276 127 195

$gender
  0    1    2
 38 1042 1130

$age
  0   1   2   3
 40 775 672 723

$candidate
  0   1   2   3
446 788 864 112
```

The observed candidates are 1 (788), 2 (864), 5 (73), 6 (27), and 7 (12), and they are merged into three categories 1 (788), 2 (864), and 3 (112). During the cell collapsing, the initial integer values are changed to ordered categorical values and then merged to search out the feasible imputation cells. Although the imputation cells are now well constructed without error message, the cell collapsing results may not be undesirable in the sense that the automatically merged candidates (parties) using the algorithm

in the FHDI_CellMake() may have the opposite political positions. Note that even if the size of initial categories is given as $k = 3$, the dimension of variables specified as the unordered categorical type is kept to preserve their original sizes.

To avoid irrational cell collapsing result, the second approach is implemented by manually combining the candidates' parties into two groups, 1 (1,6,7) and 2 (2,5), based on their political characteristics. The results with the R codes are given below:

```
> election2 <- election
> election2$candidate[election2$candidate == 5] <- 2
> election2$candidate[election2$candidate == 6] <- 1
> election2$candidate[election2$candidate == 7] <- 1
> cell.election2 = FHDI_CellMake(election2, k = 3, categorical = c(1, 1,
    0, 1))
> apply(cell.election2$cell, 2, table)
$channel
  1   2   3   4   5   6
396 708 508 276 127 195

$gender
  0    1    2
 38 1042 1130

$age
  0   1   2   3
 40 775 672 723

$candidate
  0   1   2
446 827 937
```

As discussed above, there is no errors for both automated and manual cell collapsing procedures. However, the quality of imputation in the automation case can be unsatisfactory due to the inaccuracy of joint distribution approximation. This implies that it is often required to carefully and/or manually handle the unordered categorical variables in practice. Also note that we may want to discretize the variable 'age' with a set of fixed distinct points, for example, (19-29, 30-39, 40-49,50-59, 60+) or (19-39, 40-59, 60+). In this case, the R users also need to manually replace the initial 'age' values to the discretized values in advance of using the function FHDI_CellMake().

Bayesian Testing, Variable Selection and Model Averaging in Linear Models using R with BayesVarSel

by Gonzalo Garcia-Donato and Anabel Forte

Abstract In this paper, objective Bayesian methods for hypothesis testing and variable selection in linear models are considered. The focus is on **BayesVarSel**, an R package that computes posterior probabilities of hypotheses/models and provides a suite of tools to properly summarize the results. We introduce the usage of specific functions to compute several types of model averaging estimations and predictions weighted by posterior probabilities. **BayesVarSel** contains exact algorithms to perform fast computations in problems of small to moderate size and heuristic sampling methods to solve large problems. We illustrate the functionalities of the package with several data examples.

An illustrated overview of BayesVarSel

Testing and variable selection problems are taught in almost any introductory statistical course. In this first section we assume such background to present the essence of the Bayesian approach and the basic usage of **BayesVarSel** (Garcia-Donato and Forte, 2015) with hardly any mathematical formulas. Our motivating idea in this first section is mainly to present the appeal of the Bayesian answers to a very broad spectrum of applied researchers. This introductory section concludes with a discussion about connections with potentially related R packages.

The remaining six sections are organized as follows. In the second section, on page 158, the problem is presented and the notation needed is introduced jointly with the basics of the Bayesian methodology. Then, two sections follow with explanations of the details concerning the obtention of posterior probabilities in hypothesis testing and variable selection problems, respectively, in **BayesVarSel**. In a later section, on page 163, several tools to describe the posterior distribution are explained, while the section on page 166 is devoted to model averaging techniques. The paper concludes with a section with plans for the future of the **BayesVarSel** project. This paper is supplemented with an appendix, with formulas for the most delicate ingredient in the underlying problem in **BayesVarSel**, namely the prior distributions for parameters within each model.

The version of **BayesVarSel** presented here is 1.8.0.

Testing

In testing problems, several competing hypotheses, H_i, about a phenomenon of interest are postulated. The role of statistics is to provide summaries about the evidence in favor (or against) the hypotheses once the data, $\boldsymbol{y}$, have been observed. There are many important statistical problems with roots in testing like model selection (or model choice) and model averaging.

The formal Bayesian response to testing problems is based on the posterior probabilities of the hypotheses that summarize, in a fully understandable way, all the available information. In the R package **BayesVarSel** a number of popular objective priors (in the sense explained in Berger, 2006) are available. Any of these priors have the great appeal of being fully automatic for users.

For illustrative purposes consider the nutrition problem in Lee (1997), page 143, with data:

```
> weight.gains <- c(134, 146, 104, 119, 124, 161, 107, 83, 113, 129, 97, 123,
+ 70, 118, 101, 85, 107, 132, 94)
```

There it is tested, based on the sample of 19 weight gains (expressed in grams) of rats, whether there is a difference between the population means of the group with a high proteinic diet (the first 12) or the control group (the rest):

```
> diet <- as.factor(c(rep(1,12), rep(0,7)))
> rats <- data.frame(weight.gains = weight.gains, diet = diet)
```

This problem (usually known as the two-samples t-test) is normally written as $H_0 : \mu_1 = \mu_2$ versus $H_1 : \mu_1 \neq \mu_2$, where it is assumed that the weight gains are normally distributed with an unknown (but common) standard deviation. The formulas that define each of the models under the postulated hypotheses are in R language

```
> M0 <- weight.gains ~ 1
> M1 <- weight.gains ~ diet
```

The function to perform Bayesian tests in **BayesVarSel** is `Btest` which has an intuitive and simple syntax (see the section on page 160 for a detailed description). In this example

```
> Btest(models = c(H0 = M0, H1 = M1), data = rats)
Bayes factors (expressed in relation to H0)
 H0.to.H0  H1.to.H0
1.0000000 0.8040127
---------
Posterior probabilities:
   H0    H1
0.554 0.446
```

From these results, we clearly conclude that both hypotheses are similarly supported by the data. Hence there is no evidence that the diet has any impact on the average weight.

A useful guide to interpret Bayes factors and probabilities, follows from the categories proposed by Kass and Raftery (1995), reproduced in Table 1. For the problem above, none of the hypotheses is worth more than a bare mention.

B_{10}	Probability	Evidence against the null
1 to 3	0.5 to 0.75	Not worth more than a bare mention
3 to 20	0.75 to 0.95	Substantial
20 to 150	0.95 to 0.99	Strong
$>$150	> 0.99	Decisive

Table 1: Interpretation of Bayes factors from Kass and Raftery (1995) (B_{10} stands for the Bayes factor of H_1 to the null hypothesis H_0). The column probability is obtained assuming that both hypotheses are equally likely a priori.

Another illustrative example concerns the classic dataset `savings` in Belsley et al. (2005) considered by Faraway (2002), page 29 and distributed under the package **faraway** (Faraway, 2016).

```
> data("savings", package = "faraway")
```

This dataset contains macroeconomic data on 50 different countries during 1960-1970 and the question posed is to elucidate if `dpi` (per-capita disposable income in U.S), `ddpi` (percent rate of change in per capita disposable income), population under (over) 15 (75) `pop15` (`pop75`) are all explanatory variables for `sr`, the aggregate personal saving divided by disposable income which is assumed to follow a normal distribution. This can be written as a testing problem about the regression coefficients associated with the variables with hypotheses

$$H_0 : \beta_{dpi} = \beta_{ddpi} = \beta_{pop15} = \beta_{pop75} = 0,$$

versus the alternative, say H_1, that all predictors are needed. The competing models can be defined as

```
> fullmodel <- sr ~ pop15 + pop75 + dpi + ddpi
> nullmodel <- sr ~ 1
```

and the testing problem can be solved with

```
> Btest(models = c(H0 = nullmodel, H1 = fullmodel), data = savings)
---------
Bayes factors (expressed in relation to H0)
H0.to.H0 H1.to.H0
```

```
  1.0000  21.46007
---------
Posterior probabilities:
    H0    H1
 0.045 0.955
```

Now, the conclusion is that there is strong evidence favoring H_1, the hypothesis that all considered predictors explain the response `sr`.

Of course, more hypotheses can be tested at the same time. For instance, a simplified version of H_1 that does not include `pop15` is

$$H_2 : \beta_{dpi} = \beta_{ddpi} = \beta_{pop75} = 0,$$

that can be included in the analysis as

```
> reducedmodel <- sr ~ pop75 + dpi + ddpi
> Btest(models = c(H0 = nullmodel, H1 = fullmodel, H2 = reducedmodel),
+  data = savings)
Bayes factors (expressed in relation to H0)
  H0.to.H0   H1.to.H0   H2.to.H0
 1.0000000 21.4600656  0.7017864
---------
Posterior probabilities:
   H0    H1    H2
 0.043 0.927 0.030
```

Obviously, as more hypotheses are considered, the usefulness of Table 1 to interpret posterior probabilities reduces since it was conceived for two hypotheses. Nevertheless, in this case, H_1 is clearly supported by the data as the best explanation for the experiment among those considered.

This scenario can be extended to check which subset of the four covariates is the most suitable one to explain `sr`. In general, the problem of selecting the best subset of covariates from a group of potential ones is better known as variable selection.

Variable selection

Variable selection is a multiple testing problem where each hypothesis proposes a possible subset of the p potential explanatory variables initially considered. Notice that there are 2^p hypotheses, including the simplest one (the null hypothesis) stating that none of the variables should be used.

A variable selection approach to the economic example above with $p = 4$ has 16 hypotheses and can be solved using the `Btest` function. Nevertheless, **BayesVarSel** has specific facilities to handle the specificities of variable selection problems. A main function for variable selection is `Bvs`, fully described in the section on page 161. It has a simple syntax inspired by the well-known `lm` function. The variable selection problem in this economic example can be solved by executing:

```
> Bvs(formula = sr ~ pop15 + pop75 + dpi + ddpi, data = savings)
The 10 most probable models and their probabilities are:
   pop15 pop75 dpi ddpi        prob
1      *     *   *    * 0.297315642
2      *     *        * 0.243433493
3      *              * 0.133832367
4      *                0.090960327
5      *         *    * 0.077913429
6      *     *          0.057674755
7      *         *      0.032516780
8      *     *   *      0.031337639
9                        0.013854369
10           *        * 0.006219812
```

With a first look at these results, we can see that the most probable model is the model with all covariates (probability 0.30), which is closely followed by the one without `dpi` with a

posterior probability of 0.24. Note that the results are sensitive to the choice of parameter priors (see the appendix for more details on the priors).

As we will see later, a variable selection exercise generates a lot of valuable information of which the above printed information is only a very reduced summary. This additional information can be accessed with specific methods that explore the characteristics of objects of the type created by `Bvs`.

Related packages

In a recent study, Forte et al. (2018) analyse the differences among available R packages that, according to either title and/or description, perform common variable selection (and related) tasks. The main underlying motivation in that study was to clarify the commonalities/differences of the different packages available sharing a common theoretical framework (Bayesian variable selection with a particular type of priors). Five packages turned out to have these characteristics: **BayesFactor** (Morey et al., 2015); **BayesVarSel**; **BMS** (Zeugner and Feldkircher, 2015); **mombf** (Rossell et al., 2014) and **BAS** (Clyde, 2017). These were further compared taking into account relevant aspects like flexibility on prior specification, types of summaries provided, and even computational skills. The main conclusion was that, despite the connections, there are important differences (e.g., on prior specifications or implemented summaries) that will dictate the choice of package (for more details see Forte et al., 2018).

Another package worth mentioning is **BMA** (Raftery et al., 2015). This is one of the most downloaded packages for variable selection in CRAN. **BMA** performs variable selection based on the (BIC) Bayesian Information Criterion (which was developed as an asymptotic approximation to Bayes factors).

Basic formulae

The problems considered in **BayesVarSel** concern Gaussian linear models. Consider a response variable $\boldsymbol{y}$, size n, assumed to follow the linear model (the subindex F refers to the *full* model)

$$M_F : \boldsymbol{y} = \boldsymbol{X}_0\boldsymbol{\alpha} + \boldsymbol{X}\boldsymbol{\beta} + \boldsymbol{\varepsilon}, \;\; \boldsymbol{\varepsilon} \sim N_n(\boldsymbol{0}, \sigma^2 \boldsymbol{I}_n), \tag{1}$$

where the matrices $\boldsymbol{X}_0 : n \times p_0$, $\boldsymbol{X} : n \times p$ and the regression vector coefficients are of conformable dimensions. Suppose you want to test $H_0 : \boldsymbol{\beta} = \boldsymbol{0}$ versus $H_F : \boldsymbol{\beta} \neq \boldsymbol{0}$, that is, to decide whether the regressors in $\boldsymbol{X}$ actually explain the response. This problem is equivalent to the model choice (or model selection) problem with competing models M_F and

$$M_0 : \boldsymbol{y} = \boldsymbol{X}_0\boldsymbol{\alpha} + \boldsymbol{\varepsilon}, \;\; \boldsymbol{\varepsilon} \sim N_n(\boldsymbol{0}, \sigma^2 \boldsymbol{I}_n), \tag{2}$$

and we will refer to models or hypotheses indistinctly.

Posterior probabilities are based on the Bayes factors (see Kass and Raftery, 1995), a measure of evidence provided by **BayesVarSel** when solving testing problems. The Bayes factor of H_F to H_0 is

$$B_{F0} = \frac{m_F(\boldsymbol{y})}{m_0(\boldsymbol{y})},$$

where m_F is the integrated likelihood or prior predictive marginal for the full model:

$$m_F(\boldsymbol{y}) = \int M_F(\boldsymbol{y} \mid \boldsymbol{\alpha}, \boldsymbol{\beta}, \sigma)\, \pi_F(\boldsymbol{\alpha}, \boldsymbol{\beta}, \sigma) d\boldsymbol{\beta}\, d\boldsymbol{\alpha}\, d\sigma,$$

and, similarly:

$$m_0(\boldsymbol{y}) = \int M_0(\boldsymbol{y} \mid \boldsymbol{\alpha}, \sigma)\, \pi_0(\boldsymbol{\alpha}, \sigma) d\boldsymbol{\alpha}\, d\sigma.$$

Above, π_0 and π_F are the prior distributions for the parameters within each model. The assignment of such priors (which we call model selection priors) is quite a delicate issue (see Berger and Pericchi, 2001) and has inspired many important contributions in the literature, particularly from an objective point of view. Of these, the package **BayesVarSel** allows using

many of the most important proposals, which are fully detailed in the appendix. The prior implemented by default is the robust prior by Bayarri et al. (2012), as it can be considered optimal in many senses and is based on a foundational basis.

Posterior probabilities can be obtained as

$$Pr(H_F \mid \boldsymbol{y}) = \frac{B_{F0}Pr(H_F)}{(Pr(H_0) + B_{F0}Pr(H_F))}, \; Pr(H_0 \mid \boldsymbol{y}) = 1 - Pr(H_F \mid \boldsymbol{y}),$$

where $Pr(H_F)$ is the probability, a priori, that hypothesis H_F is true.

Similar formulas can be obtained when more than two hypotheses, say $H_1, \ldots, H_N$, are tested. In this case

$$Pr(H_i \mid \boldsymbol{y}) = \frac{B_{i0}(\boldsymbol{y})Pr(H_i)}{\sum_{j=1}^{N} B_{j0}(\boldsymbol{y})Pr(H_j)}, \; i = 1, \ldots, N, \tag{3}$$

which is the posterior distribution over the model space (the set that contains all competing models). For simplicitly, the formula in (3) has been expressed, without any loss of generality, using Bayes factors to the null model but the same results would be obtained by fixing any other model. The default definition for $Pr(H_i)$ in testing problems is to use a constant prior, which assigns the same probability to all models, that is, $Pr(H_i) = 1/N$.

For instance, within the model

$$M_3 : \boldsymbol{y} = \alpha \mathbf{1}_n + \beta_1 \boldsymbol{x}_1 + \beta_2 \boldsymbol{x}_2 + \boldsymbol{\varepsilon},$$

we cannot test the hypotheses $H_1 : \beta_1 = 0, \beta_2 \neq 0$, $H_2 : \beta_1 \neq 0, \beta_2 = 0$, $H_3 : \beta_1 \neq 0, \beta_2 \neq 0$ since neither M_1 (the model defined by H_1) nor M_2 are nested in the rest. Nevertheless, it is perfectly possible to test the problem with the four hypotheses H_1, H_2, H_3 (as just defined) plus $H_0 : \beta_1 = 0, \beta_2 = 0$, but of course H_0 *must* be, a priori, a plausible hypothesis. In this last case H_0 would take the role of null model.

Hypotheses do not have to be necessarily of the type $\boldsymbol{\beta} = \mathbf{0}$ and, if testable (see Ravishanker and Dey, 2002, for a proper definition), any linear combination of the type $\boldsymbol{C}^t\boldsymbol{\beta} = \mathbf{0}$ can be considered a hypothesis. For instance one can be interested in testing $\beta_1 + \beta_2 = 0$. In Bayarri and García-Donato (2007) it was formally shown that these hypotheses can be, through reparameterizations, reduced to hypotheses like $\boldsymbol{\beta} = \mathbf{0}$. In next section we show examples of how to solve these testing problems in **BayesVarSel** .

Variable selection is a multiple testing problem but is traditionally presented with convenient specific notation that uses a p dimensional binary vector $\boldsymbol{\gamma} = (\gamma_1, \ldots, \gamma_p)$ to identify the models. Consider the full model in (1), and suppose that $\boldsymbol{X}_0$ contains fixed covariates that are believed to be sure in the true model (by default $\boldsymbol{X}_0 = \mathbf{1}_n$ that would make the intercept present in all the models). Then each $\gamma \in \{0,1\}^p$ defines a hypothesis H_γ stating which β's (those with $\gamma_i = 0$) corresponding to each of the columns in $\boldsymbol{X}$ are zero. Then, the model associated with H_γ is

$$M_\gamma : \boldsymbol{y} = \boldsymbol{X}_0\boldsymbol{\alpha} + \boldsymbol{X}_\gamma\boldsymbol{\beta}_\gamma + \boldsymbol{\varepsilon}, \;\; \boldsymbol{\varepsilon} \sim N_n(\mathbf{0}, \sigma^2\boldsymbol{I}_n), \tag{4}$$

where $\boldsymbol{X}_\gamma$ is the matrix with the columns in $\boldsymbol{X}$ corresponding to the ones in $\boldsymbol{\gamma}$. Notice that $\boldsymbol{X}_\gamma$ is a $n \times p_\gamma$ matrix where p_γ is the number of 1's in $\boldsymbol{\gamma}$.

Clearly, in this variable selection problem there are 2^p hypotheses or models and the null model is (2) corresponding to $\boldsymbol{\gamma} = \mathbf{0}$.

A particularity of variable selection is that it is affected by multiplicity issues. This is because, and specially for moderate to large p, the possibility of a model showing spurious evidence is high (just because many hypotheses are considered simultaneously). As concluded in Scott and Berger (2006) multiplicity must be controlled with the prior probabilities $Pr(H_\gamma)$ and the constant prior does not control for multiplicity. Instead, these authors propose using

$$Pr(H_\gamma) = ((p+1)\binom{p}{p_\gamma})^{-1}. \tag{5}$$

The assignment above states that models of the same dimension (the dimension of M_γ is $p_\gamma + p_0$) should have the same probability which must be inversely proportional to the number of models of that dimension. In the sequel we refer to this prior as the ScottBerger prior.

Both the ScottBerger prior and the Constant prior for $Pr(H_\gamma)$ are particular cases of the very flexible prior

$$Pr(M_\gamma \mid \theta) = \theta^{p_\gamma}(1-\theta)^{p-p_\gamma}, \tag{6}$$

where the hyperparameter $\theta \in (0,1)$ has the interpretation of the common probability that a given variable is included (independently of all others).

The Constant prior corresponds to $\theta = 1/2$ while the ScottBerger to $\theta \sim \text{Unif}(0,1)$. Ley and Steel (2009) study priors for θ of the type

$$\theta \sim Beta(\theta \mid 1, b). \tag{7}$$

They argue that, on many occasions the user has, a priori, some information regarding the number of covariates (among the p initially considered) that are expected to explain the response, say $w^\star$. As they explain, this information can be translated into the analysis assigning in (7) $b = (p - w^\star)/w^\star$. The resulting prior specification has the property that the expected number of covariates is precisely $w^\star$.

Straightforward algebra shows that assuming (7) into (6) is equivalent to (integrating out θ)

$$Pr(M_\gamma \mid b) \propto \Gamma(p_\gamma + 1)\Gamma(p - p_\gamma + b). \tag{8}$$

The prior over the model space implemented by default in **BayesVarSel** is the ScottBerger prior.

Hypothesis testing with BayesVarSel

Tests are solved in **BayesVarSel** with `Btest` which, in its default usage, only depends on two arguments: `models` a named list of `formula`-type objects defining the models compared and `data` the data frame with the data.

The prior probabilities assigned to the hypotheses is constant, that is, $Pr(H_i) = 1/N$. This default behavior can be modified specifying `prior.models = "User"` jointly with the argument `priorprobs` that must contain a named list (with names as specified in the main argument `models`) with the prior probabilities to be used for each hypotheses.

Suppose that in the last example of first subsection, we establish a priori that the simpler model is twice as likely as the other two. This can be specified as:

```
> Btest(models = c(H0 = nullmodel, H1 = fullmodel, H2 = reducedmodel),
+  data = savings, prior.models = "User",
+  priorprobs = c(H0 = 1/2, H1 = 1/4, H2 = 1/4))
---------
Bayes factors (expressed in relation to H0)
  H0.to.H0   H1.to.H0   H2.to.H0
 1.0000000 21.4600656  0.7017864
---------
Posterior probabilities:
   H0    H1    H2
0.083 0.888 0.029
```

Notice that the Bayes factor remains the same, and the change is in posterior probabilities.

`Btest` tries to identify the null model (i.e. the model nested in all the others) using the names of the variables. If such a model does not exist, the execution of the function stops with an error message. Nevertheless, there are important situations where the simplest (null) hypothesis is defined through linear restrictions (sometimes known as 'testing a subspace') making it very difficult to determine its existence just using the names. In this situation, the user must provide the name of the simplest model in the argument `null.model`.

To illustrate this case, consider for instance the extension of the savings example in Faraway (2002), page 32 where $H_{eqp} : \beta_{pop15} = \beta_{pop75}$ is tested against the full alternative.

This null hypothesis states that the effect on personal savings, sr, of both segments of populations is similar. The model under H_{eqp} can be specified as:

```
> equalpopmodel <- sr ~ I(pop15 + pop75) + dpi + ddpi
```

but the command

```
> Btest(models = c(Heqp = equalpopmodel, H1 = fullmodel), data = savings)
```

produces an error, although it is clear that H_{eqp} is nested in H_1. To overcome this error, the user must make the name of the null model explicit. In our example:

```
> Btest(models = c(Heqp = equalpopmodel, H1 = fullmodel), data = savings,
+  null.model = "Heqp")
Bayes factors (expressed in relation to Heqp)
Heqp.to.Heqp   H1.to.Heqp
  1.0000000    0.3336251
---------
Posterior probabilities:
Heqp   H1
0.75 0.25
```

Still, the code would produce an error if it detects that the model defined as the null model does not have the largest sum of squared errors or if it is not of a smaller dimension than all the others.

Variable selection with BayesVarSel

The number of entertained hypotheses in a variable selection problem, 2^p, can range from a few to an extremely large number. This makes it necessary to program specific tools to solve the multiple testing problem in variable selection problems. **BayesVarSel** provides two different functions for variable selection

- Bvs performs exhaustive enumeration of hypotheses and hence the size of problems must be small or moderate (say $p \leq 25$),
- GibbsBvs simulates from the posterior distribution over the model space using a Gibbs sampling scheme (intended to be used for large problems, with $p > 25$).

Except for a few arguments that are specific to the algorithm implemented (e.g., the number of cores in Bvs or the number of simulations in GibbsBvs) the usage of the two functions is very similar. We describe the common use in the first of the following subsections and the function-specific arguments in the second.

Bvs and GibbsBvs return objects of class "Bvs" which are a list with relevant information about the posterior distribution. For these objects, **BayesVarSel** provides a number of functions, based on the tradition of model selection methods, to summarize the corresponding posterior distribution (e.g., what is the hypothesis most probable a posteriori) and for its posterior usage (e.g., to obtain predictions or model averaged estimates). These capabilities are described in next two sections.

For illustrative purposes we use the following datasets:

UScrime data. The US crime data set was first studied by Ehrlich (1973) and is available from the package MASS (Venables and Ripley, 2002). This data set has a total of $n = 47$ observations (corresponding to states in the US) of $p = 15$ potential covariates aimed at explaining the rate of crimes in a particular category per head of population (labelled y in the data).

SDM data. This dataset has a total of $p = 67$ potential drivers for the annual GDP growth per capita between 1960 and 1996 for $n = 88$ countries (response variable labelled y in the data). This data set was initially considered by Sala-I-Martin et al. (2004) and revisited by Ley and Steel (2007).

Common arguments

The customary arguments in `Bvs` and `GibbsBvs` are `formula`, with a definition of the most complex model considered (the full model in (1)) and `data` (a data frame with the data). The default execution setting corresponds to a problem where the null model (2) contains just the intercept (i.e. $\boldsymbol{X}_0 = \mathbf{1}_n$) and prior probabilities for models are defined as in (5).

A different null model can be specified with the optional argument `null.model`, that must take the value of the formula of the the null model (a model that should be nested in the full model).

Suppose for example that in the `UScrime` dataset and apart from the constant, theory suggests that the covariate `Ed` must be used to explain the dependent variable. To consider these conditions we execute the command

```
> crime.Edfix <- Bvs(formula = y ~ ., data = UScrime,
+  null.model = y ~ Ed)
Info. . . .
Most complex model has 16 covariates
From those 2 are fixed and we should select from the remaining 14
M, So, Po1, Po2, LF, M.F, Pop, NW, U1, U2, GDP, Ineq, Prob, Time
The problem has a total of 16384 competing models
Of these, the  10 most probable (a posteriori) are kept
Working on the problem...please wait.
> crime.Edfix
```

During the execution (which takes about 0.22 seconds in a standard laptop) the function informs which variables take part of the selection process. The number of these defines p which in this problem is $p = 14$ (and the model space has 2^{14} models). In what follows, and unless otherwise stated we do not reproduce this informative output to save space.

The assignment of priors probabilities, $Pr(H_i)$, is regulated with `prior.models`, an argument that by default takes the value `'"ScottBerger"'` corresponding to the proposal in (5). Other options for this argument are `'"Constant"'`, which stands for $Pr(H_i) = 1/2^p$, and the more flexible value, `'"User"'`, under which the user must specify the prior probabilities with the extra argument `priorprobs`.

The argument `priorprobs` is a $p + 1$ numeric vector, which in its i-th position defines the probability of a model of dimension $p_0 + i - 1$ (these probabilities can be specified except for the normalizing constant).

Suppose that in the `UScrime` dataset with null model just the intercept, we want to specify the prior in eq (6) with $\theta = 1/4$, this can be done as (notice that here $p = 15$)

```
> theta <- 1/4; pgamma <- 0:15
> crime.thQ <- Bvs(formula = y ~ ., data = UScrime, prior.models = "User",
+  priorprobs = theta^pgamma*(1-theta)^(15-pgamma))
> crime.thQ
```

In variable selection problems it is quite standard to have the situation where the number of covariates is large (say larger than 30) preventing the exhaustive enumeration of all the competing models. The `SDM` dataset is an example of this situation with $p = 67$. In these contexts, the posterior distribution can be explored using the function `GibbsBvs`. To illustrate the elicitation of prior probabilities as well, suppose that the number of expected covariates to be present in the true model is $w^\star = 7$. This situation is considered in Ley and Steel (2009) and can be implemented as (see (8))

```
> set.seed(1234)
> data(SDM)
> p <- ncol(SDM) - 1; wstar <- 7;
> b <- (p-wstar)/wstar; pgamma <- 0:p
> growth.wstar7 <- GibbsBvs(formula = y ~ ., data = SDM, prior.models = "User",
+  priorprobs = gamma(pgamma+1)*gamma(p-pgamma+b), n.iter = 10000, n.thin = 1,
+  time.test = FALSE)
> growth.wstar7
```

The above code took 18 seconds to run.

One last common argument to `Bvs` and `GibbsBvs` is `time.test`. If it is set to `TRUE` and the problem is of moderate size ($p \geq 18$ in `Bvs` and $p \geq 21$ in `GibbsBvs`), an estimation of

computational time is calculated and the user is asked about the possibility of not executing the command.

Specific arguments

In `Bvs` The algorithm implemented in `Bvs` is exact in the sense that the information collected about the posterior distribution takes into account *all* competing models as these are all computed. The logical parameter `parallel`, if set to `TRUE`, distributes the task among the number of processors specified in the argument `n.nodes`. By default the parameter `parallel` is set to `FALSE`. To save computational time and memory it is quite appropriate to keep only a moderate number of the best (most probable a posteriori) models. This number can be specified with the argument `n.keep` which must be an integer number between 1 (only the most probable model is kept) and 2^p (a full ordering of models is kept) (or 2^p divided by `n.nodes` in the case of parallel computation). The default value for `n.keep` is 10.

The argument `n.keep` is not of great importance to analyze the posterior distribution over the model space. Nevertheless, it has a more relevant effect if model averaging estimates or predictions are to be obtained (see the corresponding section on page 166) since, as **BayesVarSel** is designed, only the `n.keep` retained models are used for these tasks.

In `GibbsBvs` The algorithm in `GibbsBvs` samples models from the posterior over the model space and this is done using a simple (yet very efficient) Gibbs sampling scheme introduced in George and McCulloch (1997), later studied in Garcia-Donato and Martinez-Beneito (2013) in the context of large model spaces. The type of default arguments that can be specified in `GibbsBvs` are typical in any Monte Carlo Markov Chain scheme (as usual the default values are given in the assignment)

- `init.model = "Full"` The model at which the simulation process starts. Options include `'"Null"'` (the model only with the covariates specified in `null.model`), `'"Full"'` (the model defined by `formula`), and `'"Random"'` (a randomly selected model) a vector with p zeros and ones defining a model.
- `n.burnin = 500` Length of burn in, i.e. the number of iterations to discard at the start of the simulation process.
- `n.iter = 10000` The total number of iterations performed after the burn in process.
- `n.thin = 1` Thinning rate that must be a positive integer. Set `n.thin > 1` to save memory and computation time if `n.iter` is large.
- `seed = runif(1,0,16091956)` A seed to initialize the random number generator.

Notice that the number of total iterations is `n.burnin+n.iter` but the number of models that are used to collect information from the posterior is, approximately, `n.iter/n.thin`.

Summaries of the posterior distribution

In this section we describe the tools implemented in **BayesVarSel** conceived to summarize (in the tradition of the relevant model selection literature) the posterior distribution over the model space. In R, this corresponds to describing methods to explore the content of objects of class `"Bvs"`.

Printing a `"Bvs"` object created with `Bvs` shows the best 10 models with their associated probability (see examples in first section). If the object was built with `GibbsBvs` the 10 most probable a posteriori models among the visited ones are shown. Though, in this case, posterior probabilities of models are not provided, as these are unknown.

The rest of the summaries are very similar, independent of the routine used to create it, but recall that if the object was obtained with `Bvs` (likely because p is small or moderate) the given measures here explained are *exact*. If instead `GibbsBvs` was used, the reported measures are approximations of the exact ones (that likely cannot be computed due to

the huge size of the model space). In **BayesVarSel** these approximations are based on the frequency of visits of M_γ as an estimator since, as studied in Garcia-Donato and Martinez-Beneito (2013), these provide quite accurate summaries.

The HPM is returned when an object of class `"Bvs"` is summarized (via `summary`) jointly with the inclusion probabilities for each competing variable, $Pr(x_i \mid \boldsymbol{y})$. These are the sum of the posterior probabilities of models containing that covariate and provide evidence about the individual importance of each explanatory variable. The model defined by those variables with an inclusion probability greater than 0.5 is called a Median Probability Model (MPM), which is also included in the summary. Barbieri and Berger (2004) show that, under general conditions, if a single model has to be utilized with predictive purposes, the MPM is optimal.

For instance, if we summarize the object `crime.Edfix`[1] of the fourth section, we obtain:

```
> summary(crime.Edfix)

Inclusion Probabilities:
     Incl.prob. HPM MPM
M        0.6806       *
So       0.2386
Po1      0.8489   *   *
Po2      0.3663
LF       0.2209
M.F      0.3184
Pop      0.2652
NW       0.2268
U1       0.2935
U2       0.4765
GDP      0.3204
Ineq     0.9924   *   *
Prob     0.6174       *
Time     0.2434
```

Inclusion probabilities implicitly answer a testing problem with two hypotheses, namely if the variable should or should not be included. This is made sensible (and practical) using the categorization in Table 1 to interpret them. With this guide, the importance of `Ineq` is decisive followed by `Po1`, `M` and `Prob` with a substantial evidence in favour of their relevance in explaining the response.

Graphical summaries and jointness The main graphical support in **BayesVarSel** is contained in the S3 function `plot`[2] which depends on `x` (an object of class `"Bvs"`) and the argument `option` specifying the type of plot to be produced:

- `option="joint"` A matrix plot with the joint inclusion probabilities, $Pr(x_i, x_j \mid \boldsymbol{y})$ (marginal inclusion probabilities in the diagonal).
- `option="conditional"` A matrix plot with $Pr(x_i \mid x_j, \boldsymbol{y})$, the conditional inclusion probabilities (ones in the diagonal).
- `option="not"` A matrix plot with $Pr(x_i \mid \text{Not } x_j, \boldsymbol{y})$, the conditional inclusion probabilities (zeroes in the diagonal).
- `option="dimension"` A bar plot representation of the posterior distribution of the dimension of the true model (number of variables, ranging from p_0 to $p_0 + p$).
- `option="trace"` A trace plot with the evolution of the inclusion probabilities with the iterations in Gibbs sampling (useful to check convergence; only available for `GibbsBvs`).

[1] Notice that variable `Ed` is not on the list as it was assumed to be fixed.

[2] Depending on p, this function may produce very large plots causing margin-type errors to be thrown if either the device or the active graphical window are too small. In these cases the user must consider ways of enlarging the graphical output recipient.

The first three options above are basic measures describing aspects of the joint effect of two given variables, x_i and x_j, and can be understood as natural extensions of the marginal inclusion probabilities. In Figure 1, we have reproduced the first three plots (from left to right) obtained with the following lines of code:

```
> mj <- plot(crime.Edfix, option = "joint")
> mc <- plot(crime.Edfix, option = "conditional")
> mn <- plot(crime.Edfix, option = "not")
```

Apart from the plot, these functions return (invisibly) the matrix represented for futher study. For the conditional probabilities (`conditional` and `not`) the variables in the row are the conditioning variables (e.g., in `mc` above, the position (i, j) is the inclusion probability of variable in j-th column conditional on the variable in i-th row).

Within these matrices, the most interesting results correspond to variations from the marginal inclusion probabilities (represented in the top of the plots as a separate row for reference). Our experience suggests that the most valuable of these is `option="not"`, as it can reveal key details about the relations between variables in relation with the response. For instance, take that plot in Figure 1 (plot on the left of the second row) and observe that while variable `Po2` barely has any effect on `y` (crime), it becomes relevant if `Po1` is removed. This is the probability $Pr(Po2 \mid \text{Not } Po1, \boldsymbol{y})$ with value

```
> mn["Not.Po1", "Po2"]
[1] 0.9996444
```

which, as we observed in the graph, is substantially large compared with the inclusion probability of `Po2`

```
> crime.Edfix$inclprob["Po2"]
      Po2
0.3662537
```

Similarly, we observe that `Po1` is of even more importance if `Po2` is not considered as a possible explanatory variable. All this implies that, in relation with the dependent variable, both variables contain similar information and one can act as proxy for the other.

We can further investigate this idea of a relationship between two covariates with respect to the response using the *jointness* measures proposed by Ley and Steel (2007). These are available using a function `Jointness` that depends on two arguments: `x`, an object of class `"Bvs"` and `covariates`, a character vector indicating which pair of covariates we are interested in. The default `covariates="All"` will output the matrices with the jointness measurement for every pair of covariates. In particular, three jointness measures relative to two covariates are reported by this function: i) the joint inclusion probability, ii) the ratio between the joint inclusion probability and the probability of including at least one of them, and iii) the ratio between the joint inclusion probability and the probability of including one of them alone.

For instance:

```
> joint_measures <- Jointness(crime.Edfix, covariates = c("Po1" , "Po2"))
> joint_measures
---------
The joint inclusion probability for Po1 and Po2 is:  0.22
---------
The ratio between the probability of including both covariates and the probability of
 including at least one of then is: 0.22
---------
The probability of including both covariates together is 0.27 times the probability of
 including one of them alone
```

Alternatively, the single numbers can be accessed using:

```
> joint_measures$prob_joint
[1] 0.215181
> joint_measures$joint_LS1
[1] 0.2151926
> joint_measures$joint_LS2
[1] 0.274198
```

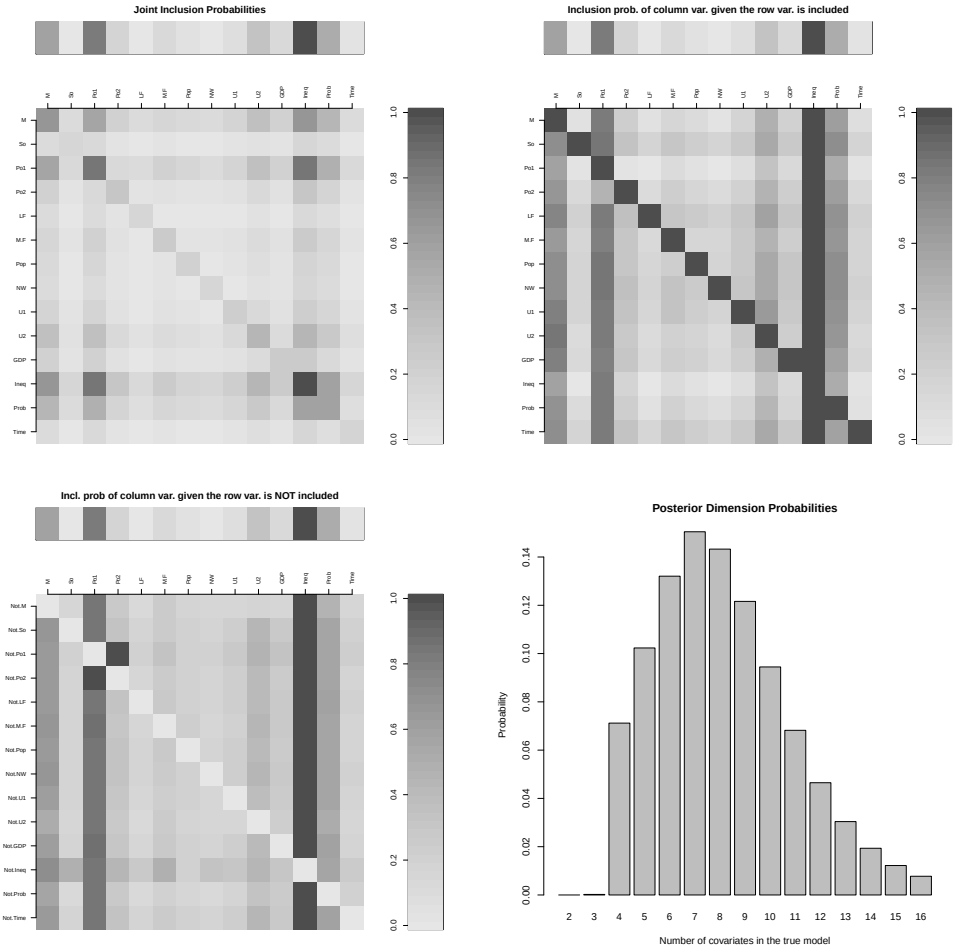


Figure 1: Plots corresponding to the four possible values of the argument `option` in `plot` over the object `crime.Edfix` of section four. From left to right: `joint`, `conditional`, `not`, and `dimension`.

With these results we must conclude that it is unlikely that both variables, `Po1` and `Po2`, are to be included together in the true model.

Finally, within `plot`, the assignment `option="dimension"` produces a plot that speaks about the complexity of the true model in terms of the number of covariates that it contains. The last plot in Figure 1 is the output of executing:

```
> plot(crime.Edfix, option = "dimension")
```

From this plot we conclude that the number of covariates is about 7 but with a high variability. The exact values of this posterior distribution are in the component `postprobdim` of the `Bvs` object.

The last possibility for the `plot` method corresponds to `option="trace"` which produces a trace plot of the inclusion probabilities. The resulting graph is useful to asses the convergence of the posterior inclusion probabilities when Gibbs sampling was used and we can informally check if the number of iterations is enough. For instance, Figure 2 shows how the posterior inclusion probabilities in this setting stabilize after roughly 8000 iterations.

```
> plot(growth.wstar7, option = "trace")
```

Model averaged estimations and predictions

In a variable selection problem it is explicitly recognized that there is uncertainty regarding which variables make up the true model. Obviously, this uncertainty should be propagated in the inferential process (as opposed to inferences using just one model) to produce more reliable and accurate estimations and predictions. These type of procedures are normally

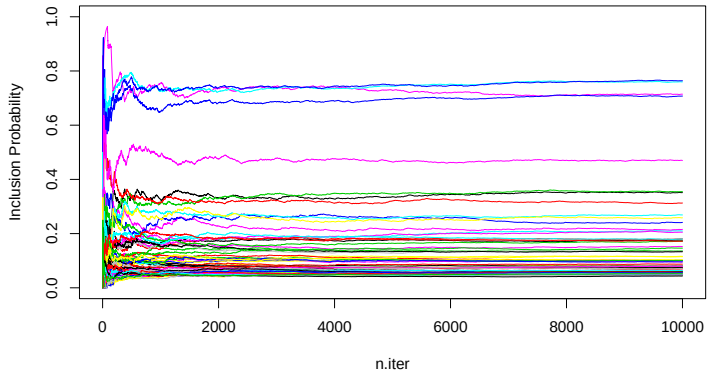


Figure 2: Plot corresponding to the "trace" `option` in `plot` over the object `growth.wstar7`.

called model averaging (Madigan and Raftery, 1994; Raftery, 1995) and are performed once the model selection exercise is performed (that is, the posterior probabilities have been already obtained). In **BayesVarSel** these inferences can be obtained acting over objects of class `"Bvs"`.

Suppose that Λ is a quantity of interest and that under model M_γ it has a posterior distribution $\pi^N(\Lambda \mid \boldsymbol{y}, M_\gamma)$ with respect to certain non-informative prior π^N_γ. Then, we can average over all entertained models using the posterior probabilities in (3) as weights to obtain

$$f(\Lambda \mid \boldsymbol{y}) = \sum_\gamma \pi^N(\Lambda \mid \boldsymbol{y}, M_\gamma)\, Pr(M_\gamma \mid \boldsymbol{y}). \tag{9}$$

In **BayesVarSel** for π^N_γ we use the reference prior developed in Berger and Bernardo (1992) and further studied in Berger et al. (2009). This is an objective prior with very good theoretical properties. The formulas for the posterior distribution with a fixed model are known (Bernardo and Smith, 1994). These priors are different from the *model selection* priors used to compute the Bayes factors (see the second section), but, as shown in Consonni and Deldossi (2016), the posterior distributions approximately coincide, and so $f(\Lambda \mid \boldsymbol{y})$ basically can be interpreted as the posterior distribution of Λ.

There are two different quantities, Λ, that are of main interest in variable selection problems. First is a regression parameter β_i and second is a future observation $y^\star$ associated with known values of the covariates $\boldsymbol{x}^\star$. In what follows we refer to each of these problems as (model averaged) estimation and prediction, respectively, to which we devote the next subsections.

Estimation

Inclusion probabilities $Pr(x_i \mid \boldsymbol{y})$ can be roughly interpreted as the probability that β_i is different from zero. Nevertheless, it does not say anything about the magnitude of the coefficient β_i nor anything about its sign.

Such type of information can be obtained from the distribution in (9) which in the case of $\Lambda \equiv (\boldsymbol{\alpha}, \boldsymbol{\beta})$ is

$$f(\boldsymbol{\alpha}, \boldsymbol{\beta} \mid \boldsymbol{y}) = \sum_\gamma St_{p_\gamma + p_0}((\boldsymbol{\alpha}, \boldsymbol{\beta}_\gamma) \mid (\hat{\boldsymbol{\alpha}}, \hat{\boldsymbol{\beta}}_\gamma), (\boldsymbol{Z}_\gamma^\top \boldsymbol{Z}_\gamma)^{-1} \frac{SSE_\gamma}{n - p_\gamma - p_0}, n - p_\gamma - p_0)\, Pr(M_\gamma \mid \boldsymbol{y}), \tag{10}$$

where $\hat{\boldsymbol{\alpha}}, \hat{\boldsymbol{\beta}}_\gamma$ is the maximum likelihood estimator under M_γ (see (4)), $\boldsymbol{Z}_\gamma = (\boldsymbol{X}_0, \boldsymbol{X}_\gamma)$ and SSE_γ is the sum of squared errors in M_γ. St above refers to the multivariate student distribution:

$$St_k(\boldsymbol{x} \mid \boldsymbol{\mu}, \boldsymbol{\Sigma}, df) \propto \left(1 + \frac{1}{df}(\boldsymbol{x} - \boldsymbol{\mu})^\top \boldsymbol{\Sigma}^{-1} (\boldsymbol{x} - \boldsymbol{\mu})\right)^{-(df+k)/2}.$$

In **BayesVarSel** the whole model averaged distribution in (10) is provided in the form of a random sample through the function `BMAcoeff` which depends on two arguments: `x`, a `"Bvs"` object and `n.sim`, the number of observations to be simulated (with default value of 10000). The returned object is an object of class `"bma.coeffs"` which is a column-named `matrix` with `n.sim` rows (one per each simulation) and $p + p_0$ columns (one per each regression parameter). The way that `BMAcoeff` works depends on whether the object was created with `Bvs` or with `GibbsBvs`. This is further explained below.

If the `Bvs` object was generated with `Bvs` In this case the models over which the average is performed are the `n.keep` (previously specified) best models. Hence, if `n.keep` equals 2^p then all competing models are used; while, if `n.keep`$<2^p$, only a proportion of them are used and posterior probabilities are re-normalized to sum to one.

On many occasions where estimations are required, the default value of `n.keep` (which we recall is 10) is small and should be increased. Ideally 2^p should be used but, as noticed by Raftery et al. (1997) this is normally unfeasible and commonly it is sufficient to average over a reduced set of good models that accumulate a reasonable posterior mass. This set is what Raftery et al. (1997) call the "Occam's window." The function `BMAcoeff` informs about the total probability accumulated in the models that are used.

For illustrative purposes, let us retake the `UScrime` dataset and, in particular, the example in first section in which, apart from the constant, the variable `Ed` was assumed as fixed. The total number of models is $2^{14} = 16384$ and we execute `Bvs` again but now with `n.keep`=2000:

```
> crime.Edfix <- Bvs(formula = y ~ ., data = UScrime,
+ null.model = y ~ Ed, n.keep = 2000)
> crime.Edfix
```

(This takes about 1.9 seconds). The `crime.Edfix` object contains identical information as the one previously created with the same name, except for the models retained, which in this case are the best 2000. These models accumulate a probability[3] of roughly 0.90, which seems quite reasonable to derive the estimates. We do so executing the second command of the following script (the seed is fixed for the sake of reproducibility).

```
> set.seed(1234)
> bma.crime.Edfix <- BMAcoeff(crime.Edfix)
Simulations obtained using the best 2000 models
that accumulate 0.9 of the total posterior probability
```

The distribution in (10) and hence the simulations obtained can be highly multimodal and providing default summaries of it (like the mean or standard deviation) is potentially misleading. For a first exploration of the model averaged distribution, **BayesVarSel** comes armed with a plotting function, `histBMA`, that produces a histogram-like representation borrowing ideas from Scott and Berger (2006) and placing a bar at zero with height proportional to the number of zeros obtained in the simulation.

The function `histBMA` depends on several arguments:

- `x` An object of class `bma.coeffs`.
- `covariate` A string specifying the name of an explanatory variable whose accompanying coefficient is to be represented. This must be the name of one of the columns in `x`.
- `n.breaks` The number of equal width bars for the histogram. Default value is 100.
- `text` If set to `TRUE` (default value) the frequency of zeroes is added at the top of the bar at zero.
- `gray.0` A numeric value between 0 and 1 that specifies the darkness, in a gray scale (0 is white and 1 is black) of the bar at zero. Default value is 0.6.
- `gray.no0` A numeric value between 0 and 1 that specifies the darkness, in a gray scale (0 is white and 1 is black) of the bars different from zero. Default value is 0.8.

[3] `sum(crime.Edfix$modelsprob$prob)`

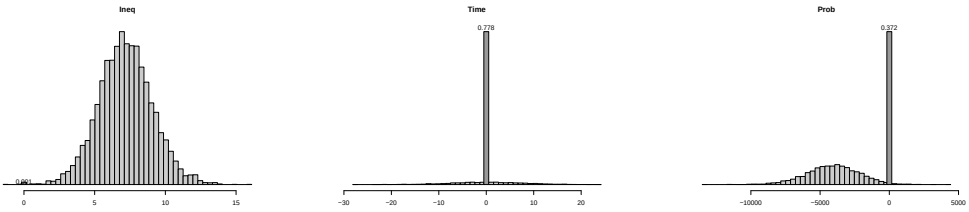


Figure 3: Representation provided by the function `histBMA` of the Model averaged posterior distributions of β_{Ineq}, $\beta_{Time}$and β_{Prob} for the `UScrime` dataset with a constant and `Ed` considered as fixed in the variable selection exercise.

For illustrative purposes let us examine the distributions of β_{Ineq} (inclusion probability 0.99); β_{Time} (0.24) and β_{Prob} (0.62) using `histBMA`

```
> histBMA(bma.crime.Edfix, covariate = "Ineq", n.breaks = 50)
> histBMA(bma.crime.Edfix, covariate = "Time", n.breaks = 50)
> histBMA(bma.crime.Edfix, covariate = "Prob", n.breaks = 50)
```

The plots obtained are reproduced in Figure 3. We can see that `Ineq` has a positive effect. This distribution is unimodal so there is no drawback to summarizing the effect of the response `Ineq` over `y` (crime) using say, the mean or quantiles, For example:

```
> quantile(bma.crime.Edfix[, "Ineq"], probs = c(0.05, 0.5, 0.95))
       5%       50%       95%
 4.075685  7.150184 10.326606
```

This implies an estimated effect of 7.2 with a 90% credible interval [4.1,10.3]. The situation of `Time` is clear and its estimated effect is basically null (in agreement with a low inclusion probability).

Much more problematic is reporting estimates of the effect of `Prob` with a highly polarized estimated effect being either very negative (around -4100) or zero (again in agreement with its inconclusive inclusion probabilty of 0.62). Notice that, in this case, the mean (approximately -2500) should not be used as a sensible estimation of the parameter β_{Prob}.

If the `Bvs` object was generated with `GibbsBvs` In this case, the average in (10) is performed over the `n.iter` (an argument previously defined) models sampled in the MCMC scheme. Theoretically this corresponds to sampling over the whole distribution (all models are considered) and leads to the approximate method pointed out in Raftery et al. (1997). All previous considerations regarding the difficult nature of the underlying distribution apply here.

Let us consider again the `SDM` dataset in which analysis we created `growth.wstar7` in the fourth section. Suppose we are interested in the effect of the variable `P60` on the response `GDP`. The `summary` method informs that this variable has an inclusion probability of 0.77.

```
> set.seed(1234)
> bma.growth.wstar7 <- BMAcoeff(growth.wstar7)
Simulations obtained using the  10000  sampled models.
Their frequencies are taken as the true posterior probabilities
> histBMA(bma.growth.wstar7, covariate = "P60", n.breaks=50)
```

The distribution is bimodal (graph not shown here to save space) with modes at zero and 2.8, approximately. Again, it is difficult to provide simple summaries to describe the model averaged behaviour of P60. Nevertheless, it is always possible to answer relevant questions such as: what is the probability that the effect of `P60` over savings is greater than one?

```
> mean(bma.growth.wstar7[, "P60"] > 1)
[1] 0.7511
```

Prediction

Suppose we want to predict a new observation $y^\star$ with associated values of covariates $(\boldsymbol{x}^\star)^\top \in \mathcal{R}^{p_0+p}$ (in what follows, the product of two vectors corresponds to the usual scalar

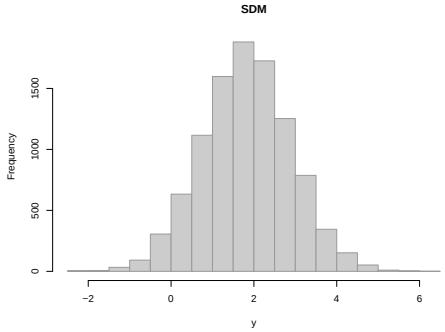


Figure 4: For SDM data and related `Bvs` object `growth.wstar7`, model averaged prediction of the "mean" case (predicting the output associated with the mean of observed covariates)

product). In this case, the distribution (9) adopts the form

$$f(y^\star \mid \boldsymbol{y}, \boldsymbol{x}^\star) = \sum_\gamma St(y^\star \mid \boldsymbol{x}^\star_\gamma\ (\hat{\boldsymbol{\alpha}}, \hat{\boldsymbol{\beta}}_\gamma), \frac{SSE_\gamma}{h_\gamma}, n - p_\gamma - p_0)\ Pr(M_\gamma \mid \boldsymbol{y}), \tag{11}$$

where

$$h_\gamma = 1 - \boldsymbol{x}^\star_\gamma \left((\boldsymbol{x}^\star_\gamma)^\top \boldsymbol{x}^\star_\gamma + \boldsymbol{Z}_\gamma^\top \boldsymbol{Z}_\gamma\right)^{-1} (\boldsymbol{x}^\star_\gamma)^\top.$$

As with estimations, **BayesVarSel** has implemented a method for the S3 function `predict`, designed to simulate a desired number of observations from (11). A main difference with model averaged estimations is that, typically, the above predictive distribution is unimodal.

The `predict` method depends on `object`, an object of class `"Bvs"`, `newdata`, a data frame with the values of the covariates (the intercept, if needed, is automatically added) and `n.sim` the number of observations to be simulated. The considerations about the calculation of the probabilities described in the previous section for the `Bvs` object depending on the type of function originally used apply here.

The `predict` method returns a matrix with `n.sim` rows (one per each simulated observation) and with the number of columns being the number of cases (rows) in the data frame `newdata`.

For illustrative purposes, consider the `"Bvs"` object named `growth.wstar7` from the analysis of the `SDM` dataset. Simulations from the predictive distribution (11) associated with values of the covariates fixed at their means can be obtained with the following code. Here, a histogram is produced (see Figure 4) as a graphical approximation of the underlying distribution.

```
> set.seed(1234)
> pred.growth.wstar7 <- predict(object = growth.wstar7,
+  newdata = data.frame(t(colMeans(SDM))))
> hist(pred.growth.wstar7[, 1], main = "SDM",
+  border = gray(0.6), col = gray(0.8), xlab = "y")
```

Future work

The first version of **BayesVarSel** was released on December 2012 with the main idea of making available the C code programmed for the work Garcia-Donato and Martinez-Beneito (2013) to solve exactly moderate to large variable selection problems. Since then, seven versions have followed with new abilities that make up the complete toolbox that we have described in this paper.

Nevertheless, **BayesVarSel** is an ongoing project that we plan to continue in the future as solid and contrasted methods become available. The emphasis is placed on the prior distribution that should be used, since this is a particularly relevant aspect of model selection/testing problems.

New functionalities that we expect to incorporate in the future are:

- The case where $n < p + p_0$ and possibly $n << p + p_0$,
- specific methods for handling factors,
- heteroscedastic errors,
- other types of error distributions.

Acknowledgments

This paper was supported in part by the Spanish Ministry of Economy and Competitivity under grant MTM2016-77501-P (BAiCORRE). The authors would like to thank the referees for their many valuable comments that helped to greatly improve both, this paper and our package **BayesVarSel**. Also, we would like to thank Jim Berger for his suggestions during the process of building the package and Carlos Vergara for his invaluable help with the use of the versions control system *Git*.

Bibliography

M. M. Barbieri and J. O. Berger. Optimal predictive model selection. *The Annals of Statistics*, 32(3):870–897, 2004. URL https://doi.org/10.1214/009053604000000238. [p164]

M. J. Bayarri and G. García-Donato. Extending conventional priors for testing general hypotheses in linear models. *Biometrika*, 94(1):135–152, 2007. URL https://doi.org/10.1093/biomet/asm014. [p159]

M. J. Bayarri, J. O. Berger, A. Forte, and G. García-Donato. Criteria for Bayesian model choice with application to variable selection. *The Annals of Statistics*, 40:1550–1577, 2012. URL https://doi.org/10.1214/12-AOS1013. [p159, 173]

D. A. Belsley, E. Kuh, and R. E. Welsch. *Regression Diagnostics: Identifying Influential Data and Sources of Collinearity*. John Wiley & Sons, 2005. URL https://doi.org/10.1002/0471725153. [p156]

J. O. Berger. The case for objective Bayesian analysis. *Bayesian Analysis*, 1(3):385–402, 2006. URL https://doi.org/10.1214/06-BA115. [p155]

J. O. Berger and J. M. Bernardo. On the development of the reference prior method. In J. M. Bernardo, editor, *Bayesian Statistics 4*, pages 35–60. London: Oxford University Press, 1992. [p167]

J. O. Berger and L. R. Pericchi. Objective Bayesian Methods for Model Selection: Introduction and Comparison. In *Model Selection*, volume 38 of *Lecture Notes–Monograph Series*, pages 135–207. Institute of Mathematical Statistics, Beachwood, OH, 2001. URL https://doi.org/10.1214/lnms/1215540968. [p158]

J. O. Berger, J. M. Bernardo, and D. Sun. The formal definition of reference priors. *The Annals of Statistics*, 37(2):905–938, 2009. URL https://doi.org/10.1214/07-AOS587. [p167]

J. M. Bernardo and A. F. M. Smith. *Bayesian Theory*. John Wiley & Sons, 1994. [p167]

M. Clyde. *BAS: Bayesian Adaptive Sampling for Bayesian Model Averaging*, 2017. URL http://CRAN.R-project.org/package=BAS. R package version 1.4.7. [p158]

G. Consonni and L. Deldossi. Objective bayesian model discrimination in follow-up experimental designs. *Test*, 25:397–412, 2016. URL https://doi.org/10.1007/s11749-015-0461-3. [p167]

I. Ehrlich. Participation in illegitimate activities: a theoretical and empirical investigation. *Journal of Political Economy*, 81(3):521–567, 1973. [p161]

J. Faraway. *Practical Regression and Anova in R*, 2002. [p156, 160]

J. Faraway. *Functions and Datasets for Books by Julian Faraway*, 2016. URL http://CRAN.R-project.org/package=faraway. R package version 1.0.7. [p156]

C. Fernández, E. Ley, and M. F. Steel. Benchmark priors for Bayesian model averaging. *Journal of Econometrics*, 100:381–427, 2001. URL https://doi.org/10.1016/S0304-4076(00)00076-2. [p174]

A. Forte, G. Garcia-Donato, and M. F. J. Steel. Methods and Tools for Bayesian Variable Selection and Model Averaging in Univariate Linear Regression. *International Statistical Review (to appear)*, 2018. URL https://doi.org/10.1111/insr.12249. [p158]

G. Garcia-Donato and A. Forte. *BayesVarSel: Bayes Factors, Model Choice and Variable Selection In Linear Models*, 2015. URL http://CRAN.R-project.org/package=BayesVarSel. R package version 1.6.1. [p155]

G. Garcia-Donato and M. A. Martinez-Beneito. On Sampling Strategies in Bayesian Variable Selection Problems with Large Model Spaces. *Journal of the American Statistical Association*, 108(501):340–352, 2013. URL https://doi.org/10.1080/01621459.2012.742443. [p163, 164, 170]

E. I. George and R. E. McCulloch. Approaches for Bayesian variable selection. *Statistica Sinica*, 7:339–373, 1997. [p163]

H. Jeffreys. *Theory of Probability*. Oxford University Press, 3rd edition, 1961. [p174]

R. E. Kass and A. E. Raftery. Bayes factors. *Journal of the American Statistical Association*, 90(430):773–795, 1995. URL https://doi.org/10.1080/01621459.1995.10476572. [p156, 158]

R. E. Kass and L. Wasserman. A reference Bayesian test for nested hypotheses and its relationship to the Schwarz criterion. *Journal of the American Statistical Association*, 90(431): 928–934, 1995. URL https://doi.org/10.1080/01621459.1995.10476592. [p174]

P. M. Lee. *Bayesian Statistics. An Introduction. Second Edition.* John Wiley & Sons, 1997. [p155]

E. Ley and M. F. Steel. On the effect of prior assumptions in Bayesian model averaging with applications to growth regression. *Journal of Applied Econometrics*, 24(4):651–674, 2009. URL https://doi.org/10.1002/jae.1057. [p160, 162]

E. Ley and M. F. J. Steel. Jointness in Bayesian variable selection with applications to growth regression. *Journal of Macroeconomics*, 29(3):476 – 493, 2007. URL https://doi.org/10.1016/j.jmacro.2006.12.002. Special Issue on the Empirics of Growth Nonlinearities. [p161, 165]

F. Liang, R. Paulo, G. Molina, M. A. Clyde, and J. O. Berger. Mixtures of g-priors for Bayesian variable selection. *Journal of the American Statistical Association*, 103(481):410–423, 2008. URL https://doi.org/10.1198/016214507000001337. [p174]

D. Madigan and A. E. Raftery. Model selection and accounting for model uncertainty in graphical models using Occam's window. *Journal of the American Statistical Association*, 89: 1535 – 1546, 1994. URL https://doi.org/10.1080/01621459.1994.10476894. [p167]

R. D. Morey, J. N. Rouder, and T. Jamil. *BayesFactor: Computation of Bayes Factors for Common Designs*, 2015. URL http://CRAN.R-project.org/package=BayesFactor. R package version 0.9.11-1. [p158]

A. Raftery, J. Hoeting, C. Volinsky, I. Painter, and K. Y. Yeung. *BMA: Bayesian Model Averaging*, 2015. URL http://CRAN.R-project.org/package=BMA. R package version 3.18.4. [p158]

A. E. Raftery. Bayesian model selection in social research. *Sociological Methodology*, 25:111 – 163, 1995. URL http://doi.org/10.2307/271063. [p167]

A. E. Raftery, D. Madigan, and J. Hoeting. Bayesian model averaging for linear regression models. *Journal of the American Statistical Association*, 92:179–191, 1997. [p168, 169]

N. Ravishanker and D. K. Dey. *A First Course in Linear Model Theory*. Chapman and Hall/CRC, 2002. [p159]

D. Rossell, J. D. Cook, D. Telesca, and P. Roebuck. *Mombf: Moment and Inverse Moment Bayes Factors*, 2014. URL http://CRAN.R-project.org/package=mombf. R package version 1.5.9. [p158]

X. Sala-I-Martin, G. Doppelhofer, and R. I. Miller. Determinants of long-term growth: A Bayesian averaging of classical estimates (BACE) approach. *American Economic Review*, 94 (4):813–835, 2004. URL https://doi.org/10.1257/0002828042002570. [p161]

J. G. Scott and J. O. Berger. An exploration of aspects of Bayesian multiple testing. *Journal of Statistical Planning and Inference*, 136(7):2144–2162, 2006. URL https://doi.org/10.1016/j.jspi.2005.08.031. [p159, 168]

W. N. Venables and B. D. Ripley. *Modern Applied Statistics with S*. Springer-Verlag, New York, 4th edition, 2002. URL http://www.stats.ox.ac.uk/pub/MASS4. ISBN 0-387-95457-0. [p161]

A. Zellner. On assessing prior distributions and Bayesian regression analysis with g-prior distributions. In A. Zellner, editor, *Bayesian Inference and Decision Techniques: Essays in Honor of Bruno De Finetti*, pages 389–399. Edward Elgar Publishing Limited, 1986. [p174]

A. Zellner and A. Siow. Posterior odds ratio for selected regression hypotheses. In J. M. Bernardo, M. H. DeGroot, D. V. Lindley, and A. F. M. Smith, editors, *Bayesian Statistics 1*, pages 585–603. Valencia: University Press, 1980. [p174]

A. Zellner and A. Siow. *Basic Issues in Econometrics*. Chicago: University of Chicago Press, 1984. [p174]

S. Zeugner and M. Feldkircher. Bayesian model averaging employing fixed and flexible priors: The BMS package for R. *Journal of Statistical Software*, 68(4):1–37, 2015. URL https://doi.org/10.18637/jss.v068.i04. [p158]

Gonzalo Garcia-Donato
Universidad de Castilla-La Mancha (Department of Economy and Finance; Instituto de Desarrollo Regional (IDR))
Plaza Universidad 2, Albacete
Spain
gonzalo.garciadonato@uclm.es

Anabel Forte
Universidad de Valencia (Department of Statistics and OR) Calle Dr. Moliner, Burjassot (Valencia)
Spain
anabel.forte@uv.es

Appendix: Model selection priors for parameters within models

A key technical component of Bayes factors and hence of posterior probabilities is the prior distribution for the parameters within each model. That is, the prior $\pi_\gamma(\boldsymbol{\alpha}, \boldsymbol{\beta}_\gamma, \sigma)$ for the specific parameters of the model

$$M_\gamma : \boldsymbol{y} = \boldsymbol{X}_0\boldsymbol{\alpha} + \boldsymbol{X}_\gamma\boldsymbol{\beta}_\gamma + \boldsymbol{\varepsilon}, \;\; \boldsymbol{\varepsilon} \sim N_n(\boldsymbol{0}, \sigma^2\boldsymbol{I}_n). \tag{12}$$

In **BayesVarSel** the prior used is specified in main functions `Btest`, `Bvs`, and `GibbsBvs` with the argument `prior.betas` with default value `'"Robust"'` that corresponds to the proposal the same name in (Bayarri et al., 2012). It is argued in this paper, based on foundational arguments, that the robust prior is an optimal choice for testing in linear models.

The robust prior for M_γ can be specified hierarchically as

$$\pi^R_\gamma(\boldsymbol{\alpha}, \boldsymbol{\beta}_\gamma, \sigma) = \sigma^{-1}\, N_{p_\gamma}(\boldsymbol{\beta}_\gamma \mid \mathbf{0}, g\,\boldsymbol{\Sigma}_\gamma), \tag{13}$$

where $\boldsymbol{\Sigma}_\gamma = \sigma^2\,(\boldsymbol{V}_\gamma^\top \boldsymbol{V}_\gamma)^{-1}$, with

$$\boldsymbol{V}_\gamma = (\boldsymbol{I}_n - \boldsymbol{X}_0(\boldsymbol{X}_0^\top \boldsymbol{X}_0)^{-1}\boldsymbol{X}_0^\top)\boldsymbol{X}_\gamma, \tag{14}$$

and

$$g \sim p^R_\gamma(g) = \frac{1}{2}\sqrt{\frac{1+n}{p_\gamma + p_0}}(g+1)^{-3/2}, \quad g > \frac{1+n}{p_\gamma + p_0} - 1. \tag{15}$$

For the null model, the prior assumed is $\pi_0(\boldsymbol{\alpha}, \sigma) = \sigma^{-1}$.

The idea of using the matrix $\boldsymbol{\Sigma}_\gamma$ to scale variable selection priors dates back to Zellner and Siow (1980) and is present in other very popular proposals in the literature. As we next describe, these proposals differ about which distribution should be used for the hyperparameter g. Many of these can be implemented in **BayesVarSel** through the argument `prior.betas`.

- `prior.betas="ZellnerSiow"` (Jeffreys (1961); Zellner and Siow (1980, 1984)) corresponds to $g \sim IGa(1/2, n/2)$ (leading to the very famous proposal of using a Cauchy).
- `prior.betas="gZellner"` (Zellner (1986); Kass and Wasserman (1995)) corresponds to fixing $g = n$ (leading to the so called Unit Information Prior).
- `prior.betas="FLS"` (Fernández et al. (2001)) corresponds to fixing $g = \max\{n, p^2\}$.
- `prior.betas="Liangetal"` (Liang et al. (2008)) corresponds to $g \sim \pi(g) \propto (1 + g/n)^{-3/2}$.

onewaytests: An R Package for One-Way Tests in Independent Groups Designs

by Osman Dag, Anil Dolgun and Naime Meric Konar

Abstract One-way tests in independent groups designs are the most commonly utilized statistical methods with applications on the experiments in medical sciences, pharmaceutical research, agriculture, biology, engineering, social sciences and so on. In this paper, we present the **onewaytests** package to investigate treatment effects on the dependent variable. The package offers the one-way tests in independent groups designs, which include ANOVA, Welch's heteroscedastic *F* test, Welch's heteroscedastic *F* test with trimmed means and Winsorized variances, Brown-Forsythe test, Alexander-Govern test, James second order test and Kruskal-Wallis test. The package also provides pairwise comparisons, graphical approaches, and assesses variance homogeneity and normality of data in each group via tests and plots. A simulation study is also conducted to give recommendations for applied researchers on the selection of appropriate one-way tests under assumption violations. Furthermore, especially for non-R users, a user-friendly web application of the package is provided. This application is available at `http://www.softmed.hacettepe.edu.tr/onewaytests`.

Introduction

There are many statistical procedures for one-way independent groups designs. The one-way fixed effects analysis of variance (ANOVA) is the most common one being used for comparing *k* independent group means. Independence of the observations in each group and between groups, normality (i.e., *k* samples come at random from normal distribution) and variance homogeneity (i.e., *k* populations have equal variances) are three basic assumptions of ANOVA. ANOVA is a valid and powerful test for identifying group differences provided that these assumptions are held. However, most of the data sets in practice are not normally distributed and group variances are heterogeneous, and it is difficult to find a data set that satisfies both assumptions. When the assumptions underlying ANOVA are violated, the drawn inferences are also invalid. There have been remarkable efforts to find an appropriate and robust test statistic under non-normality and variance heterogeneity. Several procedures alternative to ANOVA were proposed, including Welch's heteroscedastic *F* test (Welch, 1951), Welch's heteroscedastic *F* test with trimmed means and Winsorized variances (Welch, 1951), Alexander-Govern test (Alexander and Govern, 1994), James second order test (James, 1951, 1954), Brown-Forsythe test (Brown and Forsythe, 1974a,b) and Kruskal-Wallis test (Kruskal and Wallis, 1952).

In the literature, there are many studies in which the comparison of ANOVA and its alternative tests was made with respect to type I error rate and power when the assumptions of ANOVA are not satisfied. The effects of sample size (balanced/unbalanced case), non-normality, unequal variances, and combined effects of non-normality and unequal variances were investigated in details (Wilcox, 1988; Cribbie et al., 2012; Lantz, 2013; Gamage and Weerahandi, 1998; Parra-Frutos, 2013; Cribbie et al., 2007). In the light of these studies, we conduct a Monte Carlo simulation study in an attempt to give recommendations for applied researchers on the selection of appropriate one-way tests.

ANOVA, Welch's heteroscedastic *F* test, Welch's heteroscedastic *F* test with trimmed means and Winsorized variances, Kruskal-Wallis test, and Brown-Forsythe test are available under some packages (given in Table 1) on the Comprehensive R Archive Network (CRAN). Alexander-Govern test and James second order test are also found to be robust to assumption violations, but have been overlooked as their calculation and implementation were not easily available elsewhere.

In this paper, we introduce an R package, **onewaytests** (Dag et al., 2017) which implements Alexander-Govern test and James second order test, in addition to the ANOVA, Welch's heteroscedastic *F* test, Welch's heteroscedastic *F* test with trimmed means and Winsorized variances, Kruskal-Wallis test, and Brown-Forsythe test. Besides the omnibus tests, the package provides pairwise comparisons to investigate which groups create the difference providing that a statistically significant difference is obtained by the omnibus test. Moreover, the package offers several graphic types such as grouped box-and-whisker plot and error bar graph. Also, it provides statistical tests and plots (i.e. Q-Q plot and histogram) to assess variance homogeneity and normality, in the package version 1.5 and later. The **onewaytests** package is publicly available on the CRAN.

The organization of this paper is presented as follows. First, we give the theoretical background on the one-way tests in independent groups designs. Second, we introduce the **onewaytests** package and demonstrate the applicability of the package using two real-life datasets. Third, the web interface of the **onewaytests** package is introduced. A Monte Carlo simulation study is also conducted to give recommendations for applied researchers on the selection of appropriate tests included in the package.

Results of this simulation study and the general conclusions on the effects of assumption violations are mentioned.

One-way tests in independent groups designs

In this section, the statistical tests that are used to test the equality of several populations in the one-way independent groups designs are explained. Among these tests, ANOVA, Welch's heteroscedastic F test (Welch, 1951), Alexander-Govern test (Alexander and Govern, 1994), James second order test (James, 1951, 1954) and Brown-Forsythe test (Brown and Forsythe, 1974a,b) test the null hypothesis $H_0 : \mu_1 = \mu_2 = \ldots = \mu_k$ versus alternative H_1 : at least one μ_j ($j = 1, 2, \ldots, k$) is different. Welch's heteroscedastic F test with trimmed means and Winsorized variances (Welch, 1951) tests the equality of population trimmed means, $H_0 : \mu_{t1} = \mu_{t2} = \ldots = \mu_{tk}$ versus alternative H_1 : at least one μ_{tj} is different, where μ_{tj} represents the trimmed mean of the jth population ($j = 1, 2, \ldots, k$). Kruskal-Wallis test (Kruskal and Wallis, 1952) tests the null hypothesis $H_0 : \theta_1 = \theta_2 = \ldots = \theta_k$ versus alternative H_1 : at least one θ_j ($j = 1, 2, \ldots, k$) is different. For the Kruskal-Wallis test, θ_j represents sum of the ranks of the jth population.

ANOVA

The one-way fixed effects analysis of variance (ANOVA) is the most common statistical procedure to test the equality of k independent population means. Underlying assumptions associated with ANOVA include homogeneity of group variances, normality of distributions and statistical independence of errors. Under these conditions, the ANOVA test statistic,

$$F = \frac{\sum_j n_j(\bar{X}_{.j} - \bar{X}_{..})^2/(k-1)}{\sum_i \sum_j (X_{ij} - \bar{X}_{.j})^2/(N-k)}, \tag{1}$$

follows an F distribution with $k-1$ degrees of freedom for the numerator and $N-k$ degrees of freedom for the denominator. In Equation (1), k is the number of groups, N is the total number of observations, and n_j is the number of observations in the jth group. X_{ij} is the ith observation ($i = 1, 2, \ldots, n_j$) in the jth group ($j = 1, 2, \ldots, k$), $\sum_j n_j = N$, $\bar{X}_{..}$ is the overall mean, where $\bar{X}_{..} = \sum_j n_j(\bar{X}_{.j})/N$ and $\bar{X}_{.j}$ is sample mean for the jth group, where $\bar{X}_{.j} = \sum_i X_{ij}/n_j$. ANOVA is more powerful if the assumptions of normality and variance homogeneity hold true. Non-normality has minimal effect on the type I error when the variances are equal (Lantz, 2013) but when the variances are not equal, ANOVA provides poor control over the type I and type II error rates (Bishop, 1976).

Welch's heteroscedastic F test

Welch (1951) proposed a heteroscedastic alternative to ANOVA that is robust to the violation of variance homogeneity assumption. Under the null hypothesis $H_0 : \mu_1 = \mu_2 = \ldots = \mu_k$, the test statistic F_w,

$$F_w = \frac{\sum_j w_j(\bar{X}_{.j} - X'_{..})^2/(k-1)}{\left[1 + \frac{2}{3}((k-2)\nu)\right]}, \tag{2}$$

follows an F distribution with degrees of freedom $k-1$ for the numerator and $1/\nu$ degrees of freedom for the denominator. In Equation (2), $w_j = n_j/S_j^2$, $S_j^2 = \sum_i (X_{ij} - \bar{X}_{.j})^2/(n_j - 1)$,

$$X'_{..} = \frac{\sum_j w_j \bar{X}_{.j}}{\sum_j w_j},$$

and

$$\nu = \frac{3\sum_j \left[\left(1 - \frac{w_j}{\sum_j w_j}\right)^2/(n_j - 1)\right]}{k^2 - 1}.$$

Welch's heteroscedastic F test is robust and less sensitive to heteroscedasticity when compared to the ANOVA as within groups variance is based on the relationship between the different sample sizes in the different groups instead of a simple pooled variance estimate (Lantz, 2013).

Welch's heteroscedastic *F* test with trimmed means and Winsorized variances

Welch's heteroscedastic F test with trimmed means and Winsorized variances (Welch, 1951) is a robust procedure that tests the equality of means by substituting trimmed means and Winsorized variances for the usual means and variances. Therefore, this test statistic is relatively insensitive to the combined effects of non-normality and variance heterogeneity (Keselman et al., 2008). Let $X_{(1)j} \leq X_{(2)j} \leq \ldots \leq X_{(n_j)j}$ be the ordered observations in the jth group and $g_j = \|\epsilon n_j\|$, ϵ is the proportion to be trimmed in each tail of the distribution. After trimming, the effective sample size for the jth group becomes $h_j = n_j - 2g_j$. Then jth sample trimmed mean is

$$\bar{X}_{tj} = \frac{1}{h_j} \sum_{i=g_j+1}^{n_j-g_j} X_{(i)j},$$

and jth sample Winsorized mean is

$$\bar{X}_{wj} = \frac{1}{n_j} \sum_{i=1}^{n_j} Y_{ij},$$

where

$$Y_{ij} = \begin{cases} X_{(g_j+1)j} & \text{if } X_{ij} \leq X_{(g_j+1)j}, \\ X_{ij} & \text{if } X_{(g_j+1)j} < X_{ij} < X_{(n_j-g_j)j}, \\ X_{(n_j-g_j)j} & \text{if } X_{ij} \geq X_{(n_j-g_j)j}. \end{cases}$$

The sample Winsorized variance is

$$s^2_{wj} = \frac{1}{(n_j-1)} \sum_{i=1}^{n_j} (Y_{ij} - \bar{X}_{wj})^2.$$

Let

$$q_j = \frac{(n_j-1)s^2_{wj}}{h_j(h_j-1)},$$

$$w_j = \frac{1}{q_j},$$

$$U = \sum_j w_j,$$

$$\tilde{X} = \frac{1}{U} \sum_j w_j \bar{X}_{tj},$$

$$A = \frac{1}{k-1} \sum_j w_j (\bar{X}_{tj} - \tilde{X})^2,$$

$$B = \frac{2(k-2)}{k^2-1} \sum_j \frac{(1-w_j/U)^2}{h_j-1}.$$

Under $H_0 : \mu_{t1} = \mu_{t2} = \ldots = \mu_{tk}$, Welch's heteroscedastic F test with trimmed means and Winsorized variances F_{wt},

$$F_{wt} = \frac{A}{B+1}, \tag{3}$$

follows an approximately F distribution with $k-1$ and ν' degrees of freedom, where ν' is

$$\nu' = \left(\frac{3}{k^2-1} \sum_j \frac{(1-w_j/U)^2}{h_j-1} \right)^{-1}.$$

F_{wt} is not only less sensitive to heteroscedasticity and non-normality but also robust to the negative effects of outliers as it utilizes the trimmed means and Winsorised variances (Keselman et al., 2008).

Brown-Forsythe test

Brown and Forsythe (1974a,b) proposed the following test statistic:

$$F_{BF} = \frac{\sum_j n_{.j}(\bar{X}_{.j} - \bar{X}_{..})^2}{\sum_j (1 - n_j/N) S_j^2}. \tag{4}$$

Under the null hypothesis, F_{BF} statistic has an approximately F distribution with $k-1$ and f degrees of freedom, where f is obtained with

$$f = \left(\sum_j c_j^2/(n_j - 1) \right)^{-1},$$

and

$$c_j = \frac{(1 - n_j/N) S_j^2}{\left[\sum_j (1 - n_j/N) S_j^2\right]}.$$

F_{BF} is a modification of ANOVA which has the same numerators as the ANOVA but an altered denominator. This test is more powerful when some variances appear unusually low (Brown and Forsythe, 1974a).

Alexander-Govern test

Alexander and Govern (1994) presented another robust test that is alternative to ANOVA, and it is used when group variances are not homogeneous. The test statistic for Alexander-Govern test is

$$\chi^2_{AG} = \sum_{j=1}^{k} z_j^2. \tag{5}$$

Under the null hypothesis, χ^2_{AG} is distributed as χ^2 distribution with $k-1$ degrees of freedom. In Equation (5),

$$z_j = c + \frac{(c^3 + 3c)}{b} - \frac{(4c^7 + 33c^5 + 240c^3 + 855c)}{(10b^2 + 8bc^4 + 1000b)},$$

where $c = [\alpha \times ln(1 + t_j^2/v_j)]^{1/2}$, $b = 48\alpha^2$, $\alpha = v_j - 0.5$ and $v_j = n_j - 1$. The t statistic for each group is

$$t_j = \frac{\bar{X}_{.j} - X^+}{S_j'}. \tag{6}$$

The variance-weighted mean is $X^+ = \sum_{j=1}^{k} w_j \bar{X}_{.j}$ and the standard error for the jth group is

$$S_j' = \left[\frac{\sum_{i=1}^{n_j} (X_{ij} - \bar{X}_{.j})^2}{n_j(n_j - 1)} \right]^{1/2}.$$

Also, weights for each group are defined as

$$w_j = \frac{1/S_j'^2}{\sum_{j=1}^{k} 1/S_j'^2}. \tag{7}$$

χ^2_{AG} is another modification of ANOVA under heterogeneity of variance using the normalizing transformation of the one-sample t statistic. This test provides a good control of type I and II error rates for normally distributed data, but it is not robust to non-normality (Myers, 1998).

James second order test

An alternative test to ANOVA was proposed by James (1951). This test statistic (J) is

$$J = \sum_j t_j^2, \tag{8}$$

where t_j is given in Equation (6). The test statistic, J, is compared to a critical value, $h(\alpha)$, where

$$\begin{aligned}
h(\alpha) = r &+ \frac{1}{2}(3\chi_4 + \chi_2)T + \frac{1}{16}(3\chi_4 + \chi_2)^2\left(1 - \frac{k-3}{r}\right)T^2 \\
&+ \frac{1}{2}(3\chi_4 + \chi_2)(8R_{23} - 10R_{22} + 4R_{21} - 6R_{12}^2 + 8R_{12}R_{11} - 4R_{11}^2) \\
&+ (2R_{23} - 4R_{22} + 2R_{21} - 2R_{12}^2 + 4R_{12}R_{11} - 2R_{11}^2)(\chi_2 - 1) \\
&+ \frac{1}{4}(-R_{12}^2 + 4R_{12}R_{11} - 2R_{12}R_{10} - 4R_{11}^2 + 4R_{11}R_{10} - R_{10}^2)(3\chi_4 - 2\chi_2 - 1) \\
&+ (R_{23} - 3R_{22} + 3R_{21} - R_{20})(5\chi_6 + 2\chi_4 + \chi_2) \\
&+ \frac{3}{16}(R_{12}^2 - 4R_{23} + 6R_{22} - 4R_{21} + R_{20})(35\chi_8 + 15\chi_6 + 9\chi_4 + 5\chi_2) \\
&+ \frac{1}{16}(-2R_{22} + 4R_{21} - R_{20} + 2R_{12}R_{10} - 4R_{11}R_{10} + R_{10}^2)(9\chi_8 - 3\chi_6 - 5\chi_4 - \chi_2) \\
&+ \frac{1}{4}(-R_{22} + R_{11}^2)(27\chi_8 + 3\chi_6 + \chi_4 + \chi_2) + \frac{1}{4}(R_{23} - R_{12}R_{11})(45\chi_8 + 9\chi_6 + 7\chi_4 + 3\chi_2).
\end{aligned}$$

For any integers s and t, $R_{st} = \sum_j (n_j - 1)^{-s} w_j^t$, $\chi_{2s} = r^s / [(k-1)(k+1)\ldots(k+2s-3)]$ where r is the $(1-\alpha)$ centile of a χ^2 distribution with $k-1$ degrees of freedom, $T = \sum_j (1-w_j)^2/(n_j - 1)$ and w_j is defined in Equation (7). If the test statistic, J, exceeds $h(\alpha)$, then the null hypothesis is rejected.

The James second order test has been acknowledged as the best option for both normal heteroscedastic data (Alexander and Govern, 1994) and non-normal symmetric heteroscedastic data (Oshima and Algina, 1992). The disadvantage of this method is the complexity of the computation of critical values.

Kruskal-Wallis test

Kruskal and Wallis (1952) proposed the nonparametric alternative to ANOVA. Let r_{ij} denote the rank of X_{ij} when $N = n_1 + \ldots + n_k$ observations are ranked from smallest to largest. $R_j = \sum_{i=1}^{n_j} r_{ij}$ is the sum of ranks assigned to the observations in the jth group and $\bar{R}_j = R_j/n_j$ is the average rank for these observations. Under these definitions, the Kruskal-Wallis test statistic is given by

$$\chi_{KW}^2 = \frac{1}{S^2}\left(\sum_{j=1}^{k}\frac{R_j^2}{n_j} - \frac{N(N+1)^2}{4}\right), \tag{9}$$

where

$$S^2 = \frac{1}{N-1}\left(\sum_{j=1}^{k}\sum_{i=1}^{n_j} r_{ij}^2 - \frac{N(N+1)^2}{4}\right).$$

When $n_j \to \infty$, χ_{KW}^2 has an asymptotic χ^2 distribution under the null hypothesis $H_0 : \theta_1 = \theta_2 = \ldots = \theta_k$. We reject H_0 if $\chi_{KW}^2 \geq \chi_{k-1,\alpha}^2$ where $\chi_{k-1,\alpha}^2$ is the upper α percentile for the χ^2 distribution with $k-1$ degrees of freedom. Note that, when there are no ties, S^2 simplifies to $N(N+1)/12$.

The Kruskal-Wallis test is robust to non-normality as it utilizes ranks instead of actual values. However, it assumes that the observations in each group come from populations with the same shape of distribution. Therefore, if different groups have different shapes (e.g., some are skewed to the right and another is skewed to the left), the Kruskal-Wallis test may give inaccurate results (Fagerland and Sandvik, 2009).

There are other R packages including one-way tests in independent groups designs; namely, **stats** (R Core Team, 2017), **lawstat** (Gastwirth et al., 2017), **coin** (Hothorn et al., 2006), **car** (Fox and Weisberg, 2011), **WRS2** (Mair et al., 2017), **welchADF** (Villacorta, 2017). The `aov.test` (**onewaytests**) can be seen as a simplified version of `anova` (**stats**) and `Anova` (**car**), but the latter two are more flexible and comprehensive in a way that they can handle two or more variables (e.g., two-way ANOVA, multivariate ANOVA, repeated measures ANOVA, etc.). Amongst the alternatives, the **onewaytests** is the first package including seven different one-way independent design tests that are covered in one package. A brief comparison between these packages and **onewaytests** is given in Table 1. In Table 1, the numbers in parentheses indicate the rankings of the functions in terms of computational speed and the checks indicate the availability of the specific one-way design test in the packages. The

onewaytests package has the fastest running times except for Welch's heteroscedastic F test, where the **stats**'s function elapse time is 0.001 seconds faster. Note that the computational time is system-specific and may vary according to the number of iterations and the inclusion of additional arguments in the functions.

Table 1: Comparison of R packages including one-way independent design tests

Test	stats	car	lawstat	welchADF	WRS2	coin	onewaytests
F	(2)✓	(3)✓					(1)✓
F_w	(1)✓			(4)✓	(3)✓		(2)✓
F_{wt}				(3)✓	(2)✓		(1)✓
F_{BF}			(2)✓				(1)✓
χ^2_{AG}							✓
J							✓
χ^2_{KW}	(2)✓					(3)✓	(1)✓

Demonstration of the onewaytests package

The **onewaytests** package includes several functions specially designed for one-way independent groups designs along with functions for assessing fundamental assumptions via relevant tests and plots. In this section, we demonstrate the usage of **onewaytests** package by using two different data sets from different fields.

Iris data

In this part, we work with iris data set, collected by Anderson (1935), available in R. Fisher (1936) introduced this data set as an application of the linear discriminant analysis. At present, the popularity of the data set still continues within a variety of studies; examples include the studies related to data mining (Hu, 2005), multivariate normality assessment (Korkmaz et al., 2014), and so on.

This data set includes iris flowers' length and width measurements of sepal and petal characteristics in centimeters along with a grouping variable indicating the type of iris flowers (setosa, virginica and versicolor). For illustrating the implementation of our package, we use sepal length measurements as a response variable and iris types as a grouping variable. This data set has a total of 150 observations (50 observations in each type of iris flowers).

After installing and loading **onewaytests** package, the functions designed for one-way independent groups designs are available to be used. One-way tests in this package generally give test statistics and p-values to decide on the hypothesis of the statistical process, except for James second order test. In this test, test statistic and critical value are given as an output since the asymptotic distribution of the test statistic is not available.

The **onewaytests** package is also able to give some basic descriptive statistics of the given groups.

```
# obtain some basic descriptive statistics
R> describe(Sepal.Length ~ Species, data = iris)

            n  Mean   Std.Dev Median Min Max  25th 75th  Skewness Kurtosis NA
setosa     50 5.006 0.3524897    5.0 4.3 5.8 4.800  5.2 0.1164539 2.654235  0
versicolor 50 5.936 0.5161711    5.9 4.9 7.0 5.600  6.3 0.1021896 2.401173  0
virginica  50 6.588 0.6358796    6.5 4.9 7.9 6.225  6.9 0.1144447 2.912058  0
```

For all functions in the **onewaytests** package, the argument must be specified as a formula in which left and right-hand sides of the formula give sample values and its corresponding groups, respectively. The left and right-hand sides of the formula must have one variable. The variable on the left must be a numeric while the variable on the right must be a factor. Otherwise, each function returns an error message.

The functions for one-way design tests are coded from scratch. The function for pairwise comparison is coded for pairwise comparison of one-way independent groups designs by using the p-value adjustment methods available in the `p.adjust` function under **stats** package. For checking assumptions, the tests are used from different packages (**stats**, **car** and **nortest** (Gross and Ligges, 2015)) and adapted for one-way independent groups designs. The function for the graphics is coded from scratch using the **ggplot2** package (Wickham, 2009).

One-way tests in independent groups designs

The **onewaytests** package includes seven one-way tests in independent groups designs. In this part, the implementations of these tests are presented.

ANOVA: aov.test(...)

The `aov.test` is utilized to test the equality of k independent population means.

```
R> aov.test(Sepal.Length ~ Species, data = iris, alpha = 0.05, na.rm = TRUE,
     verbose = TRUE)

  One-Way Analysis of Variance
-------------------------------------------------------
  data : Sepal.Length and Species

  statistic  : 119.2645
  num df     : 2
  denom df   : 147
  p.value    : 1.669669e-31

  Result     : Difference is statistically significant.
-------------------------------------------------------
```

Here, the `statistic` is the ANOVA test statistic distributed as F with the degrees of freedom for the numerator (`num df`) and denominator (`denom df`). Also, `p.value` is the significance value of the test statistic. Since the p-value, derived from `aov.test`, is lower than 0.05, it can be concluded that there is statistically significant difference between the iris species (F = 119.2645, p-value = 1.669669×10^{-31}).

`alpha` is the level of significance to assess the statistical difference. Default is set to `alpha` = 0.05. `na.rm` is a logical value indicating whether NA values should be stripped before the computation proceeds. Default is `na.rm` = TRUE. `verbose` is a logical for printing output to R console. Default is set to `verbose` = TRUE. These arguments are available in the functions for one-way tests and checking assumptions. The users who would like to use the statistics in the output in their programs can use the following codes.

```
R> result <- aov.test(Sepal.Length ~ Species, data = iris, alpha = 0.05, na.rm = TRUE,
     verbose = FALSE)

# the ANOVA test statistic
R> result$statistic
[1] 119.2645

# the degrees of freedom for numerator and denominator
R> result$parameter
[1]   2 147

# the p-value of the test
R> result$p.value
[1] 1.669669e-31
```

Here, the codes for how to obtain the statistics from the ANOVA output are given. Since all one-way design tests return similar outputs, similar codes are not repeated in the other tests. For all tests, the level of significance is taken as 0.05.

Welch's heteroscedastic F test: welch.test(...)

One may use the `welch.test` function in the **onewaytests** package to perform Welch's heteroscedastic F test.

```
R> welch.test(Sepal.Length ~ Species, data = iris)

  Welch's Heteroscedastic F Test
-------------------------------------------------------
  data : Sepal.Length and Species
```

```
  statistic  : 138.9083
  num df     : 2
  denom df   : 92.21115
  p.value    : 1.505059e-28

  Result     : Difference is statistically significant.
----------------------------------------------------------
```

In the output, similar to `aov.test`, the `statistic` is the Welch's test statistic distributed as F with the degrees of freedom for the numerator (`num df`) and denominator (`denom df`). One can conclude that the difference between the species is statistically significant (F_w = 138.9083, p-value = 1.505059×10^{-28}).

Welch's heteroscedastic *F* test with trimmed means and Winsorized variances: welch.test(...)

The `welch.test` function in the **onewaytests** package is also used to perform Welch's heteroscedastic *F* test with trimmed means and Winsorized variances.

```
R> welch.test(Sepal.Length ~ Species, data = iris, rate = 0.1)

Welch's Heteroscedastic F Test with Trimmed Means and Winsorized Variances
------------------------------------------------------------------------------
  data : Sepal.Length and Species

  statistic  : 123.6698
  num df     : 2
  denom df   : 71.64145
  p.value    : 5.84327e-24

  Result     : Difference is statistically significant.
------------------------------------------------------------------------------
```

Here, the `statistic` is the test statistic distributed as F with the degrees of freedom for the numerator (`num df`) and denominator (`denom df`). Moreover, the `rate` is the rate of observations trimmed and Winsorized from each tail of the distribution. If `rate = 0`, it performs Welch's heteroscedastic *F* test. Otherwise, one can perform Welch's heteroscedastic *F* test with trimmed means and Winsorized variances. Default is set to `rate = 0`. One can conclude that there is a statistically significant difference between the species (F_{wt} = 123.6698, p-value = 5.84327×10^{-24}).

Brown-Forsythe test: bf.test(...)

One may use the `bf.test` function in the **onewaytests** package to perform Brown-Forsythe test to compare more than two groups.

```
R> bf.test(Sepal.Length ~ Species, data = iris)

  Brown-Forsythe Test
----------------------------------------------------------
  data : Sepal.Length and Species

  statistic  : 119.2645
  num df     : 2
  denom df   : 123.9255
  p.value    : 1.317059e-29

  Result     : Difference is statistically significant.
----------------------------------------------------------
```

In the output, the `statistic` is the Brown-Forsythe test statistic distributed as F with the degrees of freedom for numerator (`num df`) and denominator (`denom df`). It can be concluded that there is a statistically significant difference between the iris species (F_{BF} = 119.2645, p-value = 1.317059×10^{-29}).

Alexander-Govern test: ag.test(...)

The `ag.test` function in the **onewaytests** package is used to perform Alexander-Govern test.

```
R> ag.test(Sepal.Length ~ Species, data = iris)

  Alexander-Govern Test
------------------------------------------------------------
  data : Sepal.Length and Species

  statistic  : 146.3573
  parameter  : 2
  p.value    : 1.655451e-32

  Result     : Difference is statistically significant.
------------------------------------------------------------
```

Here, `statistic` is the Alexander-Govern test statistic distributed as χ^2 with the degrees of freedom (`parameter`). One can conclude that the difference between the species is statistically significant (χ^2_{AG} = 146.3573, p-value = 1.655451×10^{-32}).

James second order test: james.test(...)

One may use the `james.test` function in the **onewaytests** package to perform James second order test.

```
R> james.test(Sepal.Length ~ Species, data = iris, alpha = 0.05)

  James Second Order Test
------------------------------------------------------------------
  data : Sepal.Length and Species

  statistic     : 279.8251
  criticalValue : 6.233185

  Result        : Difference is statistically significant.
------------------------------------------------------------------
```

Here, `alpha` is the significance level, statistic is the James second order test statistic, J, `criticalValue` is the critical value, $h(\alpha)$, corresponding to the significance level, α. If J exceeds $h(\alpha)$, then the null hypothesis is rejected. Since $J = 279.8251$, obtained from `james.test`, is higher than $h(\alpha) = 6.233185$, it can be concluded that there is a statistically significant difference between the iris species.

Kruskal-Wallis test: kw.test(...)

The `kw.test` function in the **onewaytests** package is utilized to perform Kruskal-Wallis test.

```
R> kw.test(Sepal.Length ~ Species, data = iris)

  Kruskal-Wallis Test
------------------------------------------------------------
  data : Sepal.Length and Species

  statistic  : 96.93744
  parameter  : 2
  p.value    : 8.918734e-22

  Result     : Difference is statistically significant.
------------------------------------------------------------
```

In the output, `statistic` is the Kruskal-Wallis test statistic distributed as χ^2 with the degrees of freedom (`parameter`). One can conclude that the difference between the species is statistically significant (χ^2_{KW} = 96.93744, p-value = 8.918734×10^{-22}).

Pairwise comparisons

In this part, we present the pairwise comparisons to investigate which groups create the difference. We utilized the `p.adjust` function under the **stats** package (R Core Team, 2017) for our `paircomp`

function. The `paircomp` function has also the same p-value adjustment methods as `p.adjust`, including `bonferroni`, `holm` (Holm, 1979), `hochberg` (Hochberg, 1988), `hommel` (Hommel, 1988), `BH` (Benjamini and Hochberg, 1995), `BY` (Benjamini and Yekutieli, 2001), and `none`. The default is set to `"bonferroni"`. Pairwise comparisons are made by adjusting p-values according to the specified method, except when the method is James second order test, which requires adjusting the significance level instead of the p-value.

One-way tests return a list with class "owt" except for James second order test. The reason for returning a list with class "owt" is to introduce the output to the `paircomp` function for pairwise comparison adjusting p-values according to the specified method. Besides, James second order test returns a list with class "jt" introducing the output to the `paircomp` function for pairwise comparison by adjusting the significance level instead of the p-value.

In the iris example, there is a statistically significant difference between iris species in terms of sepal length measurements. All pairwise comparisons of groups can be conducted using `paircomp` function. For simplicity, the Bonferroni correction is applied to show the usage of the pairwise comparisons following the significant result obtained by Alexander-Govern test and James second order test.

```
# Alexander-Govern test
R> out <- ag.test(Sepal.Length ~ Species, data = iris, verbose = FALSE)
R> paircomp(out, adjust.method = "bonferroni")

  Bonferroni Correction (alpha = 0.05)
-------------------------------------------------------
   Level (a)  Level (b)      p.value   No difference
1     setosa versicolor 8.187007e-17          Reject
2     setosa  virginica 1.105024e-25          Reject
3 versicolor  virginica 5.913702e-07          Reject
-------------------------------------------------------

# James second order test
R> out <- james.test(Sepal.Length ~ Species, data = iris, verbose = FALSE)
R> paircomp(out, adjust.method = "bonferroni")

  Bonferroni Correction (alpha = 0.0166666666666667)
-----------------------------------------------------------------
   Level (a)  Level (b) statistic criticalValue   No difference
1     setosa versicolor  110.6912      5.959328          Reject
2     setosa  virginica  236.7350      5.992759          Reject
3 versicolor  virginica   31.6875      5.938643          Reject
-----------------------------------------------------------------
```

In both Alexander-Govern and James second order tests with Bonferroni correction, statistical differences between all types of iris flowers are significant in terms of sepal length measurements.

Checking assumptions via tests and plots

Two main assumptions, normality and variance homogeneity, can be checked through the **onewaytests** package. One can assess the variance homogeneity via `homog.test`, which has options including `Levene` (Levene's test), `Bartlett` (Bartlett's test) and `Fligner` (Fligner-Killeen test). Also, the normality of data in each group can be checked through `nor.test`, which has options including `SW` (Shapiro-Wilk test), `SF` (Shapiro-Francia test), `LT` (Lilliefors test known as Kolmogorov-Smirnov test), `AD` (Anderson-Darling test), `CVM` (Cramer-von Mises test), `PT` (Pearson Chi-square test). Moreover, the `nor.test` has options to assess the normality of data in each group through plots (Q-Q plots and Histograms with normal curves).

```
# Bartlett's homogeneity test
R> homog.test(Sepal.Length ~ Species, data = iris, method = "Bartlett")

 Bartlett's Homogeneity Test
---------------------------------------------
  data : Sepal.Length and Species

  statistic  : 16.0057
  parameter  : 2
  p.value    : 0.0003345076
```

```
  Result     : Variances are not homogeneous.
-----------------------------------------------
```

Bartlett's homogeneity test results reveal that the variances between iris species are not homogeneous (χ^2 = 16.0057, p-value = 0.0003345076).

```
# Shapiro-Wilk normality test
R> nor.test(Sepal.Length ~ Species, data = iris, method = "SW", plot = "qqplot-histogram")

  Shapiro-Wilk Normality Test
-------------------------------------------------
  data : Sepal.Length and Species

      Level Statistic   p.value   Normality
1    setosa 0.9776985 0.4595132  Not reject
2 versicolor 0.9778357 0.4647370  Not reject
3  virginica 0.9711794 0.2583147  Not reject
-------------------------------------------------
```

Shapiro-Wilk normality test results state that there is not enough evidence to reject the normality of sepal length values in each iris species since all p-values are greater than 0.05. Also, the normality of data in each group can be assessed visually by Q-Q plots and histograms with normal curves (Figure 1).

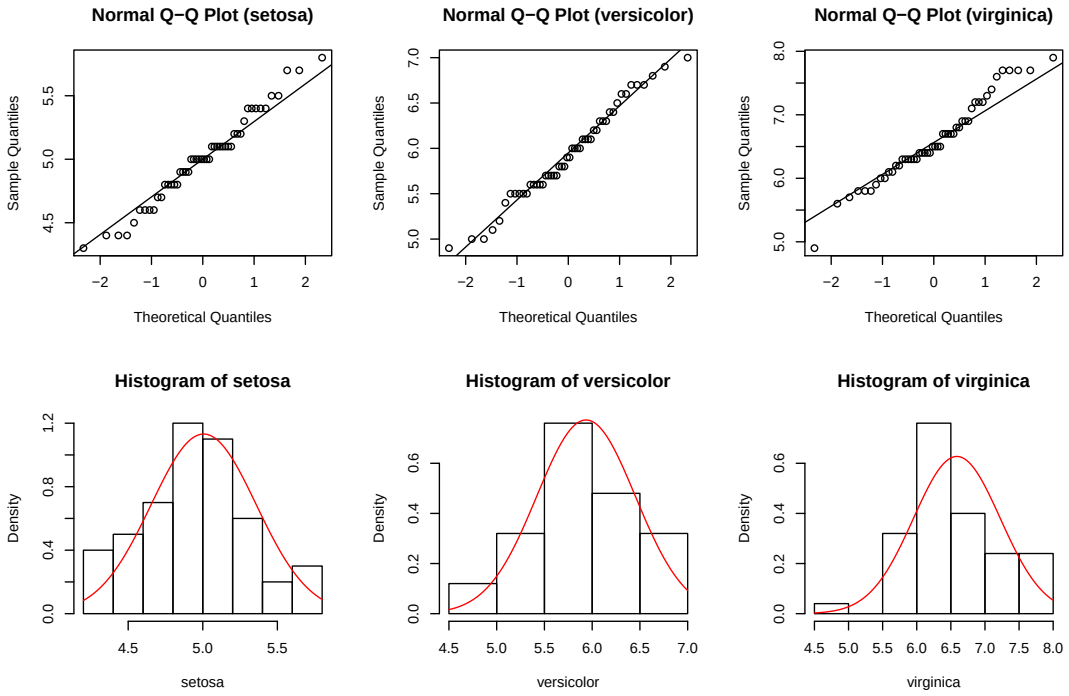


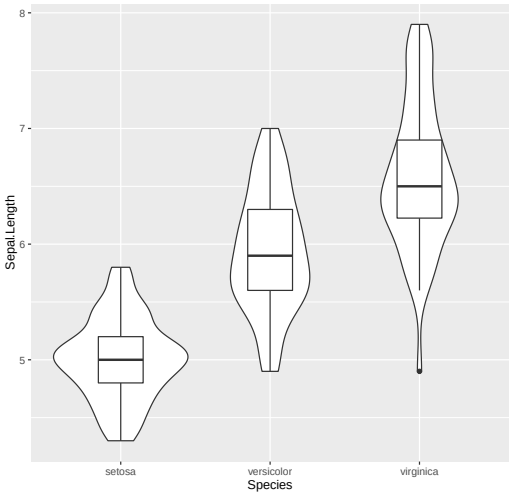
Figure 1: Q-Q plots and histograms with normal curves

Graphical approaches

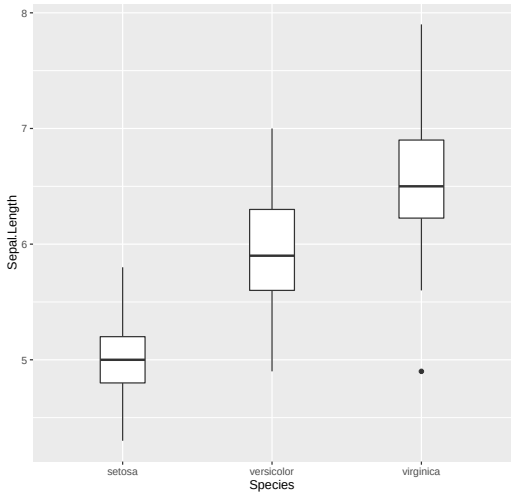
The users can obtain several graphic types of given groups via the `gplot`, which has options involving box-and-whisker plot with violin line - a rotated density line on each side - (Figure 2a), box-and-whisker plot (Figure 2b), mean ± standard deviation graph (Figure 2c) and mean ± standard error graph (Figure 2d). These graphics can be obtained via the following codes:

```
# Box-and-whisker plot with violin line
R> gplot(Sepal.Length ~ Species, data = iris, type = "boxplot")

# Box-and-whisker plot
R> gplot(Sepal.Length ~ Species, data = iris, type = "boxplot", violin = FALSE)
```

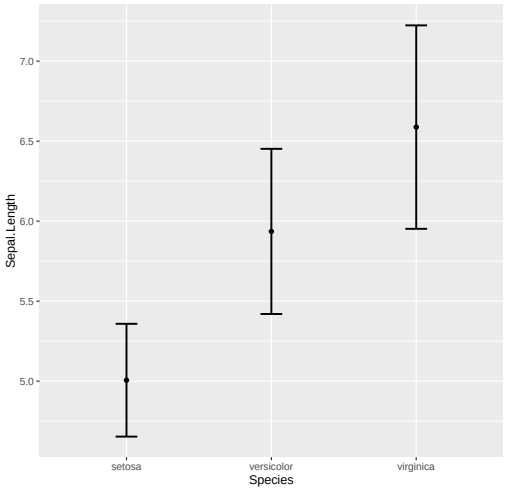
```
# Mean +- standard deviation graph
R> gplot(Sepal.Length ~ Species, data = iris, type = "errorbar", option = "sd")

# Mean +- standard error graph
R> gplot(Sepal.Length ~ Species, data = iris, type = "errorbar", option = "se")
```



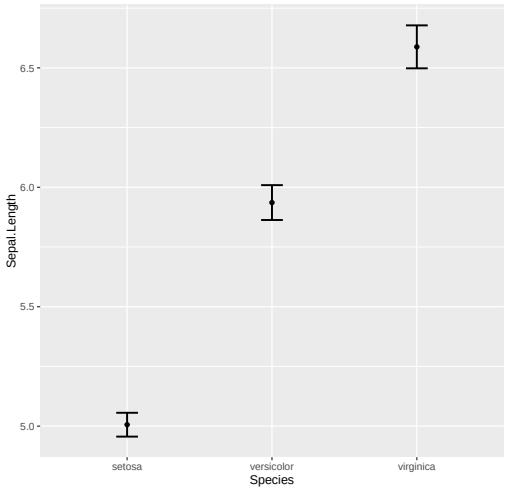
(a) Box-and-whisker plot with violin line



(b) Box-and-whisker plot



(c) Mean $\pm$ standard deviation graph



(d) Mean $\pm$ standard error graph

Figure 2: Graphics of given groups

German breast cancer data

In this part, we utilize German breast cancer study group (GBSG) data set, available in the **mfp** package (Ambler and Benner, 2015) in R. This data set was collected from a cohort study and measurements from patients with primary node positive breast cancer were taken between July 1984 and December 1989. Seven risk factors, including menopausal status, tumor size, tumor grade, age, number of positive lymph nodes, progesterone and oestrogen receptor concentrations were examined in the work done by Sauerbrei et al. (1999). For illustrative purposes, we take the dependent variable as recurrence - free survival times of patients, of whom an event for recurrence-free survival occurs, and the factor as tumor grade. This factor is a three-level categorical variable: 1 = grade I, 2 = grade II , and 3 = grade III. A total of 299 observations (18, 202, 79 observations in each group, respectively) are available.

The objective of adding GBSG dataset is showing the usage of the package in practice rather than demonstrating the usage of all functions in the **onewaytests** package. After checking the normality and variance homogeneity assumptions, an appropriate one-way test is decided to compare groups.

Also, pairwise comparisons are applied when it is necessary to determine which groups create the difference.

After installing and loading the **mfp** package, GBSG dataset can be reached by using the following R code.

```
# load GBSG data
R> data("GBSG")

# select the patients of whom an event for recurrence-free survival occurs
R> GBSG_subset <- GBSG[GBSG$cens == 1,]

# obtain the descriptive statistics of the tumor grades in terms of recurrence-free
# survival times
R> describe(rfst ~ tumgrad, data = GBSG_subset)

    n      Mean  Std.Dev Median Min  Max 25th    75th  Skewness Kurtosis NA
1  18 1052.1111 444.5332  969.0 476 1990  729 1290.25 0.8712495 2.938206  0
2 202  845.9505 511.2683  729.5  72 2456  487 1160.75 0.9484978 3.253876  0
3  79  616.6076 432.2091  476.0  98 2034  312  758.00 1.4487566 4.698735  0
```

The assumptions of variance homogeneity and normality are assessed by using `homog.test` and `nor.test`, respectively.

```
# Bartlett's homogeneity test
R> homog.test(rfst ~ tumgrad, data = GBSG_subset, method = "Bartlett")

  Bartlett's Homogeneity Test
-----------------------------------------------
  data : rfst and tumgrad

  statistic  : 3.262419
  parameter  : 2
  p.value    : 0.1956927

  Result     : Variances are homogeneous.
-----------------------------------------------
```

Bartlett's homogeneity test results reveal that there is no enough evidence to reject the variance homogeneity (χ^2 = 3.262419, p-value = 0.1956927) since p-value is larger than 0.05.

```
# Shapiro-Wilk normality test
R> nor.test(rfst ~ tumgrad, data = GBSG_subset, method = "SW")

  Shapiro-Wilk Normality Test
-------------------------------------------------
  data : rfst and tumgrad

  Level Statistic      p.value   Normality
1     1 0.9097324 8.510408e-02  Not reject
2     2 0.9195909 4.749653e-09      Reject
3     3 0.8489033 1.708621e-07      Reject
-------------------------------------------------
```

Shapiro-Wilk normality test results state that there is not enough evidence to reject the normality of recurrence - free survival times of the patients with tumor grade I since p-value is greater than 0.05. The normality of recurrence - free survival times of the patients with tumor grade II and III is not met since the p-value is smaller than 0.05. Our simulation study results suggest that ANOVA is the appropriate one-way test in such case.

```
R> aov.test(rfst ~ tumgrad,data = GBSG_subset)

  One-Way Analysis of Variance
-----------------------------------------------------------
  data : rfst and tumgrad
```

```
  statistic  : 8.875494
  num df     : 2
  denom df   : 296
  p.value    : 0.000180542

  Result     : Difference is statistically significant.
-----------------------------------------------------------
```

Since the p-value, derived from `aov.test`, is lower than 0.05, it can be concluded that there is a statistically significant difference between the tumor grades with respect to recurrence - free survival times (F = 8.875494, p-value = 0.000180542).

On the other hand, we observe that recurrence - free survival times of the patients with tumor grade II and III have few outliers and positively-skewed distributed. Liao et al. (2016) suggests that Kruskal-Wallis test should be used in such cases. Therefore, this test is also utilized along with ANOVA to compare groups.

```
R> kw.test(rfst ~ tumgrad, data = GBSG_subset)

  Kruskal-Wallis Test
-----------------------------------------------------------
  data : rfst and tumgrad

  statistic  : 23.42841
  parameter  : 2
  p.value    : 8.176855e-06

  Result     : Difference is statistically significant.
-----------------------------------------------------------
```

One can conclude that the difference between the tumor grades with respect to recurrence - free survival times is statistically significant (χ^2_{KW} = 23.42841, p-value = 8.176855×10^{-06}) since p-value is smaller than 0.05.

Both ANOVA and Kruskal-Wallis test results reveal that there is a statistically significant difference between tumor grade groups in terms of recurrence-free survival times. In the next step, we need to determine which groups create the difference.

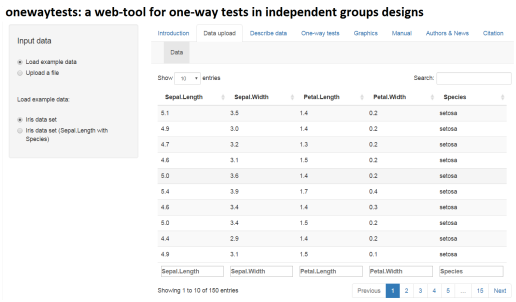
```
# ANOVA
R> out <- aov.test(rfst ~ tumgrad, data = GBSG_subset, verbose = FALSE)
R> paircomp(out, adjust.method = "bonferroni")

  Bonferroni Correction (alpha = 0.05)
-----------------------------------------------------
  Level (a) Level (b)      p.value   No difference
1         1         2 0.2980174599      Not reject
2         1         3 0.0006698433          Reject
3         2         3 0.0014901832          Reject
-----------------------------------------------------

# Kruskal-Wallis test
R> out<- kw.test(rfst ~ tumgrad, data = GBSG_subset, verbose = FALSE)
R> paircomp(out, adjust.method = "bonferroni")

  Bonferroni Correction (alpha = 0.05)
-----------------------------------------------------
  Level (a) Level (b)      p.value   No difference
1         1         2 0.0949942647      Not reject
2         1         3 0.0001333143          Reject
3         2         3 0.0002457434          Reject
-----------------------------------------------------
```

Pairwise comparisons with Bonferroni correction after both ANOVA and Kruskal-Wallis test indicate that there is no statistically significant difference between the patients with tumor grade I and

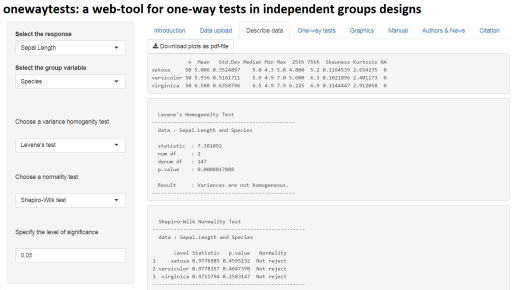
those with tumor grade II in terms of recurrence-free survival times. On the other hand, there exists a statistically significant difference between the patients with tumor grade I/II and those with tumor grade III.

Web interface of onewaytests package

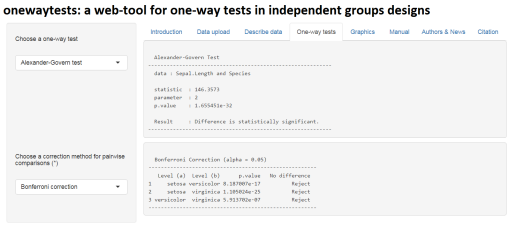
The objective of this package is to provide the users with one-way tests in independent groups designs, pairwise comparisons, graphical approaches, and assess variance homogeneity and normality of data in each group via tests and plots. At times, it is difficult for new R users or applied researchers to deal with R codes. Thus, we have developed a web interface of **onewaytests** package by using **shiny** (Chang et al., 2017). The web interface is available at `http://www.softmed.hacettepe.edu.tr/onewaytests`.



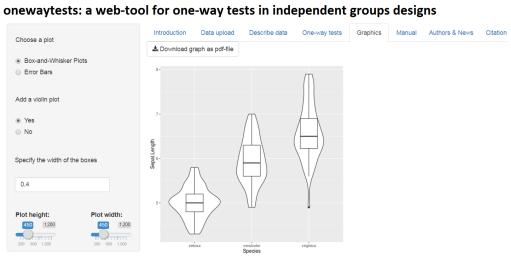
(a) Data upload



(b) Describe data



(c) One-way tests



(d) Graphics

Figure 3: Web interface of onewaytests package

Users can upload their data to the tool via *Data upload* tab (Figure 3a). There exist two demo datasets on this tab for the users to test the tool. Basic descriptive statistics can be obtained through *Describe data* tab (Figure 3b). Moreover, assumptions of variance homogeneity and normality can be checked via this tab to decide on which one-way test is appropriate to test the statistical difference. Variance homogeneity is checked by variance homogeneity tests (Levene's test, Bartlett's test, Fligner-Killeen test). The normality of data in each group is assessed by normality tests (Shapiro-Wilk, Cramer-von Mises, Lilliefors (Kolmogorov-Smirnov), Shapiro-Francia, Anderson-Darling, Pearson Chi-Square tests) and plots (Q-Q plot and Histogram with a normal curve). After describing the data, users can test statistical difference between groups through *One-way tests* tab (Figure 3c). In this tab, there exist one-way tests (ANOVA, Welch's heteroscedastic F, Welch's heteroscedastic F test with trimmed means and Winsorized variances, Brown-Forsythe, Alexander-Govern, James second order test, Kruskal-Wallis tests) and pairwise comparison methods (Bonferroni, Holm, Hochberg, Hommel, Benjamini-Hochberg, Benjamini-Yekutieli, no corrections). Moreover, users can obtain the graphics of given groups (Figure 2) through *Graphics* tab (Figure 3d).

Simulation study

In this section, it is aimed to compare the performances of seven tests in terms of type I error and adjusted power (Lloyd, 2005), and give some general suggestions on which test(s) should be used or avoided under the violations of assumptions. The adjusted power, $R(\alpha)$, is

$$R(\alpha) = \Phi(\hat{\delta} + \Phi^{-1}(\alpha)), \tag{10}$$

with

$$\hat{\delta} = \Phi^{-1}(1-\hat{\beta}) - \Phi^{-1}(\hat{\alpha}).$$

In Equation (10), $\hat{\alpha}$ is the estimated type I error, $\hat{\beta}$ is the estimated type II error, α is the nominal size for type I error, Φ and Φ^{-1} are the cumulative and inverse cumulative distribution functions of the standard normal distribution, respectively.

Simulation design

A Monte Carlo simulation is implemented to illustrate the performances of these tests for the scenarios in which the assumptions of normality and/or variance homogeneity are held or not. The algorithm of the simulation study can be depicted as follows:

1. Generate three random samples from normal or skew normal distribution with means μ_1, μ_2 and μ_3 and standard deviations σ_1, σ_2, σ_3. Group standard deviations are taken as $\sigma_1 = \sigma_2 = \sigma_3 = 1$ for homogeneous case, and $\sigma_1 = 1$, $\sigma_2 = \sqrt{2}$, $\sigma_3 = 2$ for heterogeneous case. Skewness γ is set to $\gamma = 0$ for normal distribution, $\gamma = 0.5$ for positive skew normal distribution and $\gamma = -0.5$ for negative skew normal distribution with different sample size combinations (balanced and unbalanced). Set $\mu_1 = \mu_2 = \mu_3 = 0$ for gathering type I error; $\mu_1 = 0$, $\mu_2 = 0.25$, $\mu_3 = 0.5$ or $\mu_1 = 0$, $\mu_2 = 0.5$, $\mu_3 = 1$ or $\mu_1 = 0$, $\mu_2 = 1$, $\mu_3 = 2$ for power.
2. Check whether groups are different by the corresponding one-way test in independent groups designs at the level of significance α ($\alpha = 0.05$).
3. Repeat steps i) - ii) for 10,000 times and calculate the probability of rejecting the null hypothesis when the null hypothesis is true (type I error) or false (power).
4. Calculate adjusted power (given in Lloyd (2005)) by using the estimated type I error and power found in iii).

Results

In this section, the performances of one-way tests in independent groups designs are investigated through type I error and adjusted power. All results are not given here to protect the content integrity, but attached as supplementary tables 2–4.

Type I error rates

In this part, we compare seven one-way tests in terms of their type I errors. We observed that the type I errors get closer to nominal level as sample size increases under the data generated from normal distribution ($\gamma = 0$) with equal variances ($\sigma_1 = \sigma_2 = \sigma_3 = 1$). Kruskal-Wallis test tends to be more conservative compared to the rest of them in the unbalanced small sample sizes ($n_1 = 6, n_2 = 9, n_3 = 15$). When variance homogeneity is not held under normality, unbalance of sample sizes causes a decline in type I error rates of ANOVA and Kruskal-Wallis test regardless of sample size.

When the data in each group come from right skew normal distribution ($\gamma = 0.5$) with unbalanced small sample size setting, type I error rate of ANOVA is estimated to be 0.052 under variance homogeneity ($\sigma_1 = \sigma_2 = \sigma_3 = 1$) whereas it declines to 0.026 under variance heterogeneity ($\sigma_1 = 1, \sigma_2 = \sqrt{2}, \sigma_3 = 2$). The type I error rate of ANOVA is negatively affected by the combined effect of heterocedasticity and unbalanced sample size. Especially, when the smaller variances are associated with groups having smaller sample size, the empirical type I error rate of ANOVA is halved compared to the homoscedastic case. Kruskal-Wallis test is dramatically affected by variance heterogeneity. It tends to be conservative in the unbalanced case whereas it becomes liberal in the balanced case.

James second order test, Alexander-Govern test, Welch's heteroscedastic F test and Welch's heteroscedastic F test with trimmed means and Winsorized variances control the type I error rate at the nominal size for all simulation scenarios. Brown-Forsythe test is not able to control type I error rate; especially when the variances are heterogeneous, it becomes liberal.

Adjusted powers

Adjusted power is important to compare tests having different type I errors since it adjusts power with respect to type I error. The results are illustrated in Figures 4–5 to see the clear difference among the tests.

Within the tests discussed in this study, ANOVA is the best one with respect to adjusted power when the sample sizes are not equal under normality ($\gamma = 0$) and variance homogeneity ($\sigma_1 = \sigma_2 =$

$\sigma_3 = 1$). Under the same condition, ANOVA and Brown-Forsythe test are superior to the rest of them in terms of adjusted power when the design of sample size is balanced. As expected, ANOVA performs poor under normality when variances are unstable ($\sigma_1 = 1, \sigma_2 = \sqrt{2}, \sigma_3 = 2$); however, Alexander-Govern, James and Welch's heteroscedastic *F* tests have higher adjusted powers compared to other tests. Brown-Forsythe and Kruskal-Wallis tests perform poorly compared to the others under the same condition.

When the data are generated from positive skew normal distribution ($\gamma = 0.5$) with equal variances, ANOVA and Kruskal-Wallis test perform better than other tests. ANOVA is slightly better than Kruskal-Wallis test for small sample sizes and vice versa is true for the medium sample sizes. James, Alexander-Govern and Welch's heteroscedastic *F* tests have the highest adjusted powers among the whole tests for all scenario combinations under the data generated from a positive skew normal distribution with heterogeneous variances.

ANOVA, Brown-Forsythe test and Kruskal-Wallis test perform better than the other tests when the data are generated from negative skew normal distribution ($\gamma = -0.5$) with homogeneous variances and equal sample sizes; however, Kruskal-Wallis test is slightly superior to ANOVA and Brown-Forsythe test. The adjusted power of ANOVA seems not to be affected by the skewness and it performs best under the same condition when the design of sample size is unbalanced. When the data are generated from the negative skew normal distribution with heterogeneous variances, Kruskal-Wallis test performs best for low ($\mu_1 = 0, \mu_2 = 0.25, \mu_3 = 0.5$) and medium ($\mu_1 = 0, \mu_2 = 0.5, \mu_3 = 1$) effect sizes while Welch's heteroscedastic *F* test with trimmed mean and Winsorized variance has higher performance compared to other tests for high effect size ($\mu_1 = 0, \mu_2 = 1, \mu_3 = 2$).

It is noted that the Kruskal-Wallis test is affected by the skewness of the distribution especially under heterogeneity of variance ($\sigma_1 = 1$, $\sigma_2 = \sqrt{2}$, $\sigma_3 = 2$). Kruskal-Wallis test has the highest adjusted power when the data are generated from a negatively skewed distribution ($\gamma = -0.5$). On the other hand, it has the lowest adjusted power when the data are generated from a positively skewed distribution ($\gamma = 0.5$).

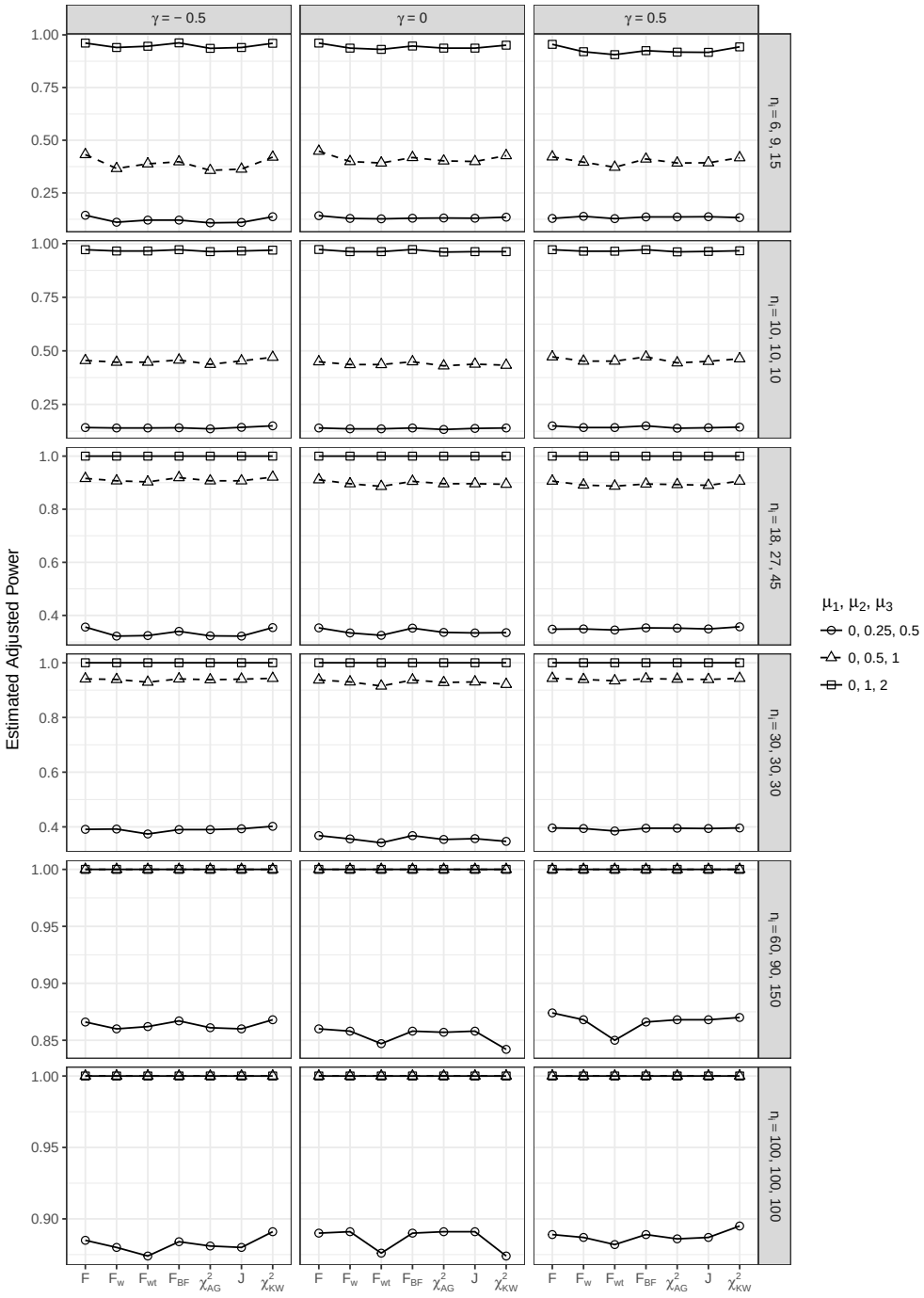


Figure 4: Estimated adjusted power results under homogeneous-variance ($\sigma_1 = \sigma_2 = \sigma_3 = 1$) scenarios

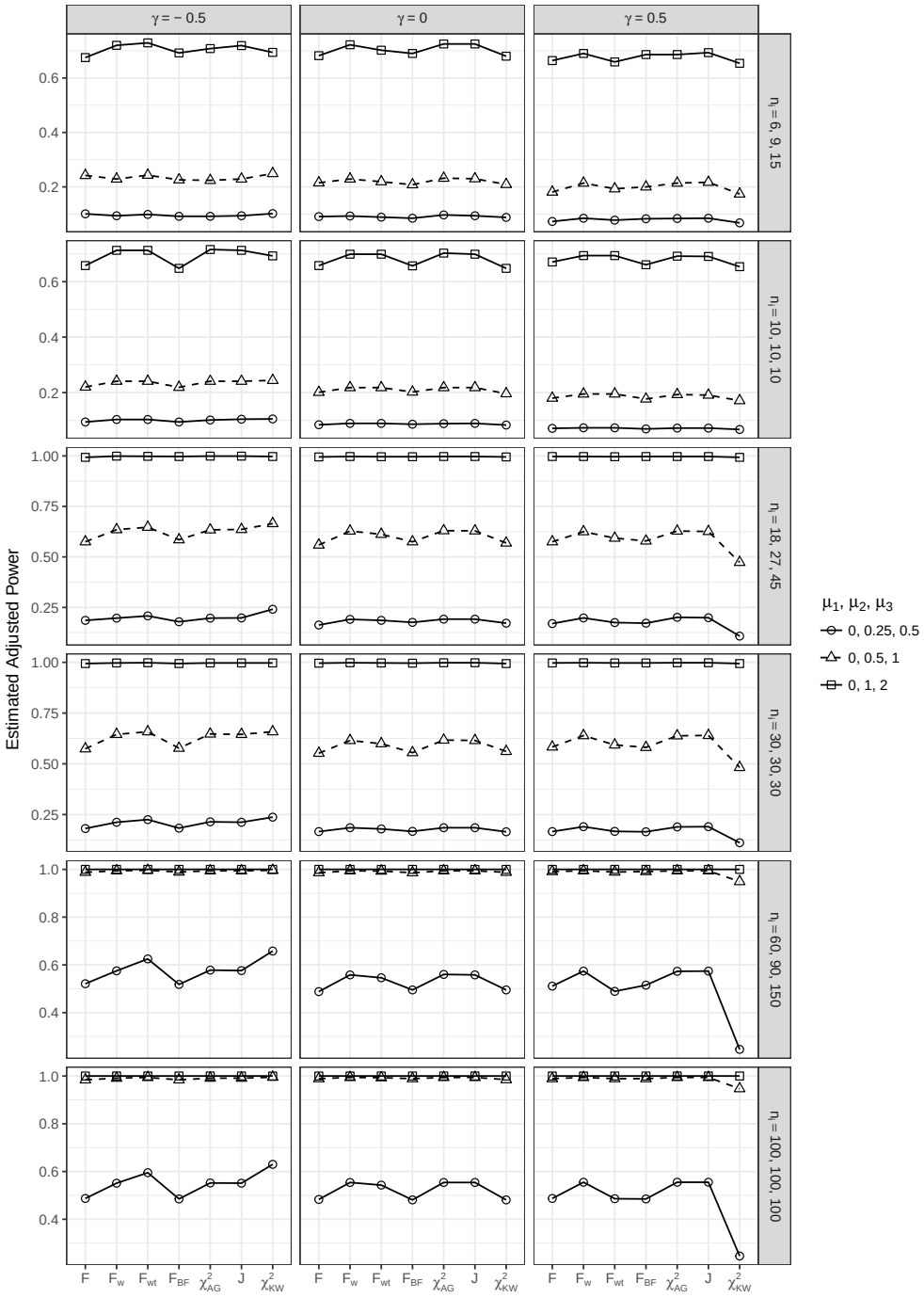


Figure 5: Estimated adjusted power results under heterogeneous-variance ($\sigma_1 = 1, \sigma_2 = \sqrt{2}, \sigma_3 = 2$) scenarios

Summary and further research

One-way tests in independent groups designs are the most commonly utilized statistical methods with applications on the experiments in medical sciences, pharmaceutical research, agriculture, biology, chemistry, engineering, and social sciences. In this paper, we present the **onewaytests** package for researchers to investigate treatment effects on the dependent variable.

The **onewaytests** package includes well-known one-way tests in independent groups designs including one-way analysis of variance, Kruskal-Wallis test, Welch heteroscedastic *F* test, Brown-Forsythe test. In addition to these well-known tests, Alexander-Govern test, James second order test

and Welch heteroscedastic F test with trimmed means and Winsorized variances, not available in most statistical software, are able to be reached via this package. Also, pairwise comparisons can be applied if the statistically significant difference is obtained.

Normality and variance homogeneity are the vital assumptions for the selection of the appropriate test. Therefore, this package also enables the users to check these assumptions via variance homogeneity and normality tests. It also enables the users some basic descriptive statistics and graphical approaches.

ANOVA is the most commonly used method for one-way independent groups designs. However, this method has two certain assumptions, homogeneity of variances and normality, to be satisfied. When these assumptions are not met, there are some other alternatives for one-way independent groups designs, which include Welch's heteroscedastic F test, Welch's heteroscedastic F test with trimmed means and Winsorized variances, Alexander-Govern test, James second order test, Brown-Forsythe test and Kruskal-Wallis test.

In this paper, we also compared seven one-way tests in **onewaytests** package with respect to the type I error rate and adjusted power. In the light of Monte Carlo simulation, it is noted that the normality assumption does not have a negative effect on the adjusted power of the tests as severe as variance homogeneity assumption. ANOVA is suggested to be used if the variance homogeneity assumption is held. Otherwise, Alexander-Govern, James second order and Welch's heteroscedastic F tests are recommended to be utilized. It is pointed out that under the negatively skew normal distribution with heterogeneous variances, Kruskal-Wallis test performs best with small and medium effect sizes while the Welch's heteroscedastic F test with trimmed means and Winsorized variances has the highest adjusted power with large effect size.

At present, the **onewaytests** package offers the one-way tests in independent groups designs, pairwise comparisons, graphical approaches, and assessment of variance homogeneity and normality via tests and plots. The package and its web-interface will be updated at regular intervals.

Acknowledgment

We thank the anonymous reviewers for their constructive comments and suggestions which helped us to improve the quality of our paper.

Bibliography

R. A. Alexander and D. M. Govern. A New and Simpler Approximation for ANOVA under Variance Heterogeneity. *Journal of Educational and Behavioral Statistics*, 19(2):91–101, 1994. URL https://doi.org/10.2307/1165140. [p175, 176, 178, 179]

G. Ambler and A. Benner. *Mfp: Multivariable Fractional Polynomials*, 2015. R package version 1.5.2. [p186]

E. Anderson. The Irises of the Gaspe Peninsula. *Bulletin of the American Iris society*, 59:2–5, 1935. [p180]

Y. Benjamini and Y. Hochberg. Controlling the False Discovery Rate: a Practical and Powerful Approach to Multiple Testing. *Journal of the Royal Statistical Society. Series B (Methodological)*, 57(1): 289–300, 1995. [p184]

Y. Benjamini and D. Yekutieli. The control of the false discovery rate in multiple testing under dependency. *The Annals of Statistics*, 29(4):1165–1188, 2001. URL https://doi.org/10.1214/aos/1013699998. [p184]

T. A. Bishop. *Heteroscedastic ANOVA, MANOVA, and Multiple-Comparisons*. PhD thesis, The Ohio State University, 1976. [p176]

M. B. Brown and A. B. Forsythe. The Small Sample Behavior of Some Statistics Which Test the Equality of Several Means. *Technometrics*, 16(1):129–132, 1974a. URL https://doi.org/10.2307/1267501. [p175, 176, 178]

M. B. Brown and A. B. Forsythe. Robust Tests for the Equality of Variances. *Journal of the American Statistical Association*, 69(346):364–367, 1974b. [p175, 176, 178]

W. Chang, J. Cheng, J. Allaire, Y. Xie, and J. McPherson. *shiny: Web Application Framework for R*, 2017. R package version 1.0.1. [p189]

R. A. Cribbie, R. R. Wilcox, C. Bewell, and H. Keselman. Tests for Treatment Group Equality When Data Are Nonnormal and Heteroscedastic. *Journal of Modern Applied Statistical Methods*, 6(1):117–132, 2007. URL https://doi.org/10.22237/jmasm/1177992660. [p175]

R. A. Cribbie, L. Fiksenbaum, H. Keselman, and R. R. Wilcox. Effect of Non-Normality on Test Statistics for One-Way Independent Groups Designs. *British Journal of Mathematical and Statistical Psychology*, 65(1):56–73, 2012. URL https://doi.org/10.1111/j.2044-8317.2011.02014.x. [p175]

O. Dag, A. Dolgun, and N. M. Konar. *Onewaytests: One-Way Tests in Independent Groups Designs*, 2017. R package version 1.5. [p175]

M. W. Fagerland and L. Sandvik. The Wilcoxon–Mann–Whitney Test under Scrutiny. *Statistics in Medicine*, 28(10):1487–1497, 2009. URL https://doi.org/10.1002/sim.3561. [p179]

R. A. Fisher. The Use of Multiple Measurements in Taxonomic Problems. *Annals of Eugenics*, 7(2): 179–188, 1936. URL https://doi.org/10.1111/j.1469-1809.1936.tb02137.x. [p180]

J. Fox and S. Weisberg. *An R Companion to Applied Regression*. Sage, Thousand Oaks CA, 2nd edition, 2011. [p179]

J. Gamage and S. Weerahandi. Size Performance of Some Tests in One-Way ANOVA. *Communications in Statistics-Simulation and Computation*, 27(3):625–640, 1998. URL https://doi.org/10.1080/03610919808813500. [p175]

J. L. Gastwirth, Y. R. Gel, W. L. W. Hui, V. Lyubchich, W. Miao, and K. Noguchi. *lawstat: Tools for Biostatistics, Public Policy, and Law*, 2017. R package version 3.1. [p179]

J. Gross and U. Ligges. *Nortest: Tests for Normality*, 2015. R package version 1.0-4. [p180]

Y. Hochberg. A Sharper Bonferroni Procedure for Multiple Tests of Significance. *Biometrika*, 75(4): 800–802, 1988. [p184]

S. Holm. A simple sequentially rejective multiple test procedure. *Scandinavian Journal of Statistics*, 6(2): 65–70, 1979. [p184]

G. Hommel. A Stagewise Rejective Multiple Test Procedure Based on a Modified Bonferroni Test. *Biometrika*, 75(2):383–386, 1988. [p184]

T. Hothorn, K. Hornik, M. A. Van De Wiel, and A. Zeileis. A Lego System for Conditional Inference. *The American Statistician*, 60(3):257–263, 2006. URL https://doi.org/10.1198/000313006X118430. [p179]

Y.-C. Hu. A new fuzzy-data mining method for pattern classification by principal component analysis. *Cybernetics and Systems: An International Journal*, 36(5):527–547, 2005. URL https://doi.org/10.1080/01969720590944294. [p180]

G. S. James. The comparison of several groups of observations when the ratios of the population variances are unknown. *Biometrika*, 38(3/4):324–329, 1951. URL https://doi.org/10.2307/2332578. [p175, 176, 178]

G. S. James. Tests of linear hypotheses in univariate and multivariate analysis when the ratios of the population variances are unknown. *Biometrika*, 41(1-2):19–43, 1954. URL https://doi.org/10.2307/2333003. [p175, 176]

H. Keselman, J. Algina, L. M. Lix, R. R. Wilcox, and K. N. Deering. A Generally Robust Approach for Testing Hypotheses and Setting Confidence Intervals for Effect Sizes. *Psychological Methods*, 13(2): 110–129, 2008. URL https://doi.org/10.1037/1082-989X.13.2.110. [p177]

S. Korkmaz, D. Goksuluk, and G. Zararsiz. MVN: An R Package for Assessing Multivariate Normality. *The R Journal*, 6(2):151–162, 2014. [p180]

W. H. Kruskal and W. A. Wallis. Use of Ranks in One-Criterion Variance Analysis. *Journal of the American Statistical Association*, 47(260):583–621, 1952. [p175, 176, 179]

B. Lantz. The Impact of Sample Non-Normality on ANOVA and Alternative Methods. *British Journal of Mathematical and Statistical Psychology*, 66(2):224–244, 2013. URL https://doi.org/10.1111/j.2044-8317.2012.02047.x. [p175, 176]

H. Liao, Y. Li, and G. Brooks. Outlier Impact and Accommodation Methods: Multiple Comparisons of Type I Error Rates. *Journal of Modern Applied Statistical Methods*, 15(1):23, 2016. URL https://doi.org/10.22237/jmasm/1462076520. [p188]

C. J. Lloyd. Estimating Test Power Adjusted for Size. *Journal of Statistical Computation and Simulation*, 75(11):921–933, 2005. URL https://doi.org/10.1080/00949650412331321160. [p189, 190]

P. Mair, F. Schoenbrodt, and R. Wilcox. *WRS2: Wilcox Robust Estimation and Testing*, 2017. 0.9-2. [p179]

L. Myers. Comparability of the James' Second-Order Approximation Test and the Alexander and Govern A Statistic for Non-Normal Heteroscedastic Data. *Journal of Statistical Computation and Simulation*, 60(3):207–222, 1998. URL https://doi.org/10.1080/00949659808811888. [p178]

T. Oshima and J. Algina. Type I Error Rates for James's Second-Order Test and Wilcox's Hm Test under Heteroscedasticity and Non-Normality. *British Journal of Mathematical and Statistical Psychology*, 45 (2):255–263, 1992. URL https://doi.org/10.1111/j.2044-8317.1992.tb00991.x. [p179]

I. Parra-Frutos. Testing Homogeneity of Variances with Unequal Sample Sizes. *Computational Statistics*, 28(3):1269–1297, 2013. URL https://doi.org/10.1007/s00180-012-0353-x. [p175]

R Core Team. *R: a Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria, 2017. URL https://www.R-project.org/. [p179, 183]

W. Sauerbrei, P. Royston, H. Bojar, C. Schmoor, M. Schumacher, G. B. C. S. Group, and others. Modelling the effects of standard prognostic factors in node-positive breast cancer. *British Journal of Cancer*, 79: 1752–1760, 1999. URL https://doi.org/10.1038/sj.bjc.6690279. [p186]

P. J. Villacorta. The welchADF Package for Robust Hypothesis Testing in Unbalanced Multivariate Mixed Models with Heteroscedastic and Non-Normal Data. *The R Journal*, 2017. [p179]

B. L. Welch. On the comparison of several mean values - an alternative approach. *Biometrika*, 38(3/4): 330–336, 1951. URL https://doi.org/10.2307/2332579. [p175, 176, 177]

H. Wickham. *Ggplot2: Elegant Graphics for Data Analysis*. Springer-Verlag, 2009. [p180]

R. R. Wilcox. A New Alternative to the ANOVA F and New Results on James's Second-Order Method. *British Journal of Mathematical and Statistical Psychology*, 41(1):109–117, 1988. URL https://doi.org/10.1111/j.2044-8317.1988.tb00890.x. [p175]

Osman Dag
Department of Biostatistics
Hacettepe University Faculty of Medicine, Sihhiye, Ankara
Turkey
osman.dag@hacettepe.edu.tr

Anil Dolgun
School of Science
RMIT University, Melbourne
Australia
anil.dolgun@rmit.edu.au

Naime Meric Konar
Department of Biostatistics
Hacettepe University Faculty of Medicine, Sihhiye, Ankara
Turkey
nmeric.konar@hacettepe.edu.tr

Supplementary Material

Table 2: Estimated type I error rates

Distribution	$\sigma_1, \sigma_2, \sigma_3$	n_1, n_2, n_3	F	F_w	F_{wt}	F_{BF}	χ^2_{AG}	J	χ^2_{KW}
Normal ($\gamma = 0$)	1, 1, 1	6, 9, 15	0.046	0.047	0.047	0.046	0.046	0.047	0.041
		10, 10, 10	0.052	0.050	0.050	0.050	0.049	0.051	0.047
		18, 27, 45	0.049	0.051	0.051	0.048	0.051	0.051	0.048
		30, 30, 30	0.051	0.052	0.053	0.051	0.052	0.052	0.051
		60, 90, 150	0.050	0.049	0.050	0.051	0.049	0.049	0.050
		100, 100, 100	0.051	0.050	0.052	0.051	0.050	0.050	0.050
	$1, \sqrt{2}, 2$	6, 9, 15	0.023	0.045	0.047	0.051	0.043	0.046	0.029
		10, 10, 10	0.058	0.051	0.051	0.053	0.050	0.052	0.054
		18, 27, 45	0.025	0.050	0.050	0.053	0.049	0.050	0.032
		30, 30, 30	0.058	0.052	0.051	0.056	0.051	0.052	0.055
		60, 90, 150	0.025	0.051	0.051	0.055	0.051	0.051	0.034
		100, 100, 100	0.059	0.052	0.052	0.059	0.052	0.052	0.059
Skew Normal ($\gamma = 0.5$)	1, 1, 1	6, 9, 15	0.052	0.055	0.053	0.051	0.056	0.055	0.047
		10, 10, 10	0.048	0.048	0.048	0.046	0.048	0.050	0.045
		18, 27, 45	0.053	0.054	0.052	0.052	0.053	0.054	0.050
		30, 30, 30	0.047	0.046	0.045	0.047	0.045	0.046	0.046
		60, 90, 150	0.046	0.046	0.050	0.046	0.046	0.046	0.049
		100, 100, 100	0.052	0.052	0.052	0.052	0.052	0.052	0.050
	$1, \sqrt{2}, 2$	6, 9, 15	0.026	0.052	0.051	0.053	0.052	0.053	0.032
		10, 10, 10	0.059	0.051	0.051	0.054	0.051	0.053	0.052
		18, 27, 45	0.023	0.051	0.053	0.055	0.050	0.051	0.038
		30, 30, 30	0.053	0.046	0.046	0.052	0.046	0.046	0.057
		60, 90, 150	0.022	0.048	0.052	0.052	0.048	0.048	0.052
		100, 100, 100	0.058	0.050	0.053	0.058	0.050	0.050	0.082
Skew Normal ($\gamma = -0.5$)	1, 1, 1	6, 9, 15	0.048	0.050	0.051	0.048	0.053	0.051	0.045
		10, 10, 10	0.051	0.048	0.048	0.048	0.048	0.048	0.043
		18, 27, 45	0.048	0.051	0.049	0.049	0.050	0.051	0.047
		30, 30, 30	0.048	0.047	0.048	0.048	0.047	0.047	0.045
		60, 90, 150	0.049	0.051	0.050	0.050	0.050	0.051	0.049
		100, 100, 100	0.053	0.054	0.054	0.053	0.053	0.054	0.052
	$1, \sqrt{2}, 2$	6, 9, 15	0.024	0.047	0.049	0.048	0.047	0.049	0.031
		10, 10, 10	0.058	0.051	0.051	0.054	0.050	0.052	0.053
		18, 27, 45	0.024	0.049	0.049	0.053	0.048	0.049	0.037
		30, 30, 30	0.054	0.048	0.048	0.052	0.047	0.048	0.059
		60, 90, 150	0.022	0.050	0.050	0.054	0.049	0.050	0.054
		100, 100, 100	0.058	0.054	0.055	0.058	0.054	0.054	0.085

Table 3: Adjusted power results

Distribution	$\sigma_1, \sigma_2, \sigma_3$	n_1, n_2, n_3	μ_1, μ_2, μ_3	F	F_w	F_{wt}	F_{BF}	χ^2_{AG}	J	χ^2_{KW}
Normal ($\gamma = 0$)	1, 1, 1	6, 9, 15	0, 0.25, 0.5	**0.141**	0.129	0.127	0.130	0.132	0.130	0.136
			0, 0.5, 1	**0.447**	0.400	0.392	0.417	0.404	0.399	0.428
			0, 1, 2	**0.961**	0.937	0.931	0.948	0.937	0.937	0.951
		10, 10, 10	0, 0.25, 0.5	0.140	0.136	0.136	**0.141**	0.133	0.138	0.139
			0, 0.5, 1	0.449	0.436	0.436	**0.451**	0.430	0.437	0.433
			0, 1, 2	**0.973**	0.963	0.963	**0.973**	0.961	0.964	0.963
		18, 27, 45	0, 0.25, 0.5	**0.353**	0.335	0.324	0.351	0.334	0.334	0.335
			0, 0.5, 1	**0.911**	0.897	0.886	0.905	0.895	0.896	0.894
			0, 1, 2	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**
		30, 30, 30	0, 0.25, 0.5	0.367	0.356	0.343	**0.368**	0.356	0.356	0.347
			0, 0.5, 1	**0.937**	0.930	0.915	**0.937**	0.929	0.930	0.920
			0, 1, 2	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**
		60, 90, 150	0, 0.25, 0.5	**0.861**	0.858	0.848	0.859	0.858	0.858	0.842
			0, 0.5, 1	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**
			0, 1, 2	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**
		100, 100, 100	0, 0.25, 0.5	**0.891**	0.890	0.877	**0.891**	**0.891**	0.890	0.874
			0, 0.5, 1	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**
			0, 1, 2	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**
	$1, \sqrt{2}, 2$	6, 9, 15	0, 0.25, 0.5	0.091	0.093	0.089	0.086	**0.096**	0.093	0.087
			0, 0.5, 1	0.214	0.228	0.220	0.209	**0.232**	0.229	0.207
			0, 1, 2	0.682	0.721	0.703	0.691	**0.724**	0.723	0.677
		10, 10, 10	0, 0.25, 0.5	0.084	**0.089**	**0.089**	0.086	0.087	**0.089**	0.082
			0, 0.5, 1	0.200	0.217	0.217	0.200	0.217	**0.219**	0.195
			0, 1, 2	0.657	0.697	0.697	0.655	**0.702**	0.700	0.647
		18, 27, 45	0, 0.25, 0.5	0.164	0.192	0.186	0.175	0.192	**0.193**	0.170
			0, 0.5, 1	0.562	0.629	0.612	0.574	**0.630**	**0.630**	0.568
			0, 1, 2	0.995	**0.997**	0.996	0.996	**0.997**	**0.997**	0.995
		30, 30, 30	0, 0.25, 0.5	0.167	0.185	0.180	0.166	**0.187**	0.185	0.165
			0, 0.5, 1	0.553	0.615	0.600	0.554	**0.619**	0.615	0.561
			0, 1, 2	0.996	0.997	0.997	0.996	**0.998**	0.997	0.994
		60, 90, 150	0, 0.25, 0.5	0.491	0.559	0.545	0.496	0.558	**0.560**	0.496
			0, 0.5, 1	0.987	**0.994**	0.992	0.987	**0.994**	**0.994**	0.988
			0, 1, 2	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**
		100, 100, 100	0, 0.25, 0.5	0.483	0.553	0.545	0.482	**0.554**	0.553	0.483
			0, 0.5, 1	0.988	**0.994**	0.992	0.988	**0.994**	**0.994**	0.985
			0, 1, 2	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**
Skew Normal ($\gamma = 0.5$)	1, 1, 1	6, 9, 15	0, 0.25, 0.5	0.128	**0.137**	0.127	0.135	0.136	**0.137**	0.134
			0, 0.5, 1	**0.419**	0.395	0.372	0.410	0.392	0.393	0.418
			0, 1, 2	**0.955**	0.919	0.906	0.924	0.918	0.917	0.944
		10, 10, 10	0, 0.25, 0.5	**0.150**	0.141	0.141	0.149	0.138	0.141	0.144
			0, 0.5, 1	**0.472**	0.450	0.450	0.470	0.443	0.450	0.463
			0, 1, 2	**0.972**	0.964	0.964	0.971	0.961	0.964	0.967
		18, 27, 45	0, 0.25, 0.5	0.348	0.350	0.345	0.352	0.351	0.350	**0.359**
			0, 0.5, 1	**0.907**	0.891	0.887	0.895	0.892	0.891	**0.907**
			0, 1, 2	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**
		30, 30, 30	0, 0.25, 0.5	0.397	0.395	0.386	0.397	0.396	0.395	**0.398**
			0, 0.5, 1	0.942	0.939	0.934	0.942	0.940	0.939	**0.943**
			0, 1, 2	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**
		60, 90, 150	0, 0.25, 0.5	**0.875**	0.868	0.851	0.867	0.869	0.868	0.870
			0, 0.5, 1	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**
			0, 1, 2	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**
		100, 100, 100	0, 0.25, 0.5	0.890	0.887	0.882	0.890	0.887	0.887	**0.894**
			0, 0.5, 1	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**
			0, 1, 2	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**

Table 4: Adjusted power results continued

Distribution	$\sigma_1, \sigma_2, \sigma_3$	n_1, n_2, n_3	μ_1, μ_2, μ_3	F	F_w	F_{wt}	F_{BF}	χ^2_{AG}	J	χ^2_{KW}
Skew Normal ($\gamma = 0.5$)	1, $\sqrt{2}$, 2	6, 9, 15	0, 0.25, 0.5	0.072	**0.085**	0.077	0.083	**0.085**	**0.085**	0.068
			0, 0.5, 1	0.181	**0.216**	0.193	0.199	0.215	**0.216**	0.172
			0, 1, 2	0.664	0.691	0.659	0.684	0.687	**0.693**	0.652
		10, 10, 10	0, 0.25, 0.5	0.072	0.072	0.072	0.069	0.072	**0.073**	0.067
			0, 0.5, 1	0.180	**0.194**	**0.194**	0.176	**0.194**	0.192	0.170
			0, 1, 2	0.672	**0.693**	**0.693**	0.661	**0.693**	0.692	0.653
		18, 27, 45	0, 0.25, 0.5	0.169	0.199	0.174	0.171	**0.200**	0.198	0.109
			0, 0.5, 1	0.573	0.625	0.591	0.579	**0.627**	0.625	0.475
			0, 1, 2	**0.997**	**0.997**	0.995	0.996	**0.997**	**0.997**	0.993
		30, 30, 30	0, 0.25, 0.5	0.167	**0.189**	0.167	0.166	0.188	**0.189**	0.111
			0, 0.5, 1	0.584	**0.639**	0.593	0.582	0.638	**0.639**	0.483
			0, 1, 2	0.997	**0.998**	0.997	0.997	**0.998**	**0.998**	0.994
		60, 90, 150	0, 0.25, 0.5	0.514	**0.573**	0.489	0.516	**0.573**	**0.573**	0.246
			0, 0.5, 1	0.991	**0.994**	0.989	0.991	**0.994**	**0.994**	0.949
			0, 1, 2	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**
		100, 100, 100	0, 0.25, 0.5	0.487	**0.554**	0.486	0.487	**0.554**	**0.554**	0.245
			0, 0.5, 1	0.990	**0.994**	0.990	0.990	**0.994**	**0.994**	0.946
			0, 1, 2	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**
Skew Normal ($\gamma = -0.5$)	1, 1, 1	6, 9, 15	0, 0.25, 0.5	**0.144**	0.111	0.120	0.120	0.107	0.111	0.135
			0, 0.5, 1	**0.432**	0.364	0.386	0.396	0.355	0.365	0.417
			0, 1, 2	0.961	0.940	0.946	**0.962**	0.935	0.940	0.959
		10, 10, 10	0, 0.25, 0.5	0.142	0.142	0.142	0.142	0.138	0.142	**0.150**
			0, 0.5, 1	0.455	0.449	0.449	0.457	0.439	0.451	**0.470**
			0, 1, 2	**0.972**	0.966	0.966	**0.972**	0.963	0.966	0.970
		18, 27, 45	0, 0.25, 0.5	**0.358**	0.322	0.325	0.340	0.322	0.322	0.356
			0, 0.5, 1	0.917	0.908	0.903	0.920	0.907	0.908	**0.921**
			0, 1, 2	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**
		30, 30, 30	0, 0.25, 0.5	0.391	0.392	0.376	0.390	0.390	0.393	**0.401**
			0, 0.5, 1	0.941	0.939	0.929	0.941	0.937	0.939	**0.943**
			0, 1, 2	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**
		60, 90, 150	0, 0.25, 0.5	0.867	0.860	0.862	**0.868**	0.861	0.860	0.867
			0, 0.5, 1	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**
			0, 1, 2	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**
		100, 100, 100	0, 0.25, 0.5	0.884	0.880	0.874	0.884	0.880	0.880	**0.892**
			0, 0.5, 1	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**
			0, 1, 2	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**
	1, $\sqrt{2}$, 2	6, 9, 15	0, 0.25, 0.5	0.101	0.094	0.098	0.092	0.091	0.094	**0.102**
			0, 0.5, 1	0.240	0.229	0.243	0.225	0.223	0.230	**0.249**
			0, 1, 2	0.672	0.720	**0.728**	0.691	0.707	0.720	0.694
		10, 10, 10	0, 0.25, 0.5	0.094	0.103	0.103	0.094	0.102	**0.104**	**0.104**
			0, 0.5, 1	0.220	0.240	0.240	0.220	0.242	0.242	**0.243**
			0, 1, 2	0.658	0.713	0.713	0.649	**0.717**	0.713	0.692
		18, 27, 45	0, 0.25, 0.5	0.188	0.197	0.207	0.178	0.197	0.197	**0.242**
			0, 0.5, 1	0.579	0.635	0.645	0.582	0.633	0.636	**0.666**
			0, 1, 2	0.993	**0.999**	**0.999**	0.996	**0.999**	**0.999**	0.998
		30, 30, 30	0, 0.25, 0.5	0.180	0.213	0.225	0.183	0.213	0.212	**0.238**
			0, 0.5, 1	0.574	0.646	0.658	0.576	0.647	0.646	**0.659**
			0, 1, 2	0.994	0.997	**0.998**	0.994	0.997	0.997	0.997
		60, 90, 150	0, 0.25, 0.5	0.518	0.576	0.624	0.517	0.577	0.576	**0.659**
			0, 0.5, 1	0.988	0.994	**0.996**	0.988	0.994	0.994	**0.996**
			0, 1, 2	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**
		100, 100, 100	0, 0.25, 0.5	0.487	0.552	0.595	0.486	0.552	0.552	**0.630**
			0, 0.5, 1	0.984	0.992	0.994	0.984	0.992	0.991	**0.995**
			0, 1, 2	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**

Inventorymodel: an R Package for Centralized Inventory Problems

by Alejandro Saavedra-Nieves

Abstract Inventory management of goods is an integral part of logistics systems; common to various economic sectors such as industry, agriculture and trade; and independent of production volume. In general, as companies seek to minimize economic losses, studies on problems of multi-agent inventory have increased in recent years. A multi-agent inventory problem is a situation in which several agents face individual inventory problems and agree to coordinate their orders with the objective of reducing their costs. The R package **Inventorymodel** allows the determination of both the optimal policy for some inventory situations with deterministic demands and the allocation of costs from a game-theoretic perspective. The required calculations may be computed for any number of agents although the computational complexity of this class of problems when the involved agents enlarge is not reduced. In this work, the different possibilities that the package offers are described and some examples of usage are also demonstrated.

Introduction

Often, once a product is chosen to be acquired, the seller advises that he or she does not have a unit of that product in stock. Probably, this vendor has not managed the storage of this commodity efficiently since the possibility of not being able to satisfy the demand for a product in time leads to shortages. Specific techniques may prevent situations such as this, allowing for reduction in economic losses associated with the lack of products.

Inventory is defined as the amount of goods stored by a company at a moment of time, which represents a capital investment for the purchase and maintenance of the product in stock. Each company desires having the proper inventory level to meet the needs of customers, and so they would prefer to handle it optimally. Inventory theory focuses on the design and optimization of production systems to minimize economic losses due to product shortages. However, this implies the existence of extra costs associated with different assumptions imposed on the model with an influence on the overall cost of the procedure. The consideration of the holding cost or variable cost for the transportation of a new order from a supplier to the warehouse are examples, among others, of new conditions to take into account in the models. Generally, this study focuses on a single product with fixed characteristics that companies need. Moreover, it presents new inventory situations in which a customer can switch to another supplier to get the desired product or an analogous product with similar properties. So, in an inventory system, the following must be considered:

- An agent that wants to organize its inventory of a certain commodity.
- A deterministic demand for a product that this agent needs over a fixed period of time.
- The agent satisfies its demand through an external provider.
- In some circumstances, the agent can consider the possibility of not fulfilling all the demand in time, that is, a shortage of the good is allowed.

The main goal entails determining the optimal amount of product to request and the moment in which it is requested. The balance between quantity and the number of orders is essential. Not following the optimal policy implies that fewer orders than necessary are placed or that they are placed more frequently than usual, and hence the cost of the procedure increases.

Usually, we talk about inventory management for a single company, with a single decision determining the amount required depending on the demand or, for instance, the moment in which that order must be placed. Although individually, such inventory models totally ignore the decisions made by other agents in the same circumstances, it is possible to consider situations wherein they cooperate. Sometimes, in order to obtain savings, it is possible to consider the joint action of several individuals who, independently, would face situations of inventory management of a product as described.

In a *centralized inventory problem*, several agents who face individual inventory problems decide to coordinate their orders to reduce costs. The determination of the optimal level of inventory involves:

(a) formulation of an appropriate mathematical model that describes the situation of inventory (shortages, transportation costs, holding costs, etc.),

(b) identification of the optimal policy under the cooperation of a group of individuals, and

(c) allocation of the incurred costs among the involved agents by using, for instance, game theoretic procedures.

The analysis of cooperative multi-agent inventory problems has received much attention in recent years. The most basic models, the *Economic Order Quantity* (EOQ) model and the *Economic Production Quantity* (EPQ) model, are introduced in Meca et al. (2003) for the cooperative case, using the Share the Ordering Costs rule (*SOC rule*) as a procedure to obtain allocations with interesting properties. However, other inventory problems have been treated in the literature under this approach, including those detailed below. In Fiestras-Janeiro et al. (2012), the *inventory transportation systems* are defined as a new approach of the EOQ problem when individual transportation costs by a new order are taken into account. The *line-rule* allows us to allocate the costs generated by the joint order in these situations. This model is generalized in Saavedra-Nieves et al. (2018), where a general function of transportation costs is considered and an extension of the line-rule is proposed. Fiestras-Janeiro et al. (2014) describe two *EOQ models with no holding costs*: a basic model and a model with transportation costs. The *Shapley value* and the *two-lines rule* are the chosen allocation procedures for sharing costs in this class of situations. Fiestras-Janeiro et al. (2015) introduce a cooperative multi-agent inventory model, where the holding costs are negligible and shortages are allowed. Saavedra-Nieves et al. (2018a) introduce a variant of the previous model by considering that shortages are cheaper than the original product. For these models, two rules are proposed to allocate the collaborating costs. Apart from those mentioned above, Nagarajan and Sošić (2008), Dror and Hartman (2011), and Fiestras-Janeiro et al. (2011) are examples of some recent surveys in this field.

This paper describes the R package **Inventorymodel** and its capabilities for determining the optimal policies in the mentioned inventory models under a game-theoretic approach. For each of them, an allocation procedure is implemented to share the collaborating costs. This package can be very useful when the number of involved agents increases due to the computational complexity of the calculus of all possible groups considerably increases.

The remainder of this paper is organized as follows. First, we introduce some game theoretic notations used in this class of problems. Then, we present the different inventory models included in the **Inventorymodel** package, describing, for each, the optimal policy and appropriate allocation rule for the collaborating costs. Usage of the package is illustrated through the analysis of different examples considered in the literature.

Preliminaries on game theory

In this section, we introduce some necessary theoretical game terminology.

The *cost games* correspond to one of the most important classes of cooperative games. They are necessary for analysing those cost situations where n agents collaborate. Its definition is based on the concept of *coalition*. In what follows, if $N := \{1, ..., n\}$ denotes the set of agents which cooperate, a coalition of N is a subset $S \subseteq N$ with $s := |S|$ elements. Following González-Díaz et al. (2010), a cost game is a pair (N, c), where N is the set of players and c is a *characteristic function*, a map from 2^N to $\mathbb{R}$ satisfying that $c(\emptyset) = 0$. Usually, $c(S)$ represents the cost of the collaboration of the players in the coalition S, for every $S \subseteq N$.

In this class of situations, some natural properties to verify are the following. We say that (N, c) is *subadditive* if, for each $S, T \subseteq N$, with $S \cap T = \emptyset$, it satisfies $c(S) + c(T) \geq c(S \cup T)$. Subadditivity implies that agents in $S \cup T$ do have economic incentives to collaborate with respect to the cooperation of the groups S and T independently. As consequence, the formation of coalition N is profitable. A game (N, c) is *concave* if, for all $i \in N$ and for all $S \subset T \subseteq N \setminus \{i\}$, it satisfies $c(S \cup \{i\}) - c(S) \geq c(T \cup \{i\}) - c(T)$. Under concavity, players will have more incentive to join the coalition with a lower cost to contribute.

When a set of N agents cooperate, an important issue is the definition of procedures to allocate the cost $c(N)$ obtained by collaborating among the involved agents. Below, we introduce two of the most important classes of allocations in this context according to different criteria in order to define the new allocation rules. The first approach focuses on the equanimity of allocations between players that cooperate. The *Shapley value*, introduced in Shapley (1953), is an example of a rule defined under this criteria to distribute $c(N)$. The Shapley value of (N, c), $\Phi(N, c)$, is defined for each $i \in N$ by:

$$\Phi_i(N, c) := \frac{1}{n!} \sum_{\sigma \in \Pi(N)} m_i^\sigma(c),$$

where $\Pi(N)$ denotes the set of permutations of N, $P_i^\sigma := \{j \in N : \sigma(j) < \sigma(i)\}$ denotes the predecessors of agent i by a permutation σ, and $m_i^\sigma(c) := c(P_i^\sigma \cup i) - c(P_i^\sigma)$ for each $\sigma \in \Pi(N)$. The Shapley value assigns to each $i \in N$ its expected value if an agreement is reached, and each agent has to pay its contribution in the costs to the indicated set of predecessors. It is assumed that the arrival order is random and all orders are equally likely.

The second criteria is represented by the stability of the allocations. The *core* of (N, c), as in Gillies

(1953), is the following set of payoff vectors for the members of N:

$$C(N,c) := \left\{ x \in \mathbb{R}^N : \sum_{i \in N} x_i = c(N), \text{ and for each non empty coalition } S \subset N, \sum_{i \in S} x_i \leq c(S) \right\}.$$

It is said that the core allocations are *stable* in the sense that abandoning N and acting independently represents no improvement (in terms of gains) to coalition S, with $S \subset N$. This is due the fact that agents in S will pay more, when N does not form, than in those situations where its members act jointly with the rest of agents in $N \setminus S$. The study of the core of a game (N,c) and its cardinality become a complex issue in game theory since the number of associated constraints increases when n enlarges. For this reason, the definition of rules whose allocations are stable is a main goal in this class of problems. Under the verification of some properties of (N,c), it is easy to study the core. In particular, if the cost game is concave, the Shapley value of (N,c) is an element of this set, and when (N,c) is subadditive, $C(N,c)$ is a non-empty set.

Centralized inventory problems

The **Inventorymodel** package includes functions to obtain the optimal policy in this class of situations. In particular, different scenarios with deterministic demands are considered: for example, shortages can be allowed, transportation costs can be taken into account by assuming that agents are in a line or a general route, or different costs of product during shortages are allowed.

Depending on the considered assumptions, different parameters should be selected for modelling the problem among those described below. If a set of agents N cooperate to place joint orders, we can distinguish the following parameters:

- $a > 0$, the fixed part of the *ordering cost* that is independent of the size of a new order,
- $a_i > 0$, the variable part of the *ordering cost* that depends on the distance between i and the supplier,
- $d_i > 0$, the deterministic *demand* of the product of agent i,
- $h_i > 0$, a *holding cost* by unit of a product stored in the own warehouse,
- $b_i > 0$, the *shortage cost* per item and per time unit,
- $K_i > 0$, the *capacity* of the warehouse of agent i.

This section introduces the different inventory models included in our package, which are dealt with in the literature by considering several assumptions about them.[1]

The EOQ and EPQ models under cooperation

The *EOQ model under cooperation*, introduced in Meca et al. (2003), establishes the optimal policy for an agent $i \in N$ to satisfy its demand of a product. This model supposes a fixed *ordering cost*, $a > 0$, and, for each $i \in N$, a deterministic *demand* of $d_i > 0$ units of product per time unit and a *holding cost* by unit, $h_i > 0$. For this model, it is assumed that shortages are not allowed.

If Q_i denotes the required size of the order to satisfy d_i, the time between two consecutive orders (length of the cycle) is Q_i/d_i. If only one agent acts, we deal with a basic EOQ problem, a well-known problem in operations research. The optimal policy is simple in this case: once the stock runs out, this agent places a new order. When a group of agents $S \subseteq N$ decides to cooperate, instead of ordering independently, it applies for a unique joint order. Moreover, all agents should order at the same time, and thus the lengths of the individual cycles coincide as well (this is, $Q_i/d_i = Q_j/d_j$ for each pair of agents $i, j \in S$). In addition, if we define the number of orders per time unit by $m_i = d_i/Q_i$, in the cooperative case we have that $m_i = m_j$ for each $i, j \in S$. Therefore, it is easy to check that the average cost per cycle is

$$C^S(\{Q_i\}_{i \in S}) := a\frac{d_k}{Q_k} + \sum_{i \in S} h_i \frac{Q_i}{2}, \text{ for any } k \in S. \tag{1}$$

Using optimization techniques, (1) takes its minimum value at $Q_i^* := \sqrt{\frac{2ad_i^2}{\sum_{j \in S} h_j d_j}}$ for each $i \in S$. Hence,

[1] To simplify the description of the different inventory models contained in this package, we only consider the cooperative case when the formation of a certain group $S \subseteq N$ is assumed. Moreover, this generalizes the case where a single agent $i \in N$ acts, by taking $S = \{i\}$.

we can associate a cost game (N, c), where $c(S) = C^S(\{Q_i^*\}_{i\in S})$ for all $S \subseteq N$. This is

$$c(S) := a\frac{d_k}{Q_k^*} + \sum_{i\in S} h_i \frac{Q_i^*}{2} = 2a\sqrt{\sum_{j\in S} m_j^{*2}}, \tag{2}$$

where $m_i^* := \frac{d_i}{Q_i^*}$ denotes the optimal number of orders per time unit for each $i \in S$.

The *EPQ model under cooperation* is introduced by Meca et al. (2003) by adding the following assumptions to those for the the EOQ model:

- Each agent i of group S faces the possibility of not fulfilling its demand on time by incurring a level of shortage of β_i units. The cost by shortage unit is $b_i > 0$.
- Once the agents order, in a unit of time, agent i obtains r_i units of the product, and the replacement rate of agent i with $r_i > d_i$.

When an agent acts individually, the optimal policy is simple as well: once the stock runs out and the maximum level of shortage is reached, this agent places a new order. Following Meca et al. (2003), when a group $S \subseteq N$ is formed, its members apply for new joint orders. The average cost per cycle is

$$C^S(\{Q_i\}_{i\in S}, \{\beta_i\}_{i\in S}) := a\frac{d_k}{Q_k} + \sum_{j\in S} h_j \frac{(Q_j(1-\frac{d_j}{r_j}) - \beta_j)^2}{2Q_j(1-\frac{d_j}{r_j})} + \sum_{j\in S} b_j \frac{\beta_j^2}{2Q_j(1-\frac{d_j}{r_j})}, \text{ for any } k \in S. \tag{3}$$

So, (3) reaches the minima in $Q_i^* := \sqrt{\frac{2ad_i^2}{\sum_{j\in S} d_j h_j \frac{b_j}{h_j+b_j}(1-\frac{d_j}{r_j})}}$ and $\beta_i^* := Q_i^* \frac{h_i(1-\frac{d_i}{r_i})}{h_i+b_i}$. With the appropriate calculations, we can associate a cost game (N, c) to the EPQ model by defining, for each $S \subseteq N$,

$$c(S) := C^S(\{Q_i^*\}_{i\in S}, \{\beta_i^*\}_{i\in S}) = 2a\sqrt{\sum_{j\in S} m_j^{*2}}, \tag{4}$$

where $m_i^* := \frac{d_i}{Q_i^*}$ denotes the optimal number of orders per time unit for each $i \in S$.

The cooperative inventory cost situations defined in this section are denoted by (N, a, m). The class of the obtained games is called *inventory games*, verifying that for each $S \subseteq N$, $c(S) = 2a\sqrt{\sum_{j\in S} m_j^{*2}}$ and also $c^2(S) = \sum_{i\in S} c(\{i\})^2$.

The *SOC rule*, introduced by Meca et al. (2003), establishes proportional allocations in situations such as detailed. Let (N, a, m) be a situation of inventory as defined, and let (N, c) be the associated cost game. When N cooperate, the *SOC rule* σ proposes to each agent $i \in N$ a core allocation that includes a part of the fixed order cost (proportional to m_i^2), a part of the holding costs and, in the EPQ model, a part relative to shortages. Hence, when N cooperates, for each $i \in N$,

$$\sigma_i(N, c) := \frac{2am_i^2}{\sqrt{\sum_{j\in N} m_j^2}}, \tag{5}$$

while $m_j \neq 0$ for $j \in N$. Otherwise, $\sigma_i(N, c) = 0$. Following Meca et al. (2003), as $\sum_{i\in S} \sigma_i(N, c) \leq c(S)$ for all $S \subset N$ and $\sum_{i\in N} \sigma_i(N, c) = c(N)$, the stability of the allocations is ensured. It is remarkable that the calculation of $\sigma_i(N, c)$ does not require the determination, in a explicit way, of the complete characteristic function of (N, c).

The inventory transportation systems

An *inventory transportation system* is an inventory situation wherein agents face a basic EOQ model. As Fiestras-Janeiro et al. (2012) propose, the cost of a new order has two components: the first is common to all and the second is proportional to the distance between the supplier and the agent.

Let $N := \{1, \ldots, n\}$ be the set of agents. We consider as parameters the fixed *ordering cost*, $a > 0$, and for each $i \in N$, the variable *ordering cost*, $a_i > 0$; the deterministic *demand* of the product, $d_i > 0$; and the *holding cost* by unit stored by the agent i, $h_i > 0$. Let $(N, a, \{a_i, d_i, h_i\}_{i\in N})$ be an inventory transportation system. The formation of a group $S \subseteq N$ implies that, instead of ordering independently, agents apply for a unique joint order. As we mention, all agents have to order at same time, and thus the lengths of the individual cycles are equal. In addition, in this model, we assume the following:

- All agents are in a same route. If S places a new order, the ordering cost is equal to, first, a and, second, the maximal distance a_S between the agents in S and the supplier. So, $a_S := \max\{a_i :$

$i \in S\}$.

- The supplier allows the formation of ordering coalitions. Given $S \subseteq N$, the cost of a new order of S is $a + a_S$, although in some specific order, an agent in S may not buy the product.

According to Fiestras-Janeiro et al. (2012), the average cost per cycle is determined by

$$C^S(\{Q_i\}_{i\in S}) := (a + a_S)\frac{d_k}{Q_k} + \sum_{j\in S} h_j \frac{Q_j}{2} = (a + a_S)\frac{d_k}{Q_k} + \frac{Q_i}{2d_i}\sum_{j\in S} d_j h_j, \text{ for any } k \in S.$$

After some algebraic calculation, when $S \subseteq N$ is formed, the optimal size of the order for $i \in S$ is $Q_i^* := \sqrt{\frac{2(a+a_S)d_i^2}{\sum_{j\in S} d_j h_j}}$, which is the optimal number of orders per time unit $m_S^* := \frac{d_i}{Q_i^*} = \sqrt{\frac{\sum_{j\in S} d_j h_j}{2(a+a_S)}}$. So, for an inventory transportation system, we can define a cost game (N, c) for $S \subseteq N$ by

$$c(S) := C^S(\{Q_i^*\}_{i\in S}) = \sqrt{2(a + a_S)\sum_{j\in S} d_j h_j} = 2(a + a_S)m_S^*. \tag{6}$$

Fiestras-Janeiro et al. (2012) propose the *line-rule* $L(N, c)$ as the procedure to share the collaborating costs. This rule assigns to each $i \in N$

$$L_i(N, c) := \frac{1}{|\Pi_L(N, c)|}\sum_{\sigma\in\Pi_L(N,c)} m_i^\sigma(N, c), \tag{7}$$

where $\Pi_L(N, c)$ denotes the set of permutations of N that reverse the order given by the distances from the agents to the supplier. Formally, $\Pi_L(N, c) := \{\sigma \in \Pi(N) : \sigma(i) \leq \sigma(j) \Rightarrow a_i \geq a_j\}$. So, this rule can be seen as a variation of the Shapley value for cost games. It requires less computational effort than the original, and Fiestras-Janeiro et al. (2012) prove that it is an element of the core of the game using the subadditivity character of (N, c) under these assumptions.

The inventory problems with no-holding costs and without shortages

A *basic EOQ system without holding costs* is a model with a deterministic demand, no holding costs, and a limited capacity of the warehouses and shortages are not allowed. Let $N := \{1, \ldots, n\}$ be the set of agents. Following Fiestras-Janeiro et al. (2014), the parameters associated with each $i \in N$ in these systems are the fixed *ordering cost*, $a > 0$; the deterministic *demand* of the product of agent i, $d_i > 0$; and the *capacity* of the warehouse of i, $K_i > 0$.

We denote these systems by $(N, a, \{d_i, K_i\}_{i\in N})$. The optimal policy of a group $S \subseteq N$ is very simple. When a group of agents S is formed, the length of a joint cycle is $\min_{j\in S}\{K_j/d_j\}$ and its associated cost per time unit is

$$\frac{\text{cost of a cycle}}{\text{length of a cycle}} = \frac{a}{\min_{j\in S}\{K_j/d_j\}} = a\max_{j\in S}\{\frac{d_j}{K_j}\}.$$

Thus, we can associate with every basic EOQ system without holding costs and with general transportation costs $(N, a, A, \{d_i, K_i\}_{i\in N})$ the cost game (N, c) which is given for all $S \subseteq N$ by

$$c(S) := a\max_{j\in S}\{\frac{d_j}{K_j}\}. \tag{8}$$

Notice that the obtained cost game can be seen as an airport game (see Littlechild and Owen (1973), for details). For it, the Shapley value is an excellent rule with stable allocations. Moreover, there is a simple procedure to obtain it. By ordering in a non-decreasing order, the ratio d_i/K_i for all $i \in N$, the expression of the Shapley value for agent $i \in N$, $\Phi_i(N, c)$, is

$$\Phi_i(N, c) := \frac{c(\{1\})}{n} + \sum_{j=2}^{i}\frac{c(\{j\}) - c(\{j-1\})}{n - j + 1}. \tag{9}$$

Below, we describe an inventory system with costs for transporting a new order.

A *basic EOQ system without holding costs and with transportation costs* is an inventory situation introduced by Fiestras-Janeiro et al. (2014), where agents face a basic EOQ model. Let $N := \{1, \ldots, n\}$ be the set of agents. For each $i \in N$, the following parameters are considered: the fixed part of the *ordering cost*, $a > 0$; the variable part of the *ordering cost*, $a_i > 0$; the deterministic *demand* of the product of i, $d_i > 0$; and the *capacity* of the warehouse of i, $K_i > 0$.

These models are denoted by $(N, a, \{a_i, d_i, K_i\}_{i\in N})$. The formation of a group $S \subseteq N$ implies that agents in S place a unique joint order instead of independent orders. All of them have to order at the same time, and thus the lengths of the individual cycles are equal. In addition, we consider the following two assumptions imposed on the inventory transportation systems (see Fiestras-Janeiro et al. (2012), for instance):

- All agents are in the same route. When a group S is formed, the ordering cost is equal to, first, a fixed value a and, second, the maximal distance a_S of agents in S to the supplier. Thus, $a_S := \max\{a_i : i \in S\}$.
- The supplier allows the formation of ordering coalitions. Given $S \subseteq N$, the cost of a new order of S is $a + a_S$, regardless of whether an agent in S does not buy the product.

The optimal policy of a group $S \subseteq N$ in this context is simple. When agents in S place joint orders, since shortages are not allowed, the length of a cycle is $\min_{j\in S}\{K_j/d_j\}$ and the minimum joint cost per time unit is

$$\frac{\text{cost of a cycle}}{\text{length of a cycle}} = \frac{a + a_S}{\min_{j\in S}\{K_j/d_j\}} = (a + a_S)\max_{j\in S}\{\frac{d_j}{K_j}\}.$$

Thus, we can associate with basic EOQ system without holding costs and with transportation costs $(N, a, \{a_i, d_i, K_i\}_{i\in N})$ the cost game (N, c), which is given for all $S \subseteq N$ by

$$c(S) := (a + a_S)\max_{j\in S}\{\frac{d_j}{K_j}\}. \tag{10}$$

For each system $(N, a, \{a_i, d_i, K_i\}_{i\in N})$, the *two-lines rule* assigns the cost allocation $TL(N, c) := (TL(N, c))_{i\in N}$. Following Fiestras-Janeiro et al. (2014), it belongs to the core of (N, c) and is given, for each $i \in N$, by

$$TL_i(N, c) := \frac{1}{2|\Pi_1(N, c)|}\sum_{\sigma\in\Pi_1(N,c)} m_i^\sigma(N, c) + \frac{1}{2|\Pi_2(N, c)|}\sum_{\sigma\in\Pi_2(N,c)} m_i^\sigma(N, c), \tag{11}$$

where $\Pi_1(N, c)$ and $\Pi_2(N, c)$ are the following sets:

- $\Pi_1(N, c) := \{\sigma \in \Pi(N) : a_i \geq a_j \Rightarrow \sigma(i) \leq \sigma(j), \text{ for all } i, j \in N\}$. It is composed of permutations that reverse the order given by the transportation costs.
- $\Pi_2(N, c) := \{\sigma \in \Pi(N) : \frac{d_i}{K_i} \geq \frac{d_j}{K_j} \Rightarrow \sigma(i) \leq \sigma(j), \text{ for all } i, j \in N\}$. This contains the permutations that reverse the order given by the *ratio* of demand/capacity.

Below, the assumption of a general transportation cost function is introduced in the previous model. Saavedra-Nieves et al. (2018) analyse the *EOQ systems without holding costs and with general transportation costs*, a generalization of the basic model with transportation costs proposed in Fiestras-Janeiro et al. (2014).

In some circumstances, it is certainly restrictive to consider that agents are located in the same route. For instance, they can be located in a circular route or other types of graphs, and that $a_S := A(S)$ for each $S \subseteq N$ and for a general function A. So, A is a general map to assign the variable part of the *transportation cost* of a group S that depends on the distance of a member of S to the supplier. In addition, we consider as parameters the fixed part of *ordering cost*, $a > 0$, the deterministic *demand* of the product of i, $d_i > 0$, and the *capacity* of the warehouse of i, $K_i > 0$.

Thus, this model is denoted by $(N, a, A, \{d_i, K_i\}_{i\in N})$. The optimal policy of a group $S \subseteq N$ is easy to obtain. Since holding costs are irrelevant, when the agents in S place joint orders, the length of a cycle is $\min_{j\in S}\{K_j/d_j\}$, and then the minimum joint cost per time unit is

$$\frac{\text{cost of a cycle}}{\text{length of a cycle}} = \frac{a + A(S)}{\min_{j\in S}\{K_j/d_j\}} = \big(a + A(S)\big)\max_{j\in S}\{\frac{d_j}{K_j}\}.$$

Thus, we can associate with every basic EOQ system without holding costs and with general transportation costs $(N, a, A, \{d_i, K_i\}_{i\in N})$ the cost game (N, c), which is given by, for all $S \subseteq N$,

$$c(S) := \big(a + A(S)\big)\max_{j\in S}\{\frac{d_j}{K_j}\}. \tag{12}$$

Notice that for each $S \subseteq N$, $c(S)$ can be written as the sum of two terms. Therefore, $c(S) := c_a(S) + c_b(S)$, with $c_a(S) = a\max_{j\in S}\{\frac{d_j}{K_j}\}$ and $c_b(S) = A(S)\max_{j\in S}\{\frac{d_j}{K_j}\}$ for each $S \subseteq N$.

For every system as the proposed, Saavedra-Nieves et al. (2018) introduce the following allocation.

This rule proposes, for every $i \in N$,

$$R_i(N,c) := \frac{1}{|\Pi(N)|} \sum_{\sigma \in \Pi(N)} m_i^{\sigma}(c_a) + \frac{1}{|\Pi_n(N)|} \sum_{\sigma \in \Pi_n(N)} m_i^{\sigma}(c_b), \tag{13}$$

where $\Pi_n(N) := \{\sigma \in \Pi(N) \text{ satisfying, for all } i \in \{1,\dots,n-1\}, \text{ that } \frac{d_{\sigma^{-1}(i)}}{K_{\sigma^{-1}(i)}} \geq \frac{d_{\sigma^{-1}(i+1)}}{K_{\sigma^{-1}(i+1)}}\}$ denotes the set of permutations that reverse the order of the ratios $\{d_i/K_i\}_{i\in N}$. Saavedra-Nieves et al. (2018) prove that if the game (N, c_b) is subadditive and the function A is submodular[2], the proposed rule is in the core of (N, c).

The inventory problems with no-holding costs and shortages

An *EOQ system without holding costs and shortages*, introduced in the reference Fiestras-Janeiro et al. (2015), is a multi-agent situation, where each agent faces a continuous review inventory problem with no-holding costs, with shortages and with a limited capacity warehouse. The parameters associated with each $i \in N$ are the fixed part of the *ordering cost*, $a > 0$; the *shortage cost* per item and per time unit, $b_i > 0$; the *demand* of the product of i, $d_i > 0$; and the *capacity* of the warehouse of i, $K_i > 0$.

This model is denoted by $(N, a, \{b_i, d_i, K_i\}_{i\in N})$. As shortages are allowed, the optimal policy changes. When agent i reaches the maximum shortage level, agent i places a new order of size $K_i + \beta_i$, where β_i is the level of shortages for i. So, the length of the cycle is $\frac{K_i+\beta_i}{d_i}$, and the length of the period with shortages is $\frac{\max\{\beta_i,0\}}{d_i}$. When a group $S \subseteq N$ cooperates, we assume that its members place joint orders, and thus:

$$\frac{1}{x_i} = \frac{K_i + \beta_i}{d_i} = \frac{K_j + \beta_j}{d_j} = \frac{1}{x_j},$$

where x_i and x_j denote the number of orders per time unit for each agent $i, j \in N$. Hence, the average cost per cycle for a coalition S is $a + \sum_{i\in S} b_i \frac{\max\{\beta_i,0\}}{2} \frac{\max\{\beta_i,0\}}{d_i}$, and the average cost by time unit is

$$\frac{a + \sum_{i\in S} b_i \frac{\max^2\{\beta_i,0\}}{2d_i}}{\frac{K_j+\beta_j}{d_j}} = \frac{ad_j}{K_j+\beta_j} + \frac{d_j}{K_j+\beta_j} \sum_{i\in S} b_i \frac{\max^2\{\beta_i,0\}}{2d_i}.$$

Denoting by x the common number of orders per time unit under cooperation, we have $\beta_i = -K_i + \frac{d_i}{x}$ for all $i \in S$, and the average cost per time unit can be written as:

$$C^S(x) := ax + x \sum_{i\in S} \frac{b_i}{2d_i} \max^2\{-K_i + \frac{d_i}{x}, 0\} = ax + \frac{1}{x} \sum_{i\in S} \frac{b_i}{2d_i} \max^2\{-K_i x + d_i, 0\}.$$

The calculus of the optimal policy in this problem consists of calculating the minima of $C^S(x)$. Fiestras-Janeiro et al. (2015) propose an iterative algorithm to obtain this value. So, once it is applied, $C^S(x)$ reaches its minimum value as follows:

$$C^S(x_S^*) := \sum_{i\in i(S)} b_i d_i \frac{1}{x_S^*} - \sum_{i\in i(S)} b_i K_i, \text{ in } x_S^* := \sqrt{\frac{\sum_{i\in i(S)} b_i d_i}{2a + \sum_{i\in i(S)} b_i \frac{K_i^2}{d_i}}} \text{ with } i(S) := \{i \in S : x_S^* < d_i/K_i\}. \tag{14}$$

Thus, we can associate with every system as described the cost game (N, c) which is given, for all $S \subseteq N$, by $c(S) := C^S(x_S^*)$. When agents in N cooperate, the authors propose a procedure to share the collaborating costs which ensures that the obtained allocation is in the core of (N, c). The G-rule proposes

$$G_i(c) := \begin{cases} b_i d_i \frac{1}{x_N^*} - b_i K_i, & \text{if } i \in i(N), \\ 0, & \text{otherwise.} \end{cases} \tag{15}$$

The stable character of the allocations can be justified by the fact of $\sum_{i\in S} G_i(c) \leq c(S)$ for all $S \subset N$ and $\sum_{i\in N} G_i(c) = c(N)$ (for details, see Fiestras-Janeiro et al. (2015)).

The previous model considers that the initial product and shortages have the same price by a unit of commodity. In some situations, the latter cost may be significantly smaller than the initial cost, as when shortages are produced by the same agent. Saavedra-Nieves et al. (2018a) justify and analyse this new inventory situation.

[2]A map $A : 2^N \to \mathbb{R}$ is submodular when the marginal contribution of an agent to a coalition does not increase when the coalition enlarges, that is, if, for all $S, T \subset N$ with $S \subset T$ and for all $i \in N \setminus T$, it holds that $A(S \cup \{i\}) - A(S) \geq A(T \cup \{i\}) - A(T)$.

An *EOQ problem without holding costs and with two acquisition costs* is a multi-agent inventory model wherein each agent faces a continuous-review inventory problem involving the same commodity. In this case, the considered parameters are the fixed part of the *ordering cost*, $a > 0$; the *shortage cost* per unit and per time unit, $b_i > 0$; the *demand* of the product of i, $d_i > 0$; and the *capacity* of the warehouse of i, $K_i > 0$. Additionally:

- $c_1 > 0$, the *acquisition cost* of one unit of the *purchased* commodity,
- $c_2 > 0$, the *acquisition cost* of one unit of the *produced* commodity, with $c_2 < c_1$.

We denote this class of situations by $(N, a, c_1, c_2, \{b_i, d_i, K_i\}_{i\in N})$. In an analogous way to the model studied in Fiestras-Janeiro et al. (2015), the inventory optimal policy varies when shortages are allowed. If agent i reaches the maximum shortage level, agent i places an order of size $K_i + \beta_i$, with β_i the level of shortages for i. The length of the cycle is given by $\frac{K_i+\beta_i}{d_i}$ and the length of the period with shortages is $\frac{\max\{\beta_i,0\}}{d_i}$. When a coalition S is formed, it verifies the following:

$$\frac{1}{x_i} = \frac{K_i + \beta_i}{d_i} = \frac{K_j + \beta_j}{d_j} = \frac{1}{x_j},$$

where x_i and x_j denote the number of orders per time unit for each agent $i, j \in S$. Hence, the average cost per cycle for a coalition S is $[a + c_1 \sum_{i\in S} (K_i + \min\{\beta_i, 0\})] + \sum_{i\in S} \left[b_i \frac{\max^2\{\beta_i,0\}}{2d_i} + c_2 \max\{\beta_i, 0\}\right]$ and the average cost by time unit is

$$\frac{d_j}{K_j + \beta_j}[a + c_1 \sum_{i\in S} (K_i + \min\{\beta_i, 0\})] + \frac{d_j}{K_j + \beta_j} \sum_{i\in S}[b_i \frac{\max^2\{\beta_i, 0\}}{2d_i} + c_2 \max\{\beta_i, 0\}].$$

Taking $\beta_i := -K_i + \frac{d_i}{x}$ for all $i \in S$, we have for each $x > 0$:

$$C^S(x) := x \left[a + c_1 \sum_{i\in S} \left(K_i + \min\{-K_i + \frac{d_i}{x}, 0\}\right)\right] + \\ + \frac{1}{x} \sum_{i\in S} \frac{b_i}{2d_i} \max^2\{-K_i x + d_i, 0\} + c_2 \sum_{i\in S} \max\{-K_i x + d_i, 0\}.$$

Authors propose another iterative procedure to obtain the optimal inventory policy with the calculus of the minima of $C^S(x)$. So, the minimum value of $C^S(x)$ is

$$C^S(x_S^*) := \sum_{i\in i(S)} b_i d_i \frac{1}{x_S^*} + c_1 \sum_{i\in S\setminus i(S)} d_i + \sum_{i\in i(S)} (c_2 d_i - b_i K_i), \tag{16}$$

where $x_S^* := \sqrt{\frac{\sum_{i\in i(S)} b_i d_i}{2a + 2(c_1 - c_2) \sum_{i\in i(S)} K_i + \sum_{i\in i(S)} b_i \frac{K_i^2}{d_i}}}$ with $i(S) := \{i \in S : x_S^* < d_i/K_i\}$. A cost game (N, c) can be associated with every system as proposed. It is given by $c(S) := C^S(x_S^*)$ for all $S \subseteq N$. When agents in N cooperate in $(N, a, c_1, c_2, \{b_i, d_i, K_i\}_{i\in N})$, the *GR-rule* allocates, for $i \in N$:

$$GR_i(c) := \begin{cases} b_i d_i \frac{1}{x_N^*} + c_2 d_i - b_i K_i, & \text{si } i \in i(N), \\ c_1 d_i, & \text{si } i \in N \setminus i(N). \end{cases} \tag{17}$$

In order to summarize the assumptions on the described models in this section, Table 1 relates them with the associated cost game and the allocation rule proposed for each case.

Inventorymodel in practice

This section presents an overview of the structure of the package. **Inventorymodel** is a tool for calculating the optimal policy in several inventory situations and for allocating the collaboration costs among the agents involved in the process. This software helps the user to determine faster the optimal policy when the number of agents increases. Remark that these functions only automatize the required operations for the calculus of the optimal costs under theses scenarios although the computational complexity still maintains.[3]

[3]Recall that for a given value of n, the length of a characteristic cost function corresponds to the quantity of coalitions to be evaluated, this is, 2^n subsets of N.

Ordering costs	Variable ordering costs	Holding costs	Shortages	(N, c)	Allocation rule
✓	-	✓	-	(2)	*SOC rule* (5)
✓	-	✓	✓	(4)	*SOC rule* (5)
✓	✓$(a_i,\ i \in N)$	✓	-	(6)	*Line-rule* (7)
✓	-	-	-	(8)	*Shapley value* (9)
✓	✓$(a_i,\ i \in N)$	-	-	(10)	*Two-lines rule* (11)
✓	✓$(A(S),\ S \subseteq N)$	-	-	(12)	*R-rule* (13)
✓	-	-	✓	(14)	*G-rule* (15)
✓	-	-	✓(two costs)	(16)	*GR-rule* (17)

Table 1: Summary of assumptions, cost games, and allocations rules.

Inventorymodel package includes several functions that enable users to determine the optimal inventory policy for the inventory models decribed in previous section. Moreover, auxiliary functions are implemented to obtain some intermediate values in the calculation. This package has as dependencies **e1071** and **GameTheoryAllocation** packages of which takes some functions in its calculations. The last one, in a primitive version, includes some generic functions related to game-theoretic concepts. The functions incorporated in the package are briefly described in Table 2.

Function	Description
`coalitions`	Function to obtain all possible coalitions with n players.
`EOQ`	Function to calculate the optimal policy and costs in an EOQ model (2).
`EOQcoo`	Function to calculate the optimal policy and associated cost game in an EOQ model when n agents cooperate (2).
`EPQ`	Function to calculate the optimal policy and costs in an EPQ model (4).
`EPQcoo`	Function to calculate the optimal policy and associated cost game in an EPQ model when n agents cooperate (4).
`inventorygames`	Function to determine the associated cost game in an EOQ (2) or an EPQ (4) model.
`inventorymodelfunction`	Main function to obtain the optimal policy by selecting any model and its parameters.
`linerule`	Function to obtain the line-rule allocation (7) for an inventory transportation system (6) when a coalition N is formed .
`linerulecoalitional`	Function to obtain the line-rule allocation (7) for an inventory transportation system (6) for each possible coalition.
`marginal_contribution_mean`	Intermediate auxiliary function to calculate the mean of the marginal contributions for a set of permutations.
`mct`	Function to obtain the costs in a basic EOQ system without holding costs and with transportation cost (10).
`mfoc`	Function to obtain the associated costs in a fixed order cost model (8).
`mwhc`	Function to obtain the associated costs in a model without holding costs (14).
`mwhc2c`	Function to obtain the associated costs in a model without holding costs and with two different costs of a product (16).
`mwhcct`	Function to obtain the associated costs in a basic EOQ system without holding costs and with a general transportation cost (12).
`shapley_mfoc`	Function to calculate the Shapley value (9) for the associated game in a fixed order cost model (8).
`SOC`	Function to obtain the allocation proposed by the SOC rule (5) under an EOQ/EPQ model, see (2) and (4).
`STI`	Function to obtain the optimal orders and associated cost in inventory transportation systems (6).
`STIcoo`	Function to obtain the optimal orders and associated cost when n agents are cooperating in the inventory transportation system (6).
`twolines`	Function to calculate the allocation proposed by the two-lines rule (11) in a basic EOQ system without holding costs and with transportation cost (10).

Table 2: Summary of functions in the **Inventorymodel** package.

Users can obtain the optimal inventory policy in each case by introducing the associated parameters to each model in the corresponding function. Table 3 describes the different options to define the inventory models included in **Inventorymodel**. However, not all of the mentioned options are necessary since only some of them are specific for each inventory situation. Table 4 summarizes the required arguments in the specific function associated with each inventory model. Examples of usage for the implemented functions are described in the next section.

Argument	Description
`model`	A character string that indicates the selected model and required only in the `inventorymodelfunction`. The possible values are `"EOQ"` (EOQ), `"EPQ"` (EPQ), `"STI"` (the inventory transportation systems), `"FOC"` (the fixed order cost model), `"MCT"` (the basic EOQ system without holding costs and with transportation cost), `"MWHC"` (the model without holding costs and shortages), `"MWHC2"` (the model without holding costs and two acquisition costs), and `"MWHCCT"` (the EOQ system without holding costs and with general transportation costs).
`n`	An integer specifying the number of agents involved in the inventory model. It corresponds to the cardinality of the set N.
`a`	A real number that indicates the fixed cost per new order.
`av`	A vector of dimension `n` that contains, in each component, the transportation costs associated with each agent $i \in N$.
`d`	A vector of dimension `n` that contains, in each component, the deterministic demands per time unit associated with each agent $i \in N$.
`h`	A vector of dimension `n` that indicates, in each component, the holding costs per product unit associated with each agent $i \in N$.
`m`	A vector of dimension `n` that indicates, in each component, the number of orders m_i associated with each agent $i \in N$. These values are necessary only when information about demands is unknown.
`r`	A vector of dimension `n` that contains, in each component, the replacement rate of a product associated with each agent $i \in N$ in a EPQ model. In general, it verifies $r_i > d_i$.
`K`	A vector of dimension `n` that details, in each component, the warehouse capacity of the product associated with each agent $i \in N$.
`b`	A vector of dimension `n` that details, in each component, the cost by shortage unit of the product associated with each agent $i \in N$.
`c1`	A real value with the acquisition cost of one unit of the purchased commodity.
`c2`	A real value with the acquisition cost of one unit of the produced commodity.
`cooperation`	A variable that indicates if cooperation is considered. If it exists, `cooperation=1`, else `cooperation=0`.
`allocation`	A variable that indicates if the defined allocation in each case is required. If it exists, `allocation=1`, else `allocation=0`.

Table 3: Summary of arguments for functions in the **Inventorymodel** package.

Additionally, the function `coalitions` takes, as a unique argument, the integer variable `n` (the number of involved agents) to calculate all possible subsets of the set of N. The auxiliary function `marginal_contribution_mean` allows the calculation of the average of the marginal contribution vectors for some indicated permutations. Therefore, it only needs as arguments a matrix with the permutations to study (`permute`) and a vector that contains the characteristic function with associated cost values for each possible coalition in a set of agents N (`costs`).

Examples of application

Several examples of application of the **Inventorymodel** package are used to illustrate its performance in the different contexts considered. After installing the package from CRAN, as an initial step a user should load the package in the R Console:

```
> library("Inventorymodel")
```

Below, in order to illustrate the performance of those functions which determine the optimal order and the associated cost game, we consider examples extracted from the references. In this situations,

Function	model	n	a	av	d	h	m	r	K	b	c1	c2	cooperation	allocation
EOQ	-	✓	✓	-	✓	✓	✓	-	-	-	-	-	-	-
EOQcoo	-	✓	✓	-	✓	✓	✓	-	-	-	-	-	-	-
EPQ	-	✓	✓	-	✓	✓	✓	✓	-	✓	-	-	-	-
EPQcoo	-	✓	✓	-	✓	✓	✓	✓	-	✓	-	-	-	-
inventorygames	✓	-	✓	-	✓	✓	✓	✓	-	✓	-	-	-	-
inventorymodelfunction	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
linerule	-	✓	✓	✓	✓	✓	✓	-	-	-	-	-	-	-
linerulecoalitional	-	✓	✓	✓	✓	✓	✓	-	-	-	-	-	-	-
mct	-	✓	✓	✓	✓	-	-	-	✓	-	-	-	✓	-
mfoc	-	✓	✓	-	✓	-	-	-	✓	-	-	-	✓	-
mwhc	-	✓	✓	-	✓	-	-	-	✓	✓	-	-	✓	✓
mwhc2c	-	✓	✓	-	✓	-	-	-	✓	✓	✓	✓	✓	✓
mwhcct	-	✓	✓	✓	✓	-	-	-	✓	-	-	-	✓	✓
shapley_mfoc	-	✓	✓	-	✓	-	-	-	✓	-	-	-	-	-
SOC	✓	✓	✓	-	✓	✓	✓	✓	-	-	-	-	-	-
STI	-	✓	✓	✓	✓	✓	✓	-	-	-	-	-	-	-
STIcoo	-	✓	✓	✓	✓	✓	✓	-	-	-	-	-	-	-
twolines	-	✓	✓	✓	✓	-	-	-	✓	-	-	-	-	-

Table 4: Arguments for each function in the **Inventorymodel** package.

the number of involved agents is always lesser than $n = 10$. For cases where n is greater, such as we have mentioned through this paper, the characteristic function requires a high memory size. Hence, the implemented functions only return the individual costs, $c(\{i\})$, and values associated to the formation of coalition N, $c(N)$, with an analogous structure of the results.

The EOQ and EPQ models under cooperation in R

This example, taken from Meca et al. (2004), describes the EOQ problem that surges when three airlines place joint orders. Let $N = \{1,2,3\}$ be the set of agents and $a = 600$ the fixed cost of a new order with values for the individual demands, and the holding costs (by unit of product) are depicted in Table 5.

i	1	2	3
d_i	500	300	400
h_i	9.6	11	10

Table 5: Demands and holding costs for agents in N in the EOQ problem.

Using `EOQcoo`, we obtain the optimal policy for each possible subgroup $S \subseteq N$ in terms of the number of orders and its associated costs (as a data frame in a list) following (2).

```
> EOQcoo(n = 3, a = 600, d = c(500, 300, 400), h = c(9.6, 11, 10))
EOQ model
Cooperative case
Optimal order
    Coalition        1        2        3 Coalitional costs
1           0   0.0000   0.0000   0.0000             0.000
2     '{ 1 }' 250.0000   0.0000   0.0000          2400.000
3     '{ 2 }'   0.0000 180.9068   0.0000          1989.975
4     '{ 3 }'   0.0000   0.0000 219.0890          2190.890
5   '{ 1,2 }' 192.4501 115.4701   0.0000          3117.691
6   '{ 1,3 }' 184.6372   0.0000 147.7098          3249.615
7   '{ 2,3 }'   0.0000 121.6327 162.1770          2959.730
8 '{ 1,2,3 }' 157.4592  94.4755 125.9673          3810.512
```

To evaluate the implementation in R of the EPQ model, we consider in the previous example a replacement rate $r_i = 600$ units of product for all $i \in N$. Moreover, the cost by shortage unit is 100, 150 and 200 for each agent, respectively. Below, the optimal policy in terms of the number of orders and shortages is depicted (4). It is obtained by using `EPQcoo` in R.

```
> EPQcoo(n = 3, a = 600, d = c(500, 300, 400), h = c(9.6, 11, 10),
+     r = rep(600, 3), b = c(100, 150, 200))
EPQ model
```

```
Cooperative case
`Optimal order`
    Coalition        1        2        3 Coalitional costs
1           0   0.0000   0.0000   0.0000            0.0000
2     '{ 1 }' 641.0928   0.0000   0.0000          935.9019
3     '{ 2 }'   0.0000 265.0557   0.0000         1358.2049
4     '{ 3 }'   0.0000   0.0000 388.8444         1234.4268
5   '{ 1,2 }' 363.7611 218.2567   0.0000         1649.4341
6   '{ 1,3 }' 387.3208   0.0000 309.8566         1549.1036
7   '{ 2,3 }'   0.0000 196.1473 261.5297         1835.3556
8 '{ 1,2,3 }' 291.2332 174.7399 232.9866         2060.2045

`Optimal shortages`
       1        2        3
0.000000 0.000000 0.000000
9.359019 0.000000 0.000000
0.000000 9.054699 0.000000
0.000000 0.000000 6.172134
5.310381 7.455973 0.000000
5.654318 0.000000 4.918359
0.000000 6.700683 4.151265
```

The output is a list which contains as a first element a data frame with optimal orders and the associated costs, and a data frame where the optimal shortages by agent are depicted for the different coalitions.

The calculation of the SOC rule (5) in the abovementioned example for the EOQ problem completes this evaluation. To conclude, we use the `SOC` function to obtain this allocation (as a numeric vector).

```
> SOC(n = 3, a = 600, d = c(500, 300, 400), h = c(9.6, 11, 10), model = "EOQ")
EOQ model
`Share the ordering costs rule (individually)`
        1        2        3
    0.000    0.000    0.000
 2400.000    0.000    0.000
    0.000 1989.975    0.000
    0.000    0.000 2190.890
 1847.521 1270.171    0.000
 1772.517    0.000 1477.098
    0.000 1337.960 1621.770
 1511.608 1039.230 1259.673
```

In an obvious way, this function returns the allocation proposed by the SOC rule correspond to each possible group of agents.

The inventory transportation systems in R

To evaluate the performance of the inventory transportation systems in R, we consider the example depicted in Fiestras-Janeiro et al. (2012). Let $N = \{1, 2, 3\}$ be a set of three involved agents, with $a = 200$ as the fixed cost of a new order. The remainder of the data are detailed in Table 6.

i	1	2	3
a_i	300	300	900
d_i	90	80	20
h_i	0.06	0.06	0.1

Table 6: Distances to the supplier, demands, and individual holding costs.

The function `STIcoo` allows the determination of the optimal policy for each possible $S \subseteq N$ in this inventory situation. Its result is shown below in terms of the number of orders and its associated costs (6).

```
> STIcoo(n = 3, a = 200, av = c(300, 300, 900), d = c(90, 80, 20),
+    h = c(0.06, 0.06, 0.1))
```

```
`Optimal order`
          1        2        3   Coalition Order cost
1    0.0000    0.000   0.0000           0    0.00000
2 1224.7449    0.000   0.0000     '{ 1 }'   73.48469
3    0.0000 1154.701   0.0000     '{ 2 }'   69.28203
4    0.0000    0.000 663.3250     '{ 3 }'   66.33250
5  891.1328  792.118   0.0000   '{ 1,2 }'  100.99505
6 1551.8080    0.000 344.8462   '{ 1,3 }'  127.59310
7    0.0000 1438.954 359.7385   '{ 2,3 }'  122.31108
8 1208.5759 1074.290 268.5724 '{ 1,2,3 }'  163.82918
```

Finally, we utilize the function `linerule` to obtain the allocation proposed by the line-rule when N is formed (7).

```
> linerule(n = 3, a = 200, av = c(300, 300, 900), d = c(90, 80, 20),
+    h = c(0.06, 0.06, 0.1))
[1] 51.38935 46.10733 66.33250
```

The inventory problems with no-holding costs and without shortages in R

First, we evaluate the basic EOQ system without holding costs in R by taking the example considered in Fiestras-Janeiro et al. (2014). There are five farms with 40, 140, 120, 130 and 120 cows, respectively. The fixed cost by order is $a = 200$ euros. The information about demands of the product and capacity of warehouses are depicted in Table 7.

i	1	2	3	4	5
d_i	0.4	1.4	1.2	1.3	1.2
K_i	4	10	8	8	6

Table 7: Demands and capacities of the warehouses.

By using the `mfoc` function, we obtain the individual inventory costs (as a numeric vector).

```
> mfoc(n = 5, a = 200, d = c(0.4, 1.4, 1.2, 1.3, 1.2), K = c(4, 10, 8, 8, 6),
+    cooperation = 0)
MFOC model
[1] 20.0 28.0 30.0 32.5 40.0
```

When `cooperation=1`, the costs associated to each possible coalition (8) are depicted in a data frame.

Below, we obtain the Shapley value (as `numeric`) with the previous costs according to that described (9). For this, we use the `shapley_mfoc` function.

```
> shapley_mfoc(n = 5, a = 200, d = c(0.4, 1.4, 1.2, 1.3, 1.2),
+    K = c(4, 10, 8, 8, 6))
Shapley-Value
MFOC model
[1]  4.000000  6.000000  6.666667  7.916667 15.416667
```

In order to evaluate the basic EOQ system without holding costs and with transportation costs in R, we consider the example proposed in Fiestras-Janeiro et al. (2014) for three farms. In this case, if $N = \{1, 2, 3\}$, the cost by a new order is $a = 100$, and the rest of the data are detailed in Table 8.

i	d_i	K_i	a_i	d_i/K_i
1	2	9	300	0.222
2	2	8	500	0.250
3	5	7	200	0.714

Table 8: Demands, capacities of the warehouses, and costs by transportation.

The function `mct` allows the calculation, for each $S \subseteq N$, of the optimal costs of the collaboration (10). As `cooperation=1`, the output is a matrix; otherwise, it is a numeric vector.

```
> mct(n = 3, a = 400, av = c(300, 500, 200), d = c(2, 2, 5), K = c(9, 8, 7),
+    cooperation = 1)
MCT model
Cooperative case
  1 2 3   Coalition     Cost
1 0 0 0           0   0.0000
2 1 0 0     '{ 1 }' 155.5556
3 0 1 0     '{ 2 }' 225.0000
4 0 0 1     '{ 3 }' 428.5714
5 1 1 0   '{ 1,2 }' 225.0000
6 1 0 1   '{ 1,3 }' 500.0000
7 0 1 1   '{ 2,3 }' 642.8571
8 1 1 1 '{ 1,2,3 }' 642.8571
```

For this class of problems, when N agents cooperate, the allocation proposed by the two-lines rule (11) for the dealt example is obtained by the function `twolines`:

```
> twolines(n = 3, a = 400, av = c(300, 500, 200), d = c(2, 2, 5), K = c(9, 8, 7))
MCT model
Cooperative case
Two-lines rule
[1]   0.0000 219.6429 423.2143
```

Finally, the performance of the implementation in R of the EOQ systems without holding costs and with general transportation costs is considered. For this, we take one example for $n = 3$ agents extracted from Saavedra-Nieves et al. (2018). In that case, $a = 10$ and $d/K = (1, 0.95, 0.9)$, and the transportation cost function is detailed in Table 9.

S	$\{1\}$	$\{2\}$	$\{3\}$	$\{1,2\}$	$\{1,3\}$	$\{2,3\}$	N
$A(S)$	10	10	10	20	20	20	30

Table 9: Transportation costs function A for each $S \subseteq N$.

The function `mwhcct` allows the determination of the optimal policy for this model (12), and with the selected option, we obtain the allocation the R-rule (13) proposes.

```
> mwhcct(n = 3, a = 10, av = c(0, 10, 10, 10, 20, 20, 20, 30), d = c(1, 0.95,
+    0.9), K = c(1, 1, 1), cooperation = 1, allocation = 1)
MWHC with Transportation Costs model
Cooperative case
`Optimal solution`
  1 2 3   Coalition Cost
1 0 0 0           0  0.0
2 1 0 0     '{ 1 }' 20.0
3 0 1 0     '{ 2 }' 19.0
4 0 0 1     '{ 3 }' 18.0
5 1 1 0   '{ 1,2 }' 30.0
6 1 0 1   '{ 1,3 }' 30.0
7 0 1 1   '{ 2,3 }' 28.5
8 1 1 1 '{ 1,2,3 }' 40.0

`Allocation R rule`
    1     2  3
13.75 13.25 13
```

The results are in a list which contains a data frame with optimal costs and a vector with the allocation that the R rule proposes for this model.

The inventory problems with no-holding costs and shortages in R

The EOQ system without holding costs and shortages in R is evaluated using the example taken from Fiestras-Janeiro et al. (2015). In that case, $a = 180$, and the rest of the necessary data in the model are given in Table 10.

i	1	2	3	4
d_i	0.45	0.95	1.05	1.20
b_i	15	15	10	12
K_i	5	7.5	8	9

Table 10: Demands, shortage costs, and capacities of the warehouses.

Using the function `mwhc`, the optimal policy (14) in terms of the number of orders and shortages is obtained, and the allocation proposed for this inventory scenario is also calculated for the case in which N cooperate.

```
> mwhc(n = 4, a = 180, b = c(15, 15, 10, 12), d = c(0.45, 0.95, 1.05, 1.2),
+    K = c(5, 7.5, 8, 9), cooperation = 0, allocation = 0)
MWHC model
`Number of orders per time unit`
[1] 0.07520921 0.10684954 0.10406757 0.11094004

Costs
[1] 14.74965 20.86510 20.89599 21.79985

`Optimal shortages`
[1] 0.9833101 1.3910067 2.0895986 1.8166538
```

In this case, the output corresponds to a list with components for the orders, the costs, and the optimal shortages. When `cooperation=1`, the output is a list with the optimal policies for the different groups, indicating the orders and the costs, and the allocation proposed by G (15).

To conclude, we analyse the EOQ problem without holding costs and with two acquisition costs in R. For this, we consider an example from Saavedra-Nieves et al. (2018a) for $n = 3$ agents whose associated data is depicted in Table 11. The remainder of the data are given by $a = 1$, $c_1 = 3.5$, and $c_2 = 3$.

i	1	2	3
d_i	30	45	46
b_i	10	10	10
K_i	30	45	46

Table 11: Demands, shortage costs, and capacities of the warehouses.

The function `mwhc2c` allows the calculation of the optimal policy in terms of the number of orders and shortages when all agents cooperate (16). Moreover, we obtain the proposed allocation in this case.

```
> mwhc2c(n = 3, a = 1, b = c(10, 10, 10), d = c(30, 45, 46),
+     K = c(30, 45, 46), c1 = 3.5, c2 = 3, cooperation = 0, allocation = 0)
MWHC model
`Number of orders per time unit`
[1] 0.9505864 0.9515422 0.9515838

Costs
[1] 105.5947 157.9165 161.4046

`Optimal shortages`
[1] 1.559468 2.291648 2.340459
```

The output has the following structure: it is a list with the orders, the costs and the optimal shortages. When `cooperation=1`, the output corresponds to a list with the optimal policies for the different groups, with the associated orders and the costs in a data frame, and the allocation proposed by GR (17) for each coalition $S \subseteq N$ in a matrix.

All examples dealt with in this section can be evaluated with `inventorymodelfunction` by introducing, for each case, the corresponding character chain for the associated model and its parameters (see Table 3 for details).

Computing time

To illustrate the time required by the different functions in the R package **Inventorymodel**, we performed the following simulations with artificial data. We generated, according to a uniform distribution, 100 samples of the involved parameters in the functions included in the package for values of n from 3 to 10.

For each sample, we evaluate the available functions. All the computations included in this paper have been performed on a personal computer with Intel(R) Core(TM) i5-4210U and 8 GB of memory using a single 2.40GHz CPU processor. The results of the evaluations for all samples are depicted in Tables 12 and 13, where the values are reported in seconds.[4]

n	3	4	5	6	7	8	9	10
`EOQcoo`	0.00	0.00	0.00	0.00	0.00	0.00	0.02	0.00
`EPQcoo`	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
`STIcoo`	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.04
`mfoc`	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
`mct`	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.01
`mwhc`	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.01
`mwhc2c`	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
`mwhcct`	0.00	0.00	0.00	0.00	0.00	0.02	0.00	0.01

Table 12: System time (in seconds) for 100 computations using **Inventorymodel** and for several values of n.

n	3	4	5	6	7	8	9	10
`EOQcoo`	0.00	0.00	0.34	0.72	1.00	2.20	4.35	9.35
`EPQcoo`	0.00	0.00	0.26	0.24	0.63	1.35	2.72	5.98
`STIcoo`	0.00	0.00	0.37	0.65	1.14	2.13	4.50	9.43
`mfoc`	0.00	0.00	0.32	0.45	0.75	1.65	3.48	7.16
`mct`	0.00	0.00	1.00	1.96	4.03	8.34	17.97	38.22
`mwhc`	0.00	0.00	0.72	1.27	2.17	4.45	9.32	18.93
`mwhc2c`	0.00	0.00	0.92	1.64	3.26	7.31	14.38	31.86
`mwhcct`	0.00	0.00	0.84	1.76	3.52	7.27	15.47	32.79

Table 13: User time (in seconds) for 100 computations using **Inventorymodel** and for several values of n.

We can observe that the runtime is mainly affected by the number of agents. As we expected, as the procedure is based on the subsets of N, the dimension of the characteristic function determines the runtime required for its calculation. In view of Tables 12 and 13, it is easy to check that when n increases, the computing time increases too. However, in most of cases, it does not exceed ten seconds of running time for all computation. Only in seven situations the computing time is bigger (without exceeding a minute).

Conclusions

This paper discusses the implementation in R of several procedures to calculate the optimal policy in different inventory situations from using game theory results. Mechanisms to allocate the collaborating

[4]These values were obtained by using the basic function `proc.time()` in R. By definition, the *user time* is the CPU time charged for the execution of user instructions of the calling process. The *system time* is the CPU time charged for execution by the system on behalf of the calling process.

costs among the agents have been incorporated as well. Thus, the **Inventorymodel** package provides R users a set of functions to determine the optimal costs in different situations defined in the literature: with holding costs, with costs by the transport of a new order, with shortages, and with different prices of a unit of purchased and produced products. When the number of agents increases, these calculations become computationally more complicated since that the number of possible groups of agents considerably enlarges. Future research and development plans for forthcoming versions of the package include the addition of new inventory models that can be dealt with in the framework presented above and the inclusion of alternative procedures in order to reduce the computational complexity present in this class of problems.

Acknowledgements

The author acknowledges the financial support of Ministerio de Economía y Competitividad through projects MTM2014-53395-C3-2-P and MTM2014-53395-C3-3-P and the financial support of Xunta de Galicia through the ERDF (Grupos de Referencia Competitiva) ED431C 2016-040. The author acknowledges M. Gloria Fiestras-Janeiro for her constructive comments on this paper.

Bibliography

M. Dror and B. C. Hartman. Survey of cooperative inventory games and extensions. *Journal of the Operational Research Society*, 62(4):565–580, 2011. URL https://doi.org/10.1057/jors.2010.65. [p201]

M. Fiestras-Janeiro, I. García-Jurado, A. Meca, and M. Mosquera. Cost allocation in inventory transportation systems. *Top*, 20(2):397–410, 2012. URL https://doi.org/10.1007/s11750-011-0207-7. [p201, 203, 204, 205, 211]

M. G. Fiestras-Janeiro, I. García-Jurado, A. Meca, and M. A. Mosquera. Cooperative game theory and inventory management. *European Journal of Operational Research*, 210(3):459–466, 2011. URL https://doi.org/10.1016/j.ejor.2010.06.025. [p201]

M. G. Fiestras-Janeiro, I. García-Jurado, A. Meca, and M. A. Mosquera. Centralized inventory in a farming community. *Journal of Business Economics*, 84(7):983–997, 2014. URL https://doi.org/10.1007/s11573-014-0710-z. [p201, 204, 205, 212]

M. G. Fiestras-Janeiro, I. García-Jurado, A. Meca, and M. A. Mosquera. Cooperation on capacitated inventory situations with fixed holding costs. *European Journal of Operational Research*, 241(3):719–726, 2015. URL https://doi.org/10.1016/j.ejor.2014.09.016. [p201, 206, 207, 213]

D. B. Gillies. *Some Theorems on n-Person Games*. PhD thesis, Princeton University, 1953. [p201]

J. González-Díaz, I. García-Jurado, and M. G. Fiestras-Janeiro. *An Introductory Course on Mathematical Game Theory*, volume 115. American Mathematical Society Providence, 2010. [p201]

S. C. Littlechild and G. Owen. A simple expression for the Shapley value in a special case. *Management Science*, 20(3):370–372, 1973. URL https://doi.org/10.1287/mnsc.20.3.370. [p204]

A. Meca, I. García-Jurado, and P. Borm. Cooperation and competition in inventory games. *Mathematical Methods of Operations Research*, 57(3):481–493, 2003. URL https://doi.org/10.1007/s001860200253. [p201, 202, 203]

A. Meca, J. Timmer, I. García, P. Borm, and others. Inventory games. *European Journal of Operational Research*, 156(1):127–139, 2004. URL https://doi.org/10.1016/s0377-2217(02)00913-x. [p210]

M. Nagarajan and G. Sošić. Game-theoretic analysis of cooperation among supply chain agents: Review and extensions. *European Journal of Operational Research*, 187(3):719–745, 2008. URL https://doi.org/10.1016/j.ejor.2006.05.045. [p201]

A. Saavedra-Nieves, I. García-Jurado, and M. G. Fiestras-Janeiro. Placing joint orders when holding costs are negligible and shortages are not allowed. In D. Mueller and R. Trost, editors, *Game Theory in Management Accounting: Implementing Incentives and Fairness*, pages 349–360. Springer-Verlag, 2018. URL https://doi.org/10.1007/978-3-319-61603-2_16. [p201, 205, 206, 213]

A. Saavedra-Nieves, I. García-Jurado, and M. G. Fiestras-Janeiro. On coalition formation in a non-convex multi-agent inventory problem. *Annals of Operations Research*, 261(1):255–273, 2018a. URL https://doi.org/10.1007/s10479-017-2616-y. [p201, 206, 214]

L. S. Shapley. A value for n-person games. *Contributions to the Theory of Games*, 2(28):307–317, 1953. [p201]

Alejandro Saavedra-Nieves
Departamento de Estatística e Investigación Operativa, Universidade de Vigo
36200, Vigo
Spain
asaavedra@uvigo.es

R Package imputeTestbench to Compare Imputation Methods for Univariate Time Series

by Marcus W Beck, Neeraj Bokde, Gualberto Asencio-Cortés, and Kishore Kulat

Abstract Missing observations are common in time series data and several methods are available to impute these values prior to analysis. Variation in statistical characteristics of univariate time series can have a profound effect on characteristics of missing observations and, therefore, the accuracy of different imputation methods. The **imputeTestbench** package can be used to compare the prediction accuracy of different methods as related to the amount and type of missing data for a user-supplied dataset. Missing data are simulated by removing observations completely at random or in blocks of different sizes depending on characteristics of the data. Several imputation algorithms are included with the package that vary from simple replacement with means to more complex interpolation methods. The testbench is not limited to the default functions and users can add or remove methods as needed. Plotting functions also allow comparative visualization of the behavior and effectiveness of different algorithms. We present example applications that demonstrate how the package can be used to understand differences in prediction accuracy between methods as affected by characteristics of a dataset and the nature of missing data.

Introduction

Univariate time series data provide valuable information that forms the basis of analysis across several disciplines. The most notable characteristic of time series data is time-dependent correlations between observations such that the likelihood of observing a single value depends on the values of past or future observations (Shumway and Stoffer, 2011; Box et al., 2015). This precludes the use of most conventional statistical methods that require independence of observations as a fundamental assumption prior to analysis. However, unique methods have been developed that account for and leverage the statistical characteristics of time series data to provide insight beyond those provided by more conventional techniques. For example, additive or multiplicative models can be used to separate a univariate time series data into unique components with different structures (e.g., Gould et al., 2008), whereas multiple methods for signal processing seek to isolate dominant components that are independent of random variation in the observed data (e.g., Mendel, 2000).

Many analysis methods for time series require complete observations at each time step across the period of interest. Complete datasets rarely exist such that missing observations are often defining characteristics of time series data (e.g., Gomez and Maravall, 1994; Honaker and King, 2010). Missing observations can occur for several reasons, although the type and amount of missing values depend on characteristics of the data (Schafer, 1997; Schafer and Graham, 2002). For example, a time series with no serial dependence between observations is more likely to have missing observations that are completely random (Rubin, 1976). A more common scenario for time series data is missing observations in 'chunks' or data missing in sequence for a period of time (Schafer, 1997; Donders et al., 2006). The occurrence of missing chunks relates directly to the correlation structure of time series data, as in cases when experiments are not continuously maintained, remote data fail to transmit from the source to the user, or funding mechanisms for monitoring programs are discontinuous. As such, the accuracy of imputation methods can vary considerably depending on characteristics of the dataset (Yozgatligil et al., 2013). Similar to the application of analysis methods that use time-dependent information, imputation methods must also faithfully reconstruct missing observations in this context.

Identifying an appropriate imputation method is often the first step towards more formal time series analysis. Different imputation methods will have differing precision in reproducing missing values, where precision will depend on how much data are missing and how the data are missing (i.e., individual observations missing at random or data missing in continuous chunks). The characteristics of the dataset will also influence imputation precision between methods. An expectation is that imputation methods that leverage characteristics of the dataset to predict missing values will perform better than more naïve methods, if indeed there is sufficient temporal structure. Accordingly, choosing an appropriate imputation method can be facilitated by using a standardized method of comparison. A simple approach for method comparison is to evaluate prediction accuracy from imputed values after removing observations from a test dataset, where the test dataset should have characteristics similar to the one requiring imputation. For example, Zhu et al. (2011) proposed a kernel-based iterative estimation method for missing data and compared the approach to other conventional frequency

	10%		80%	
Method names	*T*	*V*	*T*	*V*
Mixing	8	0.085	20	1.53
Poly	10	0.103	25	2.11
RBF	11	0.107	29	2.86
Normal	14	0.121	30	3.01
FE	13	0.117	29	2.59

Table 1: An approach for comparing imputation methods. Methods in the left column are compared by varying the amount of missing data (10% and 80%) in a complete dataset and number of iterations (*T*) for which the missing data are removed at random. *V* is the mean *RMSE* of the imputed values. Reproduced from Zhu et al. (2011).

estimators. The methods were compared by simulating different amounts of missing data, predicting the missing values with each method, and then comparing the predictions to the actual data that were removed. Table 1 reproduces the results, where the rows show root-mean squared error (*RMSE*) between observed and predicted data for each imputation method after removing and predicting 10% and 80% of the complete dataset. Additional studies have used a similar workflow to compare the performance of imputation methods (Jörnsten et al., 2007; Li et al., 2015; Nguyen et al., 2013; Ran et al., 2015; Tak et al., 2016).

There are several challenges for adopting a standardized approach to compare imputation methods. An obvious concern is the amount of missing data, such that a complete gradient from very few to many observations should be evaluated with each method (Zhu et al., 2011). For example, one method may be superior for very few missing observations but perform poorly relative to other methods for many missing observations. Additionally, missing data as random or in chunks could also influence the comparisons of prediction accuracy depending on the dataset (Donders et al., 2006). Interpretations may also be influenced by the choice of error metric (e.g., *RMSE*) as different metrics have different objectives (Yozgatligil et al., 2013).

This paper describes the **imputeTestbench** package to simultaneously compare different imputation methods for univariate time series (Bokde and Beck, 2017). The goal of this package is to provide an evaluation toolset that addresses the above challenges for identifying an appropriate imputation method before more detailed analysis. This package provides several options for simulating missing observations with repeated sampling from a complete dataset. Missing values are imputed using any of several methods and then compared with a common error metric chosen by the user. Plotting functions are available to visualize the simulation methods for missing data, the predicted time series from each method, and overall evaluation of prediction accuracy between methods. Example applications are provided to demonstrate how **imputeTestbench** can be used to understand why different methods have different prediction accuracy given characteristics of the original data and characteristics of the missing data.

Overview of imputeTestbench

The theoretical foundation of **imputeTestbench** is shown in Figure 1. Comparing imputation methods begins by identifying the range of missing observations to remove and the number of repetitions for randomly removing the data. The removed data are imputed using one of several methods for each percentage of missing observations up to the maximum. Error metrics are used to identify the prediction accuracies of each imputation method for each repetition and interval of missing data.

Components of the workflow in Figure 1 are executed with the functions in **imputeTestbench**. The primary function is `impute_errors()` which is used to evaluate different imputation methods with missing data that are randomly generated from a complete dataset. The `sample_dat()` function is used to generate missing data within `impute_errors()` and includes a plotting option to demonstrate how the missing data are generated. The default error metrics for the imputed data are in the `error_functions()` function. The remaining two functions, `plot_impute()` and `plot_errors()`, are used to visualize imputation results and error summaries for the chosen methods. Dependencies include additional packages for data manipulation (**dplyr**, Wickham and Francois (2016); **reshape2**, Wickham (2007); **tidyr**, Wickham (2017)), graphing (**ggplot2**, Wickham (2009)), and imputation (**forecast**, Hyndman et al. (2017); **imputeTS**, Moritz and Bartz-Beielstein (2017); **zoo**, Zeileis and Grothendieck (2005)).

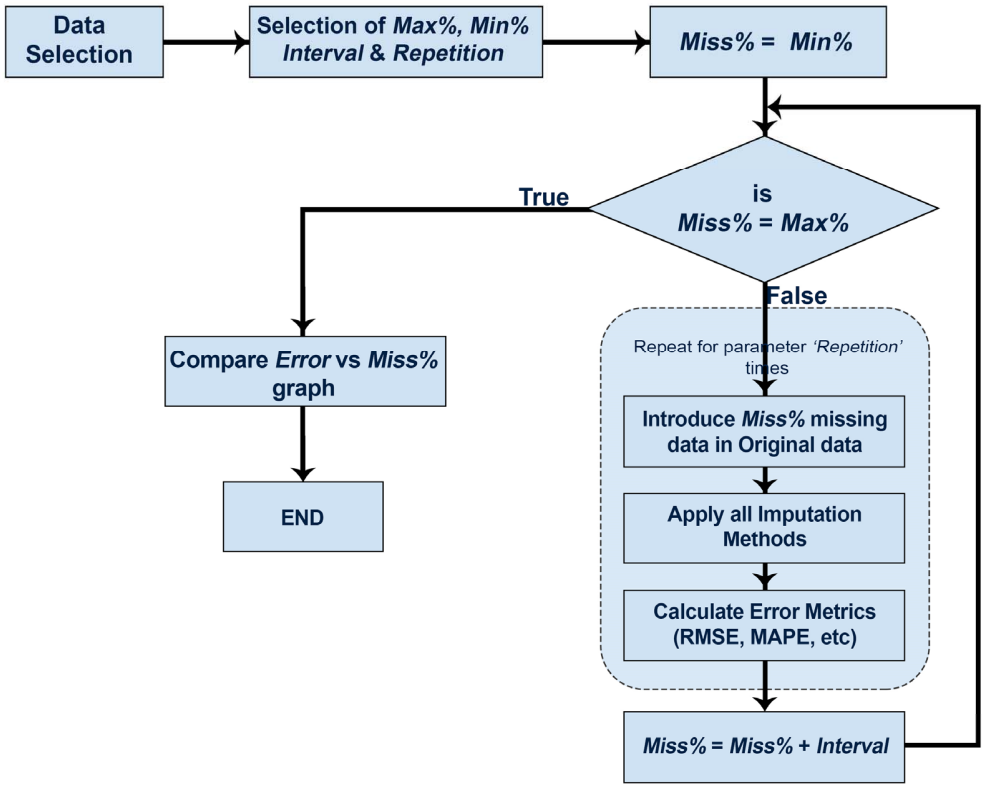


Figure 1: Workflow diagram for comparing imputation methods. *Min*% and *Max*% are minimum and maximum percent of missing values in the dataset. Details are described below.

The `impute_errors()` function:

The `impute_errors()` function evaluates the accuracy of different imputation methods based on changes in the amount and type of missing observations from the complete dataset. The default methods included in `impute_errors()` are three methods for linear interpolation (`na.approx()`, **zoo**; `na.interp()`, **forecast**; `na.interpolation()`, **imputeTS**), last-observation carried forward (`na.locf()`, **zoo**), and mean replacement (`na.mean()`, **imputeTS**). These methods are routinely applied in time series analysis, are easily understood compared to more complex approaches, and have relatively short computation times (Moritz and Bartz-Beielstein, 2017). Moreover, these methods represent a gradient from none to more complex dependence on the serial correlation of the time series - replacing missing data with overall means (`na.mean()`), replacing missing data with the last prior observation (`na.locf()`), and gap interpolation with linear methods (`na.approx()`, `na.interp()`, `na.interpolation()`). Note that the three linear methods vary considerably in the optional arguments that affect the imputation output. An expectation with the default methods included in **imputeTestbench** is varying imputation accuracy based on how each method relies on characteristics of a dataset to predict missing observations. Although we acknowledge that the effectiveness of a chosen method depends on the data, the default techniques represent a broad range that is sufficient for most applications. As noted below, additional methods can be added as needed.

The `impute_errors()` function has the following arguments:

```
impute_errors(dataIn, smps = "mcar",
  methods = c("na.approx","na.interp",
  "na.interpolation", "na.locf", "na.mean"),
  methodPath = NULL, errorParameter = "rmse",
  errorPath = NULL, blck = 50, blckper = TRUE,
  missPercentFrom = 10, missPercentTo = 90,
  interval = 10, repetition = 10, addl_arg =
  NULL)
```

`dataIn`:

A `ts` (**stats**) or numeric object that will be evaluated. The input object is a complete dataset to evaluate by simulating missing data for performance evaluation and comparison of imputation methods. The examples in the documentation use the `nottem` time series object of average air temperatures recorded at Nottingham Castle from 1920 to 1930 (**datasets** package).

`smps`:

The desired type of sampling method for removing values from the complete time series provided by `dataIn`. Options are `smps = 'mcar'` for missing completely at random (MCAR, default) and `smps = 'mar'` for missing at random (MAR). Both methods provide different approaches to generating missing data in time series. In general, MCAR removes individual observations where the likelihood of a single observation being removed does not depend on whether observations closer in time have also been removed. By contrast, MAR removes observations in continuous blocks such that the likelihood of an observation being removed depends on whether observations closer in time have also been removed. The methods are described in detail in the section Sampling methods for missing observations.

`methods`:

Methods that are used to impute the missing values generated by `smps`: replace with means (`na.mean()`), last-observation carried forward (`na.locf()`), and three methods of linear interpolation (`na.approx()`, `na.interp()`, `na.interpolation()`). All five default methods are used unless the argument is changed by the user. For example, `methods = 'na.approx'` will use only `na.approx()` with `impute_errors()`. Methods not included with the default options can be added by including the name of the function in `methods` and providing the path to the script in `methodPath`. Additional arguments passed to each method can be included in `addl_arg` described below.

`methodPath`:

A character string for the path of the user-supplied script that includes one to many functions passed to `methods`. The path can be absolute or relative within the current working directory for the R session. The `impute_errors()` function sources the file indicated by `methodPath` to add the user-supplied function to the global environment.

`errorParameter`:

The error metric used to compare the true, observed values from `dataIn` with the imputed values. Metrics included with **imputeTestbench** are root-mean squared error (*RMSE*), mean absolute percent error (*MAPE*) and mean absolute error (*MAE*). The metric can be changed using `errorParameter = 'rmse'` (default), `'mape'`, or `'mae'`. Formulas for the error metrics are as follows:

$$RMSE = \sqrt{\frac{\sum_{i=1}^{n} (x_i - \hat{x}_i)^2}{n}} \tag{1}$$

$$MAPE = 100 \cdot \frac{\sum_{i=1}^{n} |(x_i - \hat{x}_i)/x_i|}{n} \tag{2}$$

$$MAE = \frac{\sum_{i=1}^{n} |x_i - \hat{x}_i|}{n} \tag{3}$$

where n is the total number of missing observations, x are the actual observations, and $\hat{x}$ are the imputed observations. Additional error measures can be provided as user-supplied functions.

`errorPath`:

A character string for the path of the user-supplied script that includes one to many error methods passed to `errorParameter`.

`blck`:

The block size for missing data if the sampling method is at random, `smps = 'mar'`. The block size can be specified as a percentage of the total amount of missing observations to remove or as a number of time steps in the input dataset. For example, if `blck = 50` and `blkper = TRUE` (indicating `blck` is a percentage), each block has a size that is 50% of the total amount of observations to remove. A time series with 100 observations will have two missing chunks of 20 observations each if a total of 40% of the observations are removed and each chunk is 50% of the total. The total amount of observations to remove depends on values inherited from `missPercentFrom`, `missPercentTo`, and `interval`.

`blckper`:

A logical value indicating if the value for `blck` is a percentage (`blckper = TRUE`) of the total number of observations to remove or a sequential number of time steps (`blckper = FALSE`) to remove for each block. This argument only applies if `smps = 'mar'`.

`missPercentFrom`, `missPercentTo`:

The minimum and maximum percentages of missing values, respectively, that are introduced in `dataIn`. Appropriate values for these arguments are 10 to 90, indicating a range from few missing observations to almost completely absent observations.

`interval`:

The interval of missing data from `missPercentFrom` to `missPercentTo`. The default value is 10% such that missing percentages in `dataIn` are evaluated from 10% to 90% at an interval of 10%, i.e., 10%, 20%, 30%, ..., 90%. Combined, these arguments are identical to `seq(from = 10,to = 90,by = 10)`.

`repetition`:

The number of repetitions at each `interval`. Missing values are placed randomly in the original data such that multiple repetitions must be evaluated for a robust comparison of the imputation methods.

Considering the default values for the above arguments, the `impute_errors()` function returns an `"errprof"` object as the *error profile* for the imputation methods:

```
library(imputeTestbench)
set.seed(123)
a <- impute_errors(dataIn = nottem)
a
## $Parameter
## [1] "rmse"
##
## $MissingPercent
## [1] 10 20 30 40 50 60 70 80 90
##
## $na.approx
## [1] 0.87 1.49 1.87 3.01 3.91 4.58 6.75
## [8] 8.68 9.82
##
## $na.interp
## [1] 0.76 1.09 1.41 1.67 1.91 2.09 2.30
## [8] 2.53 2.92
##
## $na.interpolation
## [1] 0.87 1.49 1.87 3.01 3.91 4.58 6.75
## [8] 8.68 9.82
##
## $na.locf
## [1]  1.8  2.6  3.7  5.4  6.5  7.3  9.1
## [8] 11.0 11.0
##
## $na.mean
## [1] 2.6 3.8 4.6 5.4 6.2 6.7 7.2 7.6 8.4
```

The "errprof" object is a list with seven elements. The first element, Parameter, is a character string of the error metric used for comparing imputation methods. The second element, MissingPercent, is a numeric vector of the missing percentages that were evaluated in the input dataset. The remaining five elements show the average error for each imputation method at each interval of missing data in MissingPercent. The averages at each interval are based on the repetitions specified in the initial call to impute_errors(), where the default is repetition = 10. Although the print method for the "errprof" object returns a list, the object stores the unique error estimates for every imputation method, repetition, and missing data interval. These values are used to estimate the averages in the printed output and to plot the distribution of errors with plot_errors() shown below. All error values can be accessed from the errall attribute, i.e., attr(a,'errall').

Viewing results from impute_errors()

The plot_errors() function can be used to view summaries of the imputation errors for each method. This function uses the "errprof" object as input and returns a graph of error values to compare results from the methods across the range of missing data. Three plot types are provided by plot_errors() and are specified with the plotType argument using one of three values: "boxplot", "bar", or "line". The default value is plotType = 'boxplot' that graphs the distribution of error values for each method and missing data interval using boxplot summaries (i.e., 25th, 50th, and 75th percentile shown by the box, whiskers as 1.5 times interquartile range, and outliers beyond). The boxplots are created using all error values stored in the 'errall' attribute of the "errprof" object (Figure 2).

```
plot_errors(a)
```

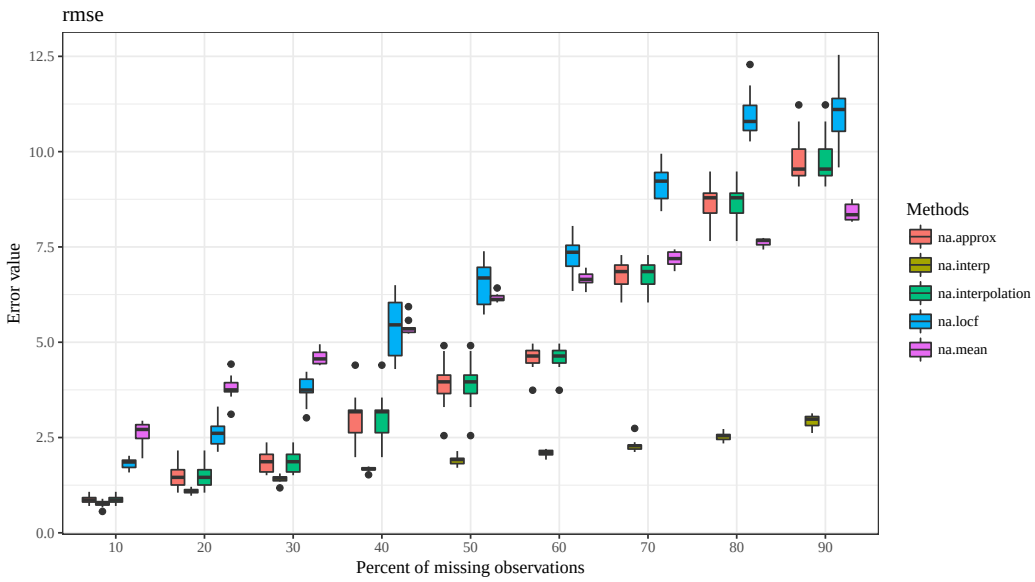


Figure 2: Distribution of error values from the "errprof" object for each imputation method and interval of missing observations. The plot is created with plot_errors() using the boxplot option.

The bar and line options for plotType show the average error values for each repetition. Similar information is shown as the boxplot option, although the range of error values for each imputation method is not shown ('line' option shown in Figure 3).

```
plot_errors(a, plotType = 'line')
```

Sampling methods for missing observations

The impute_errors() function uses sample_dat() to remove observations for imputation from the input dataset. Observations are removed using one of two methods relevant for univariate time series: MCAR and MAR. The MCAR sampling scheme assumes all observations have equal probability of being selected for removal and is appropriate for understanding imputation accuracy with univariate time series that are not serially correlated (Rubin, 1976). Conversely, the MAR sampling scheme selects observations in blocks such that the probability of selection for a single observation depends on

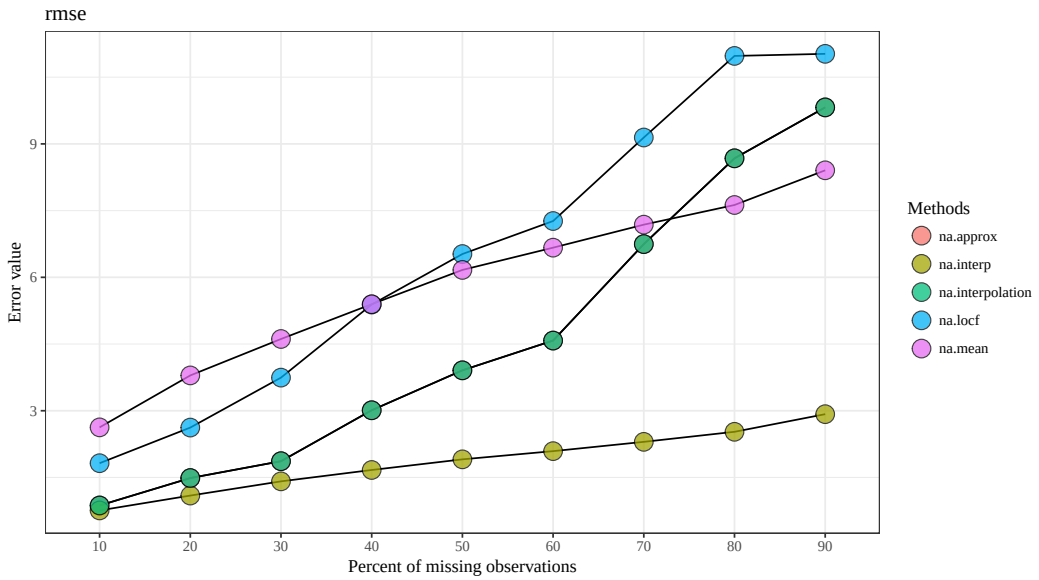


Figure 3: Average error values for each imputation method and interval of missing observations. The `line` option is used for `plot_errors()`.

whether an observation closer in time was also selected (Schafer and Graham, 2002). The MAR scheme is appropriate for time series with serial correlation. The `sample_dat()` function has the following syntax:

```
sample_dat(datin, smps = "mcar", repetition = 10, b = 50, blck = 50, blckper = TRUE,
  plot = FALSE)
```

`datin`:

Input `"ts"` object or numeric vector, inherited from `dataIn` from `impute_errors()`.

`smps, repetition, blck, blckper`:

Arguments that are inherited as is from `impute_errors()` indicating the sampling type (`smps`), number of repetitions for each missing data type (`repetition`), block size (`blck`), and block type as percentage or count (`blckper`).

`b`:

Numeric indicating the total amount of missing data as a percentage to remove from the complete time series. The arguments `missPercentFrom`, `missPercentTo`, and `interval` from `impute_errors()` define `b` for each simulation of missing observations with `sample_dat()`. For example, missing data will be simulated with `sample_dat()` for percentages of the total sample size as `b = 10,20,...,90` if `missPercentFrom = 10`, `missPercentTo = 90`, and `interval = 10` for `impute_errors()`.

`plot`:

Logical indicating if a plot is returned that shows one repetition of the sampling scheme defined by the arguments (Figure 4).

The MCAR sampling scheme is used if `smps = 'mcar'`, where the only relevant arguments are `missPercentFrom`, `missPercentTo`, and `interval` from `impute_errors()` for the missing data. The amount of data to remove for each interval is passed to the `b` argument in `sample_dat()`. Alternatively, the MAR sampling scheme requires additional arguments to control the block size for removing data in continuous chunks, in addition to the total amount of data to remove defined by `b`. The block size argument, `blck`, can be given as a percentage or as number of observations in sequence. The type of block size passed to `blck` is controlled by `blckper`, where `blckper = TRUE` indicates a percentage and `FALSE` indicates a count for `blck`. For example, if the total sample size of the dataset is 1000, `b`

= 50, blck = 20, and blckper = TRUE means half the dataset is removed (b = 50, 500 observations) and each block will have 100 observations (20% of 500). For both percentages and counts, the blocks are automatically selected until the total amount of missing data is equal to that specified by b. Final blocks may be truncated to make the total amount of missing observations equal to b. The starting location of each block is selected at random and overlapping blocks are not uniquely counted for the required sample size given by b (i.e., blocks larger than that specified by blck may occur of two separate blocks overlap).

The sample_dat() function is typically not used independently of impute_errrors(), although an optional plotting argument is provided to visualize different sampling schemes for removing data. Figure 4 shows examples of sampling with MCAR and MAR.

The plot_impute() function

The third plotting function available in **imputeTestbench** is plot_impute(). This function returns a plot of the imputed values for each imputation method in impute_errors() for one repetition of sampling for missing data. The plot shows the results in individual panels for each method with the points colored as retained or imputed (i.e., original data not removed and imputed data that were removed). An optional argument, showmiss, can be used to plot the original values as open circles that were removed from the data. The plot_impute() function shows results for only one simulation and missing data type (e.g., smps = 'mcar' and b = 50). Although the plot from plot_errors() is a more accurate representation of overall performance of each method, plot_impute() is useful to better understand how the methods predict values for a sample dataset (Figure 5).

```
plot_impute(dataIn = nottem, showmiss = T)
```

Including additional arguments

Additional arguments for the imputation functions used by impute_errors() can easily be modified. This is useful for changing the default arguments, particularly for the three linear interpolation methods (na.approach(), na.interp(), na.interpolation()). Although these methods are very similar in the default configuration, the amount of flexibility varies considerably. For example, na.interpolation() provides spline interpolation as an alternative that is not provided by the other functions. Using each method without modifying the additional arguments is not suggested given that no information is gained by comparing the linear interpolation methods with the default setup.

Accordingly, additional arguments can be passed to any of the imputation methods using the addl_arg argument. These additional arguments are passed as a list of lists, where the list contains one to many elements that are named by the methods in methods. The elements of the list are lists with arguments that are specific to each imputation method. For example, the default imputation function na.mean() has an additional option argument that specifies the algorithm for missing values, where possible values are "mean", "median", and "mode". This argument can be changed from the default option = "mean" with addl_arg in impute_errors(), as shown below. Arguments to user-supplied imputation functions (below) can be changed similarly.

```
# changing the option argument for na.mean
impute_errors(dataIn = nottem,
  addl_arg = list(na.mean = list(option = 'mode'))
  )
```

Adding imputation methods and error metrics

Additional imputation methods saved as an R script can be added to impute_errors(). Attention should be given to the format of the user-supplied function shown below. The time series data (as numeric or "ts" object) with missing values is required as input and the return value is the input dataset with imputed values, where the input and output objects have the same length. The impute_errors() function will not process the data correctly if the format is incorrect. The file path for the R script is supplied as an input string to the methodPath argument and the function name is added to the methods argument for the impute_errors() function.

```
# function to include with impute_errors
new_imp <- function(In){
  out <- imputeTS::na.random(In)
  out <- as.numeric(out)
```

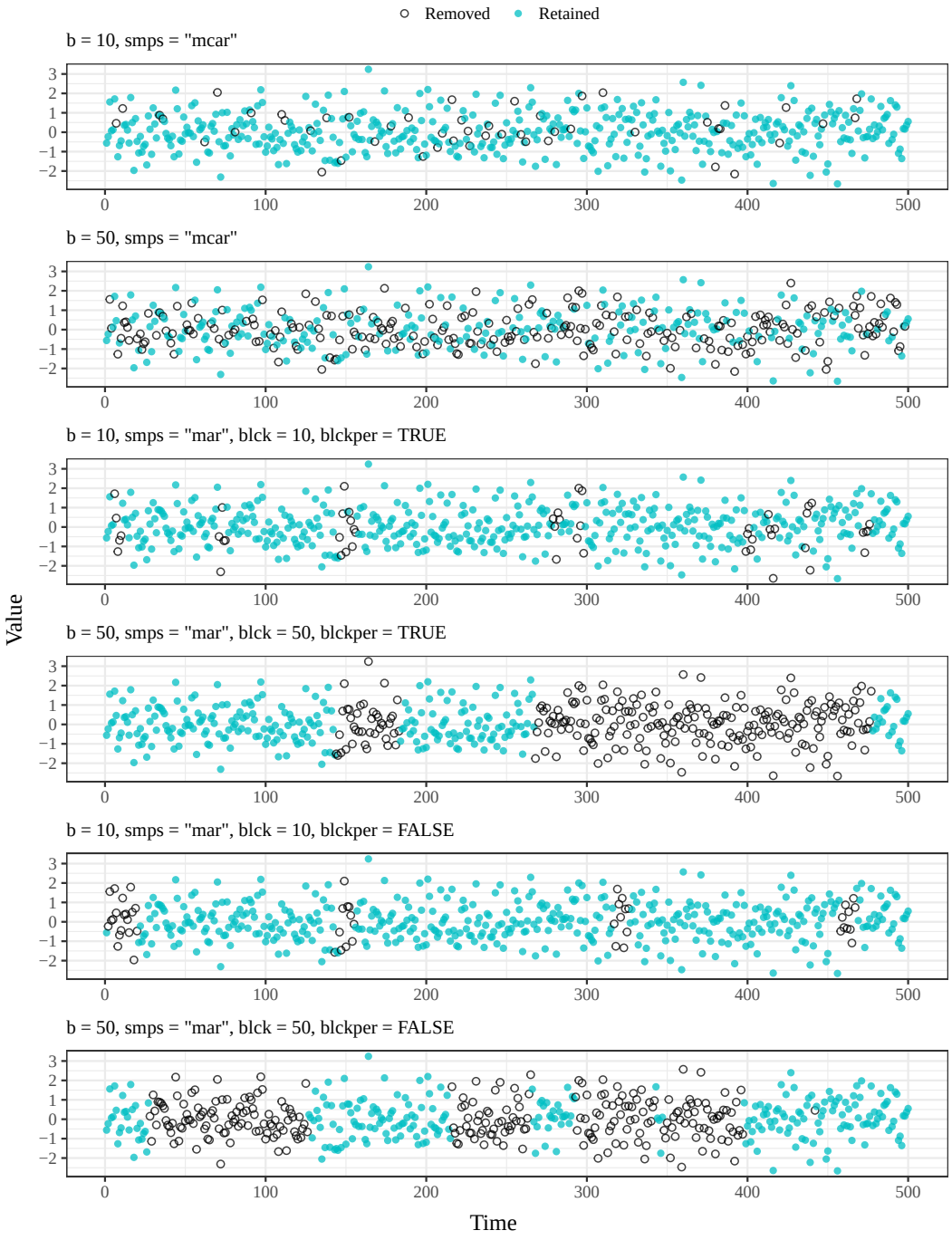


Figure 4: Examples of sampling schemes for missing data provided by `sample_dat()`, plotted using the argument `plot = T`. Values to be removed and imputed are shown as open circles and the data to be kept are in blue. From top to bottom, sampling is MCAR with 10% missing, MCAR with 50% missing, MAR with 10% missing and block size 10% of total missing, MAR with 50% missing and block size 50% of total missing, MAR with 10% missing and block size of ten observations, and MAR with 50% missing and block size of fifty observations.

```
  return(out)
}
```

As described above, the error metrics included with **imputeTestbench** are *RMSE*, *MAE*, and *MAPE*. These metrics provide different information and contrasting approaches to evaluate imputation methods. For example, *RMSE* is a commonly used metric that maintains the scale of observations in the input data. The *MAE* metric is similar but more interpretable as the differences are in proportion to the absolute values of the errors, which differs from *RMSE* that uses the sum of squared deviations. The *MAPE* metric is a simple extension of *MAE* that scales the output by the range of observations in

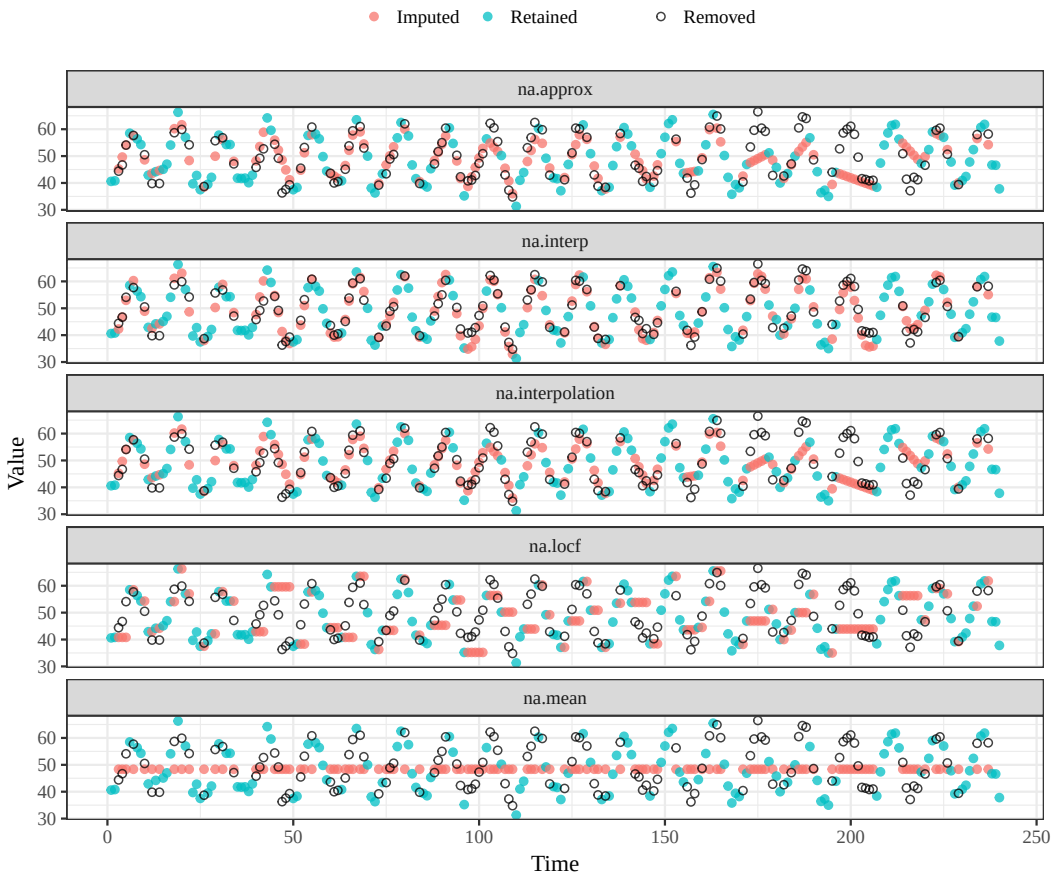


Figure 5: Output from the `plot_impute()` function that shows the data that were retained (blue), removed (open circles), and imputed (red).

the input data and is useful for comparing datasets that differ in scale. Users should choose a metric based on the information provided by each. For example, imputation comparisons with different datasets should use the `MAPE` metric that standardizes the scale of assessment.

Each of the metrics are called by the `error_functions()` function internally within `impute_errors()`. Additional error metrics are added using an approach similar to that used for adding imputation methods. The following example shows use of the percent change in variance (*PCV*, Tak et al. (2016)) as an alternative error metric:

$$PCV = \frac{var(\bar{V}) - var(V)}{var(V)} \tag{4}$$

The user-supplied error function must include two arguments as input, the first being a vector of observed values and the second being a vector of imputed missing values equal in length to the first. The function must also return a single value as a summary of the errors or differences. As before, the new error function should be saved as an R script. The file path is added to the `errorPath` argument and the error function name is added as a character string to the `errorParameter` argument for `impute_errors()`.

```
# error metric to include with impute_errors
pcv <- function(dataIn, imputed)
{
  d <- (var(imputed) - var(dataIn)) *
    100/ var(imputed)
  d <- as.numeric(d)
  return(d)
}
```

Demonstration of imputeTestbench

This example demonstrates how **imputeTestbench** can be used to compare imputation methods, and more importantly, how it can be used to better understand the effects of time series characteristics on prediction accuracies. The objective of the comparison is to relate prediction accuracies of each method to the time series characteristics of each dataset, including a description of how the amount and type of missing data influence the results. Three univariate datasets with different characteristics are evaluated (Figure 6). The first dataset, `nrm`, is a sample of 100 random observations from a standard normal distribution to simulate a dataset with no temporal correlation. The second dataset, `austres`, is a `"ts"` object of Australian population in thousands, measured quarterly from 1971 to 1994 (Brockwell and Davis, 1996). This dataset includes a simple correlation structure with minimal random noise. The final dataset is `nottem` as described above. This dataset is characterized by a cyclical or repeating seasonal component. Lagged correlations show the differences in the temporal dependencies between the datasets (Figure 6).

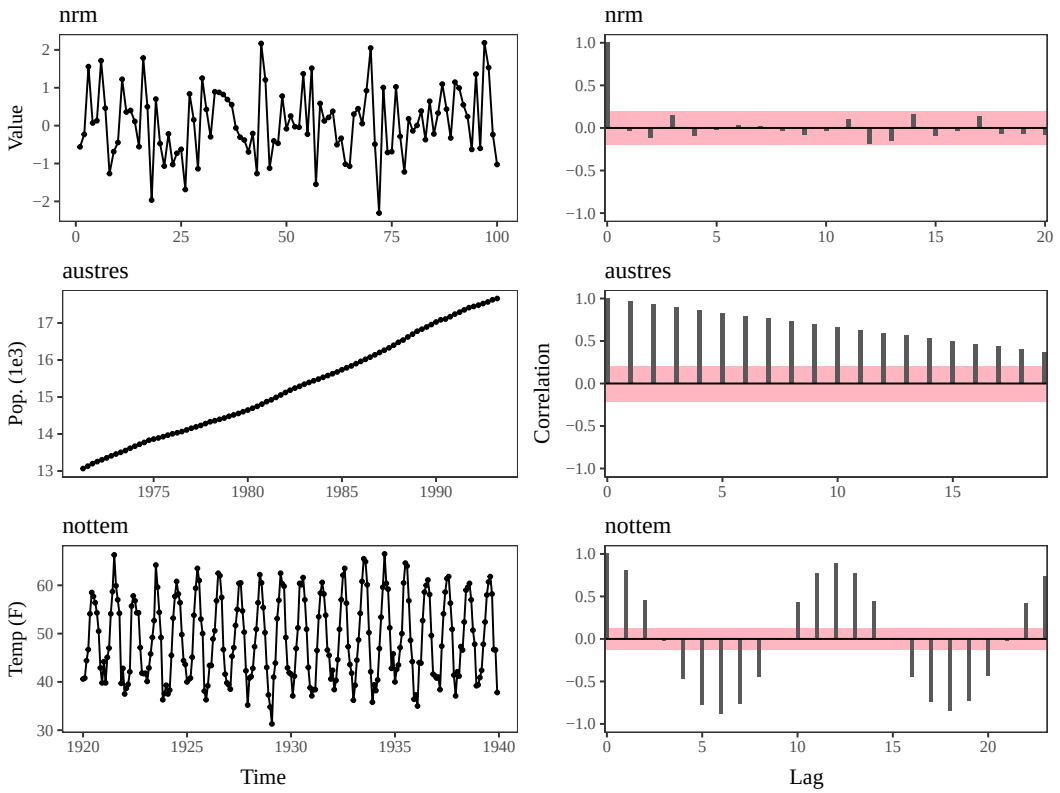


Figure 6: Three datasets used to demonstrate application of **imputeTestbench**. The top dataset, `nrm`, was created with random samples from a normal distribution. The bottom two datasets are `austres` and `nottem` from the **datasets** package. Each dataset has different temporal correlation structures shown at different lags (right column). The light red shading indicates a threshold beyond which correlations are significant ($\alpha = 0.05$).

Each dataset was evaluated by simulating missing observations from 10% to 90% of the complete data using MCAR (`smps = 'mcar'`) and MAR (`smps = 'mar'`) sampling. The size of each chunk (`blck` argument) for MAR sampling was evaluated at 20% and 100% of the total percentage of missing observations to evaluate an effect of chunk size (small chunks to one large chunk) on imputation accuracy. As such, the analysis evaluated imputation accuracy between the default methods as affected by dataset type (`nrm` - no correlation structure, `austres` - simple correlation, `nottem` - seasonal correlation) and characteristics of the missing observations (MAR, MCAR, varying amounts of missing data and chunk sizes). All comparisons used the *MAPE* metric to evaluate imputation errors given scale differences between the datasets. The following code demonstrates the analysis setup.

```
# load packages, set seed
library(tidyverse)
library(imputeTestbench)
set.seed(123)
```

```
# create nested list of datasets
dats <- list(
  nrm = rnorm(100),
  aust = austres,
  nott = nottem
  ) %>%
  enframe('nms', 'dataIn')

# create parameters to vary with imputeTestbench
prms <- crossing(
  smps = c('mcar', 'mar'),
  blck = c(20, 100),
  nms = dats$nms
  ) %>%
  mutate( # blck does not matter for mcar
    blck = ifelse(smps %in% 'mcar', NA, blck)
  ) %>%
  unique

# join parameters with data
# map parameter and data to impute_errors
ests <- prms %>%
  left_join(dats, by = 'nms') %>%
  mutate(
    est = pmap(
      list(smps = smps, blck = blck, dataIn = dataIn),
      .f = impute_errors,
      errorParameter = 'mape'
      )
    )
```

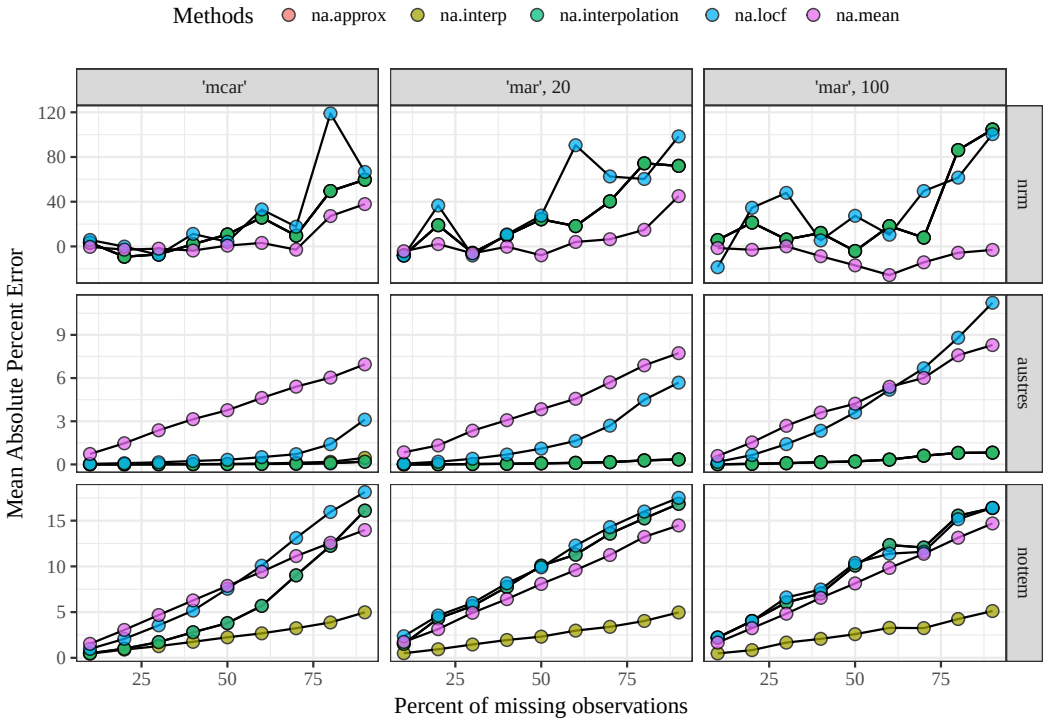


Figure 7: Comparisons of prediction errors for three datasets with different temporal correlation structures (see Figure 6). Missing observations were simulated from the complete datasets as MCAR and MAR with missing chunks of different sizes from 20% and 100% of the total. The total percentage of missing observations from the complete datasets varied from 10% to 90%.

The prediction errors varied considerably between the datasets, imputation method, and type

of simulation for missing observations (Figure 7). As expected, the type of simulation (MCAR and MAR with different chunk sizes) did not have a noticeable influence on prediction accuracy for the normally distributed dataset with no correlation structure (`nrm`). This dataset is described by only two parameters (mean and standard deviation) and the observations are completely independent such that imputation accuracy was similar for both MCAR and MAR sampling, although accuracy decreased with the addition of missing observations which is not unexpected. The `na.mean()` function generally outperformed the other imputation methods that incorporate some level of information about the relationship between observations. The `nrm` dataset has observations that are not correlated and the use of imputation methods that leverage temporal dependence in the input dataset is expected to induce bias in the imputation, as shown by larger errors.

By comparison, the prediction errors for the `austres` and `nottem` datasets differed in both the type of simulation for missing observations and the imputation method. For `austres`, linear interpolations consistently produced imputations with the least error and each of the three linear interpolation methods (`na.approx()`, `na.interp()`, `na.interpolation()`) performed equally well regardless of simulation type for missing data. This result is not surprising given that `austres` can be described as a monotonic linear increase with minimal variation. That is, imputing a straight line between observations reproduces the original dataset with high accuracy. However, relative prediction accuracies between methods were affected by the type of simulation for removing observations. Imputations with `na.locf()` method had increasing error with increasing size of the missing chunk and errors exceeded `na.mean()` for large chunk sizes and high percentages of missing data (i.e., > 60% missing as one large chunk). These results can be explained by viewing sample imputations with `plot_impute()` (Figure (8). As in Figure 7, the `na.mean()` function performs poorly for most scenarios because it does not capture the linear increase through time. However, the `na.locf()` and `na.approx()` functions perform equally well for small percentages of missing data but error values diverge for larger percentages. Errors for `na.locf()` exceed those for `na.mean()` when large chunks are removed given that the latter produces less biased estimates, although the method performs poorly overall.

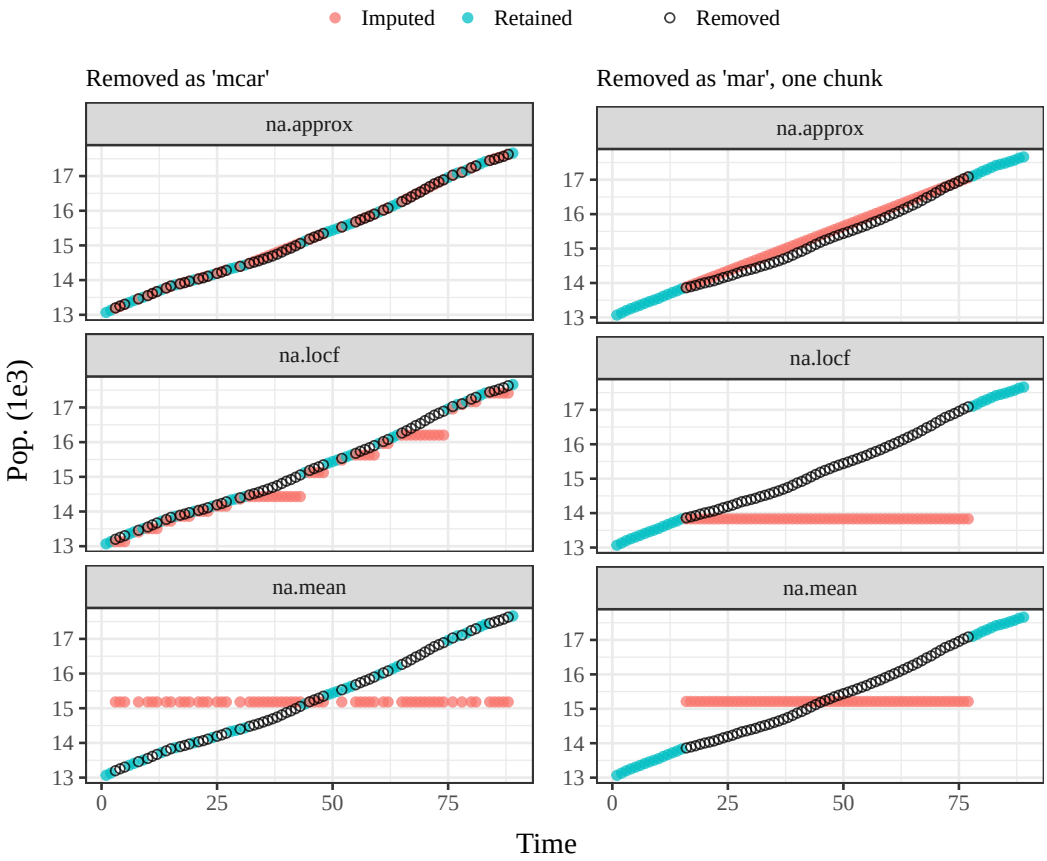


Figure 8: An example from `plot_impute()` of imputed values for the austres dataset using `na.approx`, `na.locf`, and `na.mean`. Seventy percent of observations were removed for each dataset. The left column shows the observations removed using MCAR and the right shows the observations removed using MAR.

Finally, imputations of the `nottem` dataset had similar prediction errors independent of how

the data were removed, although the imputation methods varied considerably. Specifically, the `na.interp()` function consistently had lower prediction errors for all percentages of missing observations. Interestingly, the `na.approx()` and `na.interpolation()` functions did not have similar performance as `na.interp()`, although all three methods are similarly documented as linear interpolations. Examples in Figure 5 demonstrate differences in the imputations between the three methods. Both `na.approx()` and `na.interpolation()` do not describe the seasonal variation in the data that is well-described by `na.interp()`. As noted in the documentation, `na.interp()` optionally performs periodic decomposition for seasonal time series based on characteristics of the input data (Hyndman et al., 2017). This highlights the need to carefully understand the assumptions of each method and that the default behavior between ostensibly similar approaches could vary. Users should be familiar with the help documentation and are advised to make use of the `addl_args` argument in `impute_errors()` to modify the default behavior of the chosen imputation functions. The **imputeTestbench** package provides sufficient flexibility to accommodate differences among methods for each approach evaluated with `impute_errors()`.

Summary

The applied example demonstrated the value of **imputeTestbench** for informing the use of imputation methods as a necessary step towards more formal time series analysis. For all examples, a complete dataset was used to demonstrate how characteristics of the temporal correlation structure can influence the prediction accuracy. The default imputation methods in the package have different approaches to imputing missing values that vary in the amount of dependence on the correlation structure of input data. As expected, the accuracy of each imputation method varied depending on the dataset and a general conclusion is that users should carefully evaluate the correlation structure and periodic components of actual data as an approach to choosing an imputation method. The **imputeTestbench** package greatly facilitates this preliminary step by simultaneously comparing different methods and considering the type and amount of missing observations. In practice, observational data that contain missing values cannot be used directly with the package because the intent is to evaluate hypothetical scenarios of missing observations with complete data. As such, the sample dataset used with **imputeTestBench** should have characteristics similar to the dataset for which imputation is needed. Our applied example evaluated three time series data with different characteristics and additional comparisons of datasets with more complex temporal dependencies could be helpful towards informing practical applications.

We also demonstrated how the package can be modified to include additional imputation methods or comparison metrics. By default, the package provides a core set of existing imputation methods (`na.approx()`, `na.interp()`, `na.interpolation()`, `na.locf()`, and `na.mean()`), which are simultaneously compared using *RMSE*, *MAE*, or *MAPE* error metrics. These simple methods will likely be sufficient for most users, although we recognize that the comparison of other methods and error metrics will be required in more advanced cases. As such, the package allows users to include additional imputation methods for comparison, which could be particularly useful given the capability of R to interface with other programming languages (e.g, **Rcpp** for compiled languages, Eddelbuettel and François (2011); **matlabr** for MatLab, Muschelli (2016)). As such, the simple architecture of **imputeTestbench** to add or remove multiple methods and error metrics makes it a robust and useful tool to evaluate existing and proposed imputation techniques for univariate time series.

Acknowledgments

We thank the R user community for providing feedback that improved earlier versions of the software. The views expressed in this paper are those of the authors and do not necessarily reflect the views or policies of the U.S. Environmental Protection Agency.

Bibliography

N. Bokde and M. W. Beck. *imputeTestbench: Test Bench for the Comparison of Imputation Methods*, 2017. URL https://cran.r-project.org/package=imputeTestbench. R package version 3.0.1. [p219]

G. E. P. Box, G. M. Jenkins, G. C. Reinsel, and G. M. Ljung. *Time Series Analysis: Forecasting and Control*. John Wiley & Sons, Hoboken, New Jersey, 5th edition, 2015. ISBN 978-1-118-67502-1. [p218]

P. J. Brockwell and R. A. Davis. *Introduction to Time Series and Forecasting*. Springer-Verlag, 1996. ISBN 978-1-4757-2526-1. [p228]

A. R. T. Donders, G. J. M. G. van der Heijden, T. Stijnen, and K. G. M. Moons. Review: A gentle introduction to imputation of missing values. *Journal of Clinical Epidemiology*, 59:1087–1091, 2006. URL https://doi.org/10.1016/j.jclinepi.2006.01.014. [p218, 219]

D. Eddelbuettel and R. François. Rcpp: Seamless R and C++ integration. *Journal of Statistical Software*, 40(8):1–18, 2011. URL https://doi.org/10.18637/jss.v040.i08. [p231]

V. Gomez and A. Maravall. Estimation, prediction, and interpolation for nonstationary series with the Kalman filter. *Journal of the American Statistical Association*, 89(426):611–624, 1994. URL https://doi.org/10.1080/01621459.1994.10476786. [p218]

P. G. Gould, A. B. Koehler, J. K. Ord, R. D. Snyder, R. J. Hyndman, and F. Vahid-Araghi. Forecasting time series with multiple seasonal patterns. *European Journal of Operational Research*, 191(1):207–222, 2008. URL https://doi.org/10.1016/j.ejor.2007.08.024. [p218]

J. Honaker and G. King. What to do about missing values in time-series cross-section data. *American Journal of Political Science*, 54(2):561–581, 2010. URL https://doi.org/10.1111/j.1540-5907.2010.00447.x. [p218]

R. J. Hyndman, M. O'Hara-Wild, C. Bergmeir, S. Razbash, and E. Wang. *Forecast: Forecasting Functions for Time Series and Linear Models*, 2017. URL http://github.com/robjhyndman/forecast. R package version 8.1. [p219, 231]

R. Jörnsten, M. Ouyang, and H.-Y. Wang. A meta-data based method for DNA microarray imputation. *BMC Bioinformatics*, 8(1):109, 2007. URL https://doi.org/10.1186/1471-2105-8-109. [p219]

H. Li, C. Zhao, F. Shao, G.-Z. Li, and X. Wang. A hybrid imputation approach for microarray missing value estimation. *BMC Genomics*, 16(Suppl 9):S1, 2015. URL https://doi.org/10.1186/1471-2164-16-s9-s1. [p219]

J. M. Mendel. Uncertainty, fuzzy logic, and signal processing. *Signal Processing*, 80(6):913–933, 2000. URL https://doi.org/10.1016/s0165-1684(00)00011-6. [p218]

S. Moritz and T. Bartz-Beielstein. imputeTS: Time Series Missing Value Imputation in R. *The R Journal*, 2017. URL https://journal.r-project.org/archive/2017/RJ-2017-009/index.html. [p219, 220]

J. Muschelli. *Matlabr: An Interface for MATLAB Using System Calls*, 2016. URL https://CRAN.R-project.org/package=matlabr. R package version 1.1.3. [p231]

C. D. Nguyen, J. B. Carlin, and K. J. Lee. Diagnosing problems with imputation models using the Kolmogorov-Smirnov test: a simulation study. *BMC Medical Research Methodology*, 13(1):1, 2013. URL https://doi.org/10.1186/1471-2288-13-144. [p219]

B. Ran, H. Tan, J. Feng, Y. Liu, and W. Wang. Traffic speed data imputation method based on tensor completion. *Computational Intelligence and Neuroscience*, 2015:22, 2015. URL https://doi.org/10.1155/2015/364089. [p219]

D. B. Rubin. Inference and missing data. *Biometrika*, 63(3):581–592, 1976. URL https://doi.org/10.1093/biomet/63.3.581. [p218, 223]

J. L. Schafer. *Analysis of Incomplete Multivariate Data*. Chapman and Hall, CRC, Boca Raton, Florida, 1997. ISBN 978-0412040610. [p218]

J. L. Schafer and J. W. Graham. Missing data: Our view of the state of the art. *Psychological Methods*, 7 (2):147–177, 2002. URL https://doi.org/10.1037//1082-989x.7.2.147. [p218, 224]

R. H. Shumway and D. S. Stoffer. *Time Series Analysis and Its Applications: With R Examples*. Springer-Verlag, New York, New York, 3rd edition, 2011. [p218]

S. Tak, S. Woo, and H. Yeo. Data-driven imputation method for traffic data in sectional units of road links. *IEEE Transactions on Intelligent Transportation Systems*, 17(6):1762–1771, 2016. URL https://doi.org/10.1109/tits.2016.2530312. [p219, 227]

H. Wickham. Reshaping data with the reshape package. *Journal of Statistical Software*, 21(12):1–20, 2007. URL https://doi.org/10.18637/jss.v021.i12. [p219]

H. Wickham. *Ggplot2: Elegant Graphics for Data Analysis*. Springer-Verlag, 2009. ISBN 978-0-387-98140-6. URL http://ggplot2.org. [p219]

H. Wickham. *Tidyr: Easily Tidy Data with* `spread()` *and* `gather()` *Functions*, 2017. URL https://github.com/tidyverse/tidyr. R package version 0.6.3. [p219]

H. Wickham and R. Francois. *Dplyr: A Grammar of Data Manipulation*, 2016. URL https://CRAN.R-project.org/package=dplyr. R package version 0.7.1. [p219]

C. Yozgatligil, S. Aslan, C. Iyigun, and I. Batmaz. Comparison of missing value imputation methods in time series: The case of Turkish meteorological data. *Theoretical and Applied Climatology*, 112(1-2): 143–167, 2013. URL https://doi.org/10.1007/s00704-012-0723-x. [p218, 219]

A. Zeileis and G. Grothendieck. Zoo: S3 infrastructure for regular and irregular time series. *Journal of Statistical Software*, 14(6):1–27, 2005. URL https://doi.org/10.18637/jss.v014.i06. [p219]

X. Zhu, S. Zhang, Z. Jin, Z. Zhang, and Z. Xu. Missing value estimation for mixed-attribute data sets. *IEEE Transactions on Knowledge and Data Engineering*, 23(1):110–121, 2011. URL https://doi.org/10.1109/tkde.2010.99. [p218, 219]

Marcus W Beck
USEPA National Health and Environmental Effects Research Laboratory, Gulf Ecology Division
1 Sabine Island Drive, Gulf Breeze, FL 32651
USA
Current address: *Southern California Coastal Water Research Project*
3535 Harbor Blvd, Suite 110, Costa Mesa, CA, 92626
USA
marcusb@sccwrp.org

Neeraj Bokde
Visvesvaraya National Institute of Technology, Nagpur
North Ambazari Road, Nagpur
India
neeraj.bokde@students.vnit.ac.in

Gualberto Asencio-Cortés
Universidad Pablo de Olavide
ES-41013, Sevilla
Spain
guaasecor@upo.es

Kishore Kulat
Visvesvaraya National Institute of Technology, Nagpur
North Ambazari Road, Nagpur
India
kdkulat@ece.vnit.ac.in

ICSOutlier: Unsupervised Outlier Detection for Low-Dimensional Contamination Structure

by Aurore Archimbaud, Klaus Nordhausen, and Anne Ruiz-Gazen

Abstract Detecting outliers in a multivariate and unsupervised context is an important and ongoing problem notably for quality control. Many statistical methods are already implemented in R and are briefly surveyed in the present paper. But only a few lead to the accurate identification of potential outliers in the case of a small level of contamination. In this particular context, the Invariant Coordinate Selection (ICS) method shows remarkable properties for identifying outliers that lie on a low-dimensional subspace in its first invariant components. It is implemented in the **ICSOutlier** package. The main function of the package, `ics.outlier`, offers the possibility of labelling potential outliers in a completely automated way. Four examples, including two real examples in quality control, illustrate the use of the function. Comparing with several other approaches, it appears that ICS is generally as efficient as its competitors and shows an advantage in the context of a small proportion of outliers lying in a low-dimensional subspace. In quality control, the method may help in properly identifying some defective products while not detecting too many false positives.

Introduction

The unsupervised detection of multivariate outliers is a timeless subject of interest in statistics, see Aggarwal (2017), Hodge and Austin (2004), Hadi et al. (2009) for a complete overview. Indeed, this can be the goal of the analysis, like in fraud detection, in medical applications or manufacturing-defect detection. It can also be used for preprocessing in almost any statistical analysis that is sensitive to the presence of outliers, e.g. Principal Component Analysis (PCA). Many statistical methods exist and have already been implemented in R through tens of packages. Only some of these packages are dedicated to the unsupervised context and are mentioned below. One of the most common methods for multivariate outlier detection is the Mahalanobis Distance (MD), and many packages are based on this distance: **mvoutlier** (Filzmoser and Gschwandtner, 2017), **CerioliOutlierDetection** (Green and Martin, 2017a), **rrcovHD** (Todorov, 2016), **faoutlier** (Chalmers and Flora, 2015). Additionally, traditional methods such as the angle-based methods are implemented in the packages **abodOutlier** (Jimenez, 2015), **HighDimOut** (Fan, 2015), the distribution-based methods in **alphaOutlier** (Rehage and Kuhnt, 2016), **extremevalues** (van der Loo, 2010), **HDoutliers** (Fraley, 2016), **outliers** (Komsta, 2011), the density-based methods in **DMwR2** (Torgo, 2016), **HighDimOut**, **ldbod** (Williams, 2017), **Rlof** (Hu et al., 2015) and the depth-based methods in **depth** (Genest et al., 2017). Finally, other approaches have been implemented, such as the one based on Projection Pursuit (PP) in **REPPlab** (Fischer et al., 2016b), and PCA in **OutlierDC** (Eo and Cho, 2014), **pcadapt** (Luu and Blum, 2017), **rrcov** (Todorov and Filzmoser, 2009) and **rrcovHD**. However, it is important to note that all the implemented functions do not necessary return outlierness measures and the identities of the outlying observations. Limiting only to some packages fulfilling these conditions and to multivariate methods for numerical variables leads to focusing on **mvoutlier**, **CerioliOutlierDetection** and **rrcov**. These packages implement modified versions of the robust Mahalanobis distance, different algorithms for the outlier identification in high dimensions (Filzmoser and Todorov, 2013), classical and robust PCA. In PCA, the labelling of the outliers is based on the comparison of two outlierness measures: a score distance (SD) based on the first principal components and a distance in the orthogonal space (OD). All these methods rely on distances. To extend the analysis, we include comparisons with Local Outlier Factor (LOF) (**Rlof** package) and Angle-Based Outlier Factor (ABOD) (**abodOutlier** package) approaches even if they only return outlierness measures and not the identities of the outlying observations.

These methods dedicated to outlier detection in an unsupervised context have different properties. Contrary to PCA, the Mahalanobis distance and its modified versions are invariant under any affine transformation as long as the location and scatter estimators are affine equivariant. However, several drawbacks have been noticed in the use of these distances. First, Filzmoser et al. (2005) highlight the fact that an outlier is not necessarily an extreme value. So, using a fixed threshold to label outliers for every data set is not the wisest solution and the threshold should be adjusted to the sample size. Moroever, as Cerioli (2010) notes, the threshold used usually is not adapted to the case in which there is no outlier in the data. Finally, for Cerioli (2010), the comparison of the distances of each observation to the threshold should not be performed independently without taking into account any simultaneous adjustments. Otherwise, the method may lead to many false positives,

i.e non outlying observations that are identified as outliers. Note that this is a serious concern that applies to almost all the outlier-detection methods. To the best of our knowledge, the **mvoutlier** and **CerioliOutlierDetection** packages are the first ones that tend to correct these drawbacks, but only for the robust Mahalanobis distance with the Minimum Covariance Determinant (MCD) estimators.

The **ICSOutlier** (Archimbaud et al., 2016) package is another approach to the detection of outliers. It appears that ICS is as efficient as its competitors in many situations and shows an advantage in the case of a small proportion of outliers lying in a low-dimensional subspace. This method is well-designed in particular for quality control. Its current version is able to handle a small proportion of outliers that lie on a subspace, using the affine equivariant Invariant Coordinate Selection (ICS) method, as presented by Archimbaud et al. (2018). The method relies on a generalized diagonalization and on the selection of the invariant components associated with the largest eigenvalues. Note that when more than say 10% of the observations are suspected to be outlying, invariant components associated with the smallest eigenvalues may also be of interest and this alternative will be considered in a future version of the package. ICS fulfills all the properties mentioned previously: (i) detect the absence of outliers, (ii) not be sensitive to the standardization of the data and (iii) be a multivariate method which controls the number of observations identified as outliers. The restriction to a small proportion of outliers that belong to a subspace is of interest in some areas of manufacturing products with a high level of quality control (e.g. automotive, avionics or aerospace).

In the following sections, the principle of the Invariant Coordinate Selection (ICS) method is recalled as well as the three steps of the outlier detection procedure: (i) invariant components selection, (ii) outlierness measure definition and (iii) outlier identification. Then, we explain how to use the **ICSOutlier** package and finally we analyze the efficiency of the ICS method compared to other methods on four examples, including a new real data set included in the package.

Invariant Coordinate Selection (ICS) for outlier detection

Principle

The ICS method is a powerful method designed for exploring multivariate data by revealing their structure (Nordhausen et al., 2008). As explained in Tyler et al. (2009), it is based on a simultaneous spectral decomposition of two scatter matrices and leads to an affine invariant coordinate system. Readers not so familiar with the concept of scatter matrices are refered for example to Rousseeuw and Hubert (2013), Nordhausen and Tyler (2015) and references therein for definitions, examples and properties.

Following Archimbaud et al. (2018), for a p-variate dataset $\mathbf{X}_n = (\mathbf{x}_1, \ldots, \mathbf{x}_n)'$, let $\mathbf{V}_1(\mathbf{X}_n)$ and $\mathbf{V}_2(\mathbf{X}_n)$ be two affine equivariant scatter matrices (symmetric and positive definite) and $\mathbf{m}_1(\mathbf{X}_n)$ and $\mathbf{m}_2(\mathbf{X}_n)$ their associated location estimators. ICS is looking for the diagonal $p \times p$ matrix $\mathbf{D}(\mathbf{X}_n)$ containing the eigenvalues of $\mathbf{V}_1(\mathbf{X}_n)^{-1}\mathbf{V}_2(\mathbf{X}_n)$ in decreasing order and the $p \times p$ matrix $\mathbf{B}(\mathbf{X}_n) = (\mathbf{b}_1, \ldots, \mathbf{b}_p)'$ containing the corresponding eigenvectors as its rows and such that

$$\mathbf{B}(\mathbf{X}_n)\mathbf{V}_1(\mathbf{X}_n)\mathbf{B}'(\mathbf{X}_n) = \mathbf{I}_p.$$

We have:

$$\mathbf{V}_1(\mathbf{X}_n)^{-1}\mathbf{V}_2(\mathbf{X}_n)\mathbf{B}(\mathbf{X}_n)' = \mathbf{B}(\mathbf{X}_n)'\mathbf{D}(\mathbf{X}_n).$$

Letting $\mathbf{V}_1(\mathbf{X}_n)$ be "more robust" than $\mathbf{V}_2(\mathbf{X}_n)$, the interpretation of these eigenvalues is the main point of the ICS method: they correspond to some kurtosis measure that depends on the choice of $\mathbf{V}_1(\mathbf{X}_n)$ and $\mathbf{V}_2(\mathbf{X}_n)$. For example if $\mathbf{V}_1$ is the usual empirical variance-covariance matrix and $\mathbf{V}_2$ is the scatter matrix based on the fourth moments (the default in the `ics.outlier` function), the eigenvalues are ordered according to their classical Pearson kurtosis values in decreasing order. In this case and for a small proportion of outliers, it is advisable to analyze the projections that maximize the kurtosis and are associated with the largest eigenvalues (Archimbaud et al., 2018).

Then, using the location estimator $\mathbf{m}_1(\mathbf{X}_n)$ associated with the scatter matrix $\mathbf{V}_1(\mathbf{X}_n)$, the corresponding centered scores are obtained as $\mathbf{z}_i = \mathbf{B}(\mathbf{X}_n)(\mathbf{x}_i - \mathbf{m}_1(\mathbf{X}_n))$ for $i = 1, \ldots, n$, so that:

$$\mathbf{Z}_n = (\mathbf{z}_1, \ldots, \mathbf{z}_n)' = (\mathbf{X}_n - \mathbf{1}_n\mathbf{m}_1'(\mathbf{X}_n))\mathbf{B}'(\mathbf{X}_n).$$

They are called the invariant components (IC) because of their affine invariance property in the sense that if $\mathbf{X}_n^* = \mathbf{X}_n\mathbf{A} + \mathbf{1}_n\mathbf{b}'$ for any $p \times p$ regular matrix $\mathbf{A}$ and any p-vector $\mathbf{b}$,

$$(\mathbf{X}_n^* - \mathbf{1}_n\mathbf{m}_1(\mathbf{X}_n^*)')\mathbf{B}(\mathbf{X}_n^*)' = (\mathbf{X}_n - \mathbf{1}_n\mathbf{m}_1(\mathbf{X}_n)')\mathbf{B}(\mathbf{X}_n)'\mathbf{J},$$

where $\mathbf{1}_n$ is a n-vector of ones and $\mathbf{J}$ is a $p \times p$ diagonal matrix with diagonal elements ± 1, which

means the invariant coordinates change at most their signs.

These scores are related to the Mahalanobis Distance (MD). For any observation $\mathbf{x}_i$ with $i = 1, \ldots, n$, the squared Euclidian norm of its centered invariant coordinates is exactly the squared Mahalanobis distance from $\mathbf{m}_1(\mathbf{X}_n)$ in the sense of $\mathbf{V}_1(\mathbf{X}_n)$:

$$\mathbf{z}_i'\mathbf{z}_i = (\mathbf{x}_i - \mathbf{m}_1(\mathbf{X}_n))'\mathbf{V}_1(\mathbf{X}_n)^{-1}(\mathbf{x}_i - \mathbf{m}_1(\mathbf{X}_n)).$$

The added value of using ICS over MD is realized when the structure of the data is on a subspace of dimension $q < p$. In this context and with a small percentage of outliers, it is of interest to focus only on the q projections that maximize a kurtosis measure. Indeed, if the number k of selected Invariant Coordinates corresponds to the dimension q of the subspace on which the outliers lie, then the ICS method is expected to recover the subspace of interest for outlier detection. As detailed in Tyler et al. (2009), this subspace corresponds to Fisher's discriminant subspace in case of mixtures of Gaussian distributions. Nevertheless, the main difficulty is to correctly estimate this dimension k.

Invariant coordinates selection

Depending on the combination of the scatters $\mathbf{V}_1(\mathbf{X}_n)$ and $\mathbf{V}_2(\mathbf{X}_n)$ chosen, the **ICSOutlier** package incorporates automated ways to select these k invariant coordinates. The two approaches proposed in Archimbaud et al. (2018) are implemented here: a test based on a quasi-inferential procedure and some normality tests. Considering $\mathbf{V}_1(\mathbf{X}_n)$ be more robust than $\mathbf{V}_2(\mathbf{X}_n)$ and a small percentage of outliers (less than 10% is recommended), the structure of the outlierness should be contained in the first k non-normal components. Note that the structure may be also contained in the last components (associated with the smallest eigenvalues) if the proportion of outliers is large. What is meant by "large" depends on the scatter pair but also on the distribution of the data as detailed in Archimbaud et al. (2018). Since the invariant coordinates are already ordered decreasingly according to a kurtosis measure, it is enough to test whether each component is Gaussian beginning by the one associated with the largest eigenvalue and stop the test procedure as soon as we find a Gaussian component. So, if $k+1$ denotes the rank of the first Gaussian component, the components are sequentially tested at the adapted level $\alpha_j = \alpha/j$, for $j = 1, \ldots, k$, as in Dray (2008). Because of the sequentiality, the Bonferroni correction adjusts the significance of each test and ensures a nominal level α.

The first approach is a Parallel Analysis (PA) based on Monte Carlo simulations of eigenvalues for Gaussian populations of the same dimension as the initial dataset. This method is common for selecting components in PCA as described in Peres-Neto et al. (2005). It is based on $mEig$ simulations of a Gaussian population and, for each $j = 1, \ldots, p$, the computation of the $1 - \alpha_j$ percentile of the j^{th} eigenvalue of ICS which is considered as a cut-off for the j^{th} component. Then, sequentially from $j = 1$, if the observed j^{th} eigenvalue exceeds the cut-off then the j^{th} invariant component is declared non-Gaussian and is selected. As soon as one eigenvalue is smaller than its associated cut-off, the invariant component is considered as Gaussian and the test procedure is stopped.

The second approach is directly based on usual normality tests and is applied to the invariant coordinates. In the package the user can choose out of the following five tests: the D'Agostino test of skewness (DA), the Anscombe-Glynn (AG) test of kurtosis, the Bonett-Seier (BS) test of Geary's kurtosis, the Jarque-Bera (JB) test for normality which is based on both skewness and kurtosis measures and the Shapiro-Wilk (SW) normality test (see Yazici and Yolacan (2007) and Bonett and Seier (2002) for a complete description of each). The process of testing is still sequential with an adaptation of the level of each test. Once the first normal coordinate is found, the test procedure stops and only the non-Gaussian coordinates are selected.

Measure of outlierness

Let k denote the number of invariant components selected by one of the two test procedures detailed previously or by looking at the screeplot of the eigenvalues. Then, for each observation $\mathbf{x}_i$, $i = 1, \ldots, n$, its squared ICS distance is computed based on the k selected invariant coordinates by:

$$ICSD^2_{\mathbf{V}_1(\mathbf{X}_n)^{-1}\mathbf{V}_2(\mathbf{X}_n)}(\mathbf{x}_i, k) = \mathbf{z}_{i,k}'\mathbf{z}_{i,k}.$$

where $\mathbf{z}_{i,k} = \mathbf{B}(\mathbf{X}_n, k)(\mathbf{x}_i - \mathbf{m}_1(\mathbf{X}_n))$, for $i = 1, \ldots, n$, and the $k \times p$ matrix $\mathbf{B}(\mathbf{X}_n, k)$ contains the k first rows of $\mathbf{B}(\mathbf{X}_n)$.

Outlier identification

Finally, the outliers are identified based on the comparison of their ICS distance with the expectation under the Gaussian distribution as in Archimbaud et al. (2018). The cut-off is derived from $mDist$ Monte Carlo simulations of a Gaussian population of the same dimension as the initial dataset. For each observation, the ICS distance is computed using the k selected components. For a given level β, the cut-off corresponds to the average of the $1-\beta$ percentiles of the distances over all the simulations. An observation is labeled as outlier if its ICS distance is higher than this cut-off. By default, $\beta = 5\%$.

Using the package ICSOutlier

The main function available in **ICSOutlier** is `ics.outlier` which directly calls all the other functions included in the package. First it is necessary to apply the `ics2` function from the **ICS** (Nordhausen et al., 2008) package in order to create an object of class `ics2`. The **ICS** package has been available for a few years, and contains a function called `ics` for implementing the ICS method. However, the `ics` function cannot be used directly for outlier detection, and a new function, `ics2`, had to be added to the **ICS** package (version 1.3-0). The arguments of `ics2` are mostly the same as those of the function `ics`, except for now it is necessary to add the location vectors associated with the scatter estimators. The main function `ics.outlier` of the package **ICSOutlier** uses the output of the `ics2` function as an input. The function `ics.outlier` implements the three steps necessary for outlier detection using ICS in an automated way:

(i) Select the invariant coordinates which recover at best the subspace where the outliers are lying using some test procedure.

(ii) Compute the ICS distance based on the selected invariant coordinates as an outlierness measure for each observation.

(iii) Label the outliers using the cut-off value obtained through simulations.

The links between the `ics2` function from the **ICS** package and the functions available in the **ICSOutlier** package are illustrated on Figure 1.

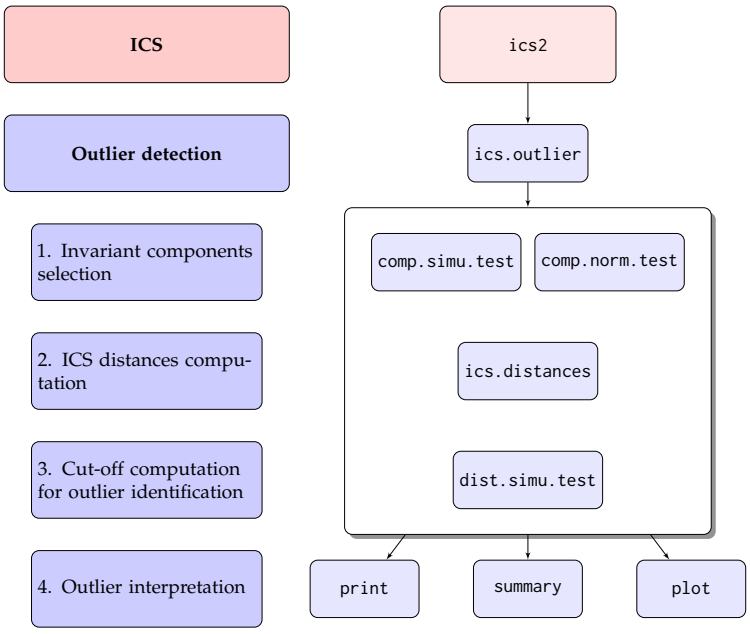


Figure 1: Organization chart of the functions called by `ics.outlier`

The `ics.outlier` function only takes as parameter an object of class `ics2` and not an object of class `ics` because this later returns only uncentered invariant coordinates contrary to the `ics2` class. From a practical point a view, the `S1` and `S2` arguments of the `ics2` function should be the name of the function which returns a list with the first (resp. second) location vector $\mathbf{m}_1$ (resp. $\mathbf{m}_2$) and the first scatter matrix $\mathbf{V}_1$ (resp. $\mathbf{V}_2$).

In addition to this ics2 object, there are other parameters of ics.outlier that can be tuned, including an option to parallelize the computations in order to increase the speed of the function:

- method: name of the method used to select the ICS components. Options are "simulation" for the parallel analysis approach and "norm.test" for the normality tests approach. Depending on the method, either comp.norm.test or comp.simu.test is used.
- test: name of the marginal normality test to use if method = "norm.test". Possibilities are "jarque.test", "anscombe.test", "bonett.test", "agostino.test", "shapiro.test". Default is "agostino.test".
- mEig: number of simulations performed to derive the cut-off values for selecting the ICS components. Only if method = "simulation". See comp.simu.test for details.
- level.test: level for the comp.norm.test or comp.simu.test functions. The inital level for selecting the invariant coordinates.
- adjust: logical. For selecting the invariant coordinates, the level of the test can be adjusted for each component to deal with multiple testing. See comp.norm.test and comp.simu.test for details. Default is TRUE.
- level.dist: level for the dist.simu.test function. The (1-level(s))th quantile(s) used to determine the cut-off value(s) for the ICS distances.
- mDist: number of simulations performed to derive the cut-off value for the ICS distances. See dist.simu.test for details.
- type: currently the only option is "smallprop" which means that only the first ICS components can be selected. See comp.norm.test or comp.simu.test for details.
- ncores: number of cores to be used in dist.simu.test and comp.simu.test. If NULL or 1, no parallel computing is used. Otherwise makeCluster with type = "PSOCK", from the **parallel** package, is used.
- iseed: The seed passed on to clusterSetRNGStream if parallel computation is used. Default is NULL which means no fixed seed is used.
- pkg: A character vector listing all the packages which need to be loaded on the different cores via require if parallel computation is used. Must be at least "ICSOutlier" and must contain the packages needed to compute the scatter matrices.
- qtype: specifies the quantile algorithm used in quantile.

By default the ics2 function performs ICS with the usual mean vector and the usual empirical variance-covariance matrix returned by MeanCov and the location vector based on the third moments associated to the scatter matrix based on the fourth moments returned by Mean3Cov4.

Examples

In the following examples, we illustrate first, using an artifical data set, how ics.outlier behaves in the case of no outlier and then apply the function to three data sets available in R. For reproducibility all examples have a fixed seed.

Example with no outlier

One of the advantages of using ICS for outlier detection is its ability to detect the absence of outliers. If the first invariant coordinate is considered normal that means there is no outlier in the data set. So, in presence of a normal multivariate data set the function should select in most cases no component and hence will not detect outliers.

For the first example we simulate a bivariate normally distributed data set with 500 observations and we determine the cut-offs for identifying the outliers at the level 2.5% for all methods.

```
library("ICSOutlier", quietly = TRUE)
# Data simulation
set.seed(123)
X <- matrix(rnorm(1000, 0, 0.1), 500, 2)
```

Using the default ICS setting and choosing the Jarque-Bera test of kurtosis to select the components at the default level 5% can easily be done as follows.

```
icsX <- ics2(X)
icsOutlierJB <- ics.outlier(icsX, test = "jarque", level.test = 0.05,
                            level.dist = 0.025)
print(icsOutlierJB)

R> [1] "0 components were selected and no outliers were detected."
```

Hence in this outlier-free normal data no component was considered non-normal and therefore as desired no outliers were detected.

As a comparison we compute the robust Mahalanobis distances using the MCD with a breakdown point of 25% and compute the commonly used cut-off value coming from the χ^2_2 distribution and the adjusted cut-off value from (Green and Martin, 2017b) as implemented in the package **CerioliOutlierDetection**.

```
# Robust Mahalanobis distance with MCD estimates with a breakdown point of 25%
library("robustbase")
MCD <-  covMcd(X, alpha = 0.75)
RD <- mahalanobis(X,  MCD$center, MCD$cov)

# Cut-off based on the chi-square distribution
cutoff.chi.sq <- qchisq(0.975, df = ncol(X))
cutoff.chi.sq

R> [1] 7.377759

# Cut-off based Green and Martin (2017)
library("CerioliOutlierDetection")
cutoff.GM <- hr05CutoffMvnormal(n.obs = nrow(X), p.dim = ncol(X), mcd.alpha = 0.75,
                                signif.alpha = 0.025, method = "GM14",
                                use.consistency.correction = TRUE)$cutoff.asy
cutoff.GM

R> [1] 14.22071
```

To visualize these results, the robust squared Mahalanobis distances are plotted with two different cut-off values in Figure 2.

```
# Code for the Figure 2
colPoints <- ifelse(RD >= min(c(cutoff.chi.sq, cutoff.GM)), 1, grey(0.5))
pchPoints <- ifelse(RD >= min(c(cutoff.chi.sq, cutoff.GM)), 16, 4)
plot(seq_along(RD), RD,  pch = pchPoints, col = colPoints,
     ylim=c(0, max(RD, cutoff.chi.sq, cutoff.GM) + 2), cex.axis = 0.7, cex.lab = 0.7,
     ylab = expression(RD**2), xlab = "Observation Number")
abline(h = c(cutoff.chi.sq, cutoff.GM), lty = c("dashed", "dotted"))
legend("topleft", lty = c("dashed", "dotted"), cex = 0.7, ncol = 2, bty = "n",
       legend = c(expression(paste(chi[p]**2, " cut-off")), "GM cut-off"))
```

Hence in this example using the χ^2_2 cut-off yields 12 outliers while the more sophisticated cut-off does not classify any observation as outlying.

Another situation where the `ics.outlier` function can conclude to an absence of outliers is when the squared ICS distances are below the corresponding cut-off for all observations, as illustrated in the help file of the function: `?ics.outlier`.

HTP data set

The real data set `HTP`, introduced in Archimbaud et al. (2018), is included in this package. The data set provides the results of 88 numerical tests for 902 high-tech parts. Based on these results the producer considered all parts functional and all of them were sold. However two parts, 581 and 619, showed defects in use and were returned to the manufacturer. These two observations can thus be considered as outliers and the objective is to detect them. We use the `ics.outlier` function with its default settings, as most users initially would do but with the parallelizing option for the computations.

```
# HTP dataset
library("ICSOutlier")
set.seed(123)
```

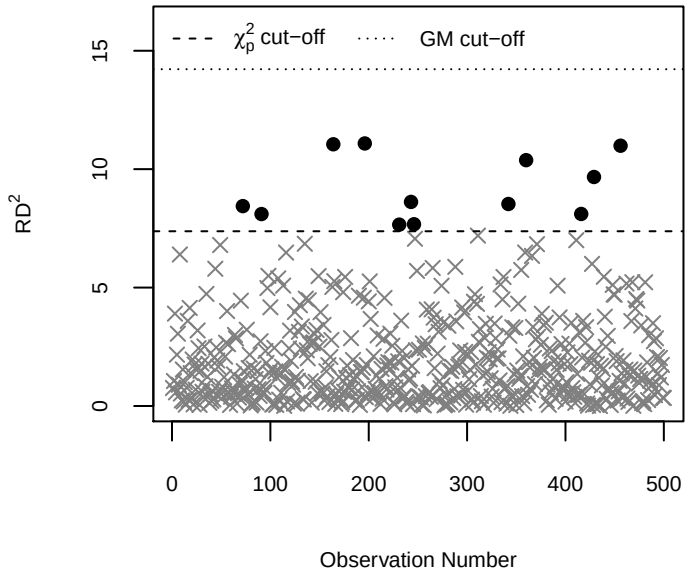


Figure 2: Squared robust Mahalanobis distances and two different cut-off values.

```
data(HTP)
outliers <- c(581, 619)

# default ICS
icsHTP <- ics2(HTP)

# Outlier detection with selection of components based on normality tests
# by default it can take quite long as mDist = 10000 so we choose to
# use all but one available cores to parallelize the simulations.
library(parallel)
icsOutlierDA <- ics.outlier(icsHTP, ncores = detectCores()-1, iseed = 123)
summary(icsOutlierDA)

R>
R> ICS based on two scatter matrices and two location estimates
R> S1:  MeanCov
R> S2:  Mean3Cov4
R>
R> Searching for a small proportion of outliers
R>
R> Components selected at nominal level 0.05: 14
R> Selection method: norm.test (agostino.test)
R> Number of outliers at nominal level 0.025: 43

# Code for the Figure 3
plot(icsOutlierDA, cex.lab = 0.7, cex.axis = 0.7)
points(outliers, icsOutlierDA@ics.distances[outliers], pch = 5)
text(outliers, icsOutlierDA@ics.distances[outliers], outliers, pos = 2, cex = 0.7)
```

Based on this result we are able to identify the two outliers among the 5% of the observations declared as outliers, taking into account the 14 first components selected by the D'Agostino normality test.

However, following Archimbaud et al. (2018), a simple alternative for selecting components is to use the screeplot of the `icsHTP` object, in a similar way as for PCA. Note that while the eigenvalues

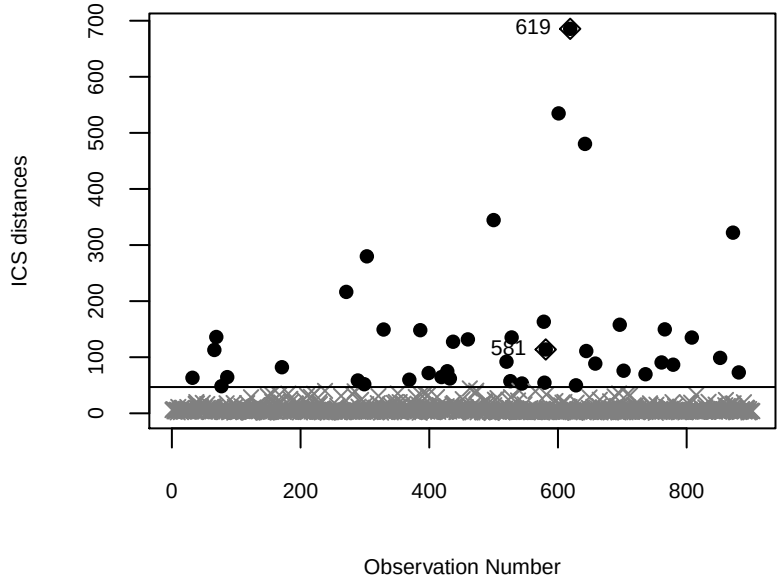


Figure 3: Squared ICS distances for HTP data with default parameters.

plotted on the screeplot are related to the variance, for PCA they correspond to a generalized kurtosis for ICS (see Tyler et al. (2009)).

```
# Code for the Figure 4
screeplot(icsHTP, cex.lab = 0.7, cex.axis = 0.7, cex.names = 0.7, cex.main = 0.7)
```

Based on this screeplot, three components might be a reasonable choice and then the functions `ics.distances` and `dist.simu.test` can be used to obtain the desired distances and cut-off value.

```
ics.dist.scree <- ics.distances(icsHTP, index = 1:3)
# by default it can take quite long as m = 10000, so we choose to
# use all but one available cores to parallelize the simulations.
library(parallel)
ics.cutOff <- dist.simu.test(icsHTP, 1:3, ncores = detectCores()-1, iseed = 123)
ics.cutOff

R>    97.5%
R> 12.80771
```

Before visualizing these results we also apply two competing outlier detection methods. First the Finite-Sample Reweighted MCD (FSRMCD) outlier detection test of Cerioli (Cerioli, 2010), implemented in the package **CerioliOutlierDetection** and secondly the SIGN1 algorithm (Filzmoser et al., 2008) implemented in the package **mvoutlier**. For the FSRMCD algorithm, we choose as nominal level $\alpha = 1 - \gamma^{1/n}$ for the individual outlier tests, with γ the nominal size of the intersection test and n the number of observations. Then the ICS distances based on the three selected components are plotted against the distances from the two competing methods together with the corresponding cut-off values.

```
# FSRMCD
library("CerioliOutlierDetection")
FSRMCD <- cerioli2010.fsrmcd.test(HTP, signif.alpha = 1 - 0.975**(1/nrow(HTP)),
                                  mcd.alpha = 0.75)
# Two critical values: one for points included in the reweighted MCD (weights == 1)
# and one for points excluded from the reweighted MCD (weights == 0)).
FSRMCD.cutoffs <- unique(FSRMCD$critvalfcn(FSRMCD$signif.alpha))
FSRMCD.cutoffs
```

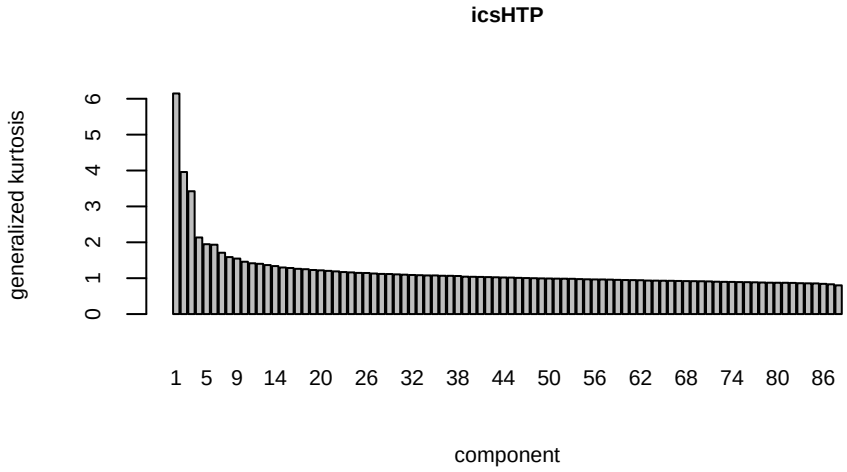


Figure 4: Screeplot of ICS eigenvalues for HTP data and default parameters.

```
R> [1] 144.8583 183.2797

# SIGN1
library("mvoutlier", quietly = TRUE)
SIGN1 <- sign1(HTP, qcrit = 0.975)

# Code for the Figure 5
par(mfrow = c(1, 2))
par(mar = c(4, 4, 2, 0.2))

# Comparison ICS vs FSRMCD
colPoints <- ifelse(ics.dist.scree >= ics.cutOff , 1, grey(0.5))
pchPoints <- ifelse(ics.dist.scree >= ics.cutOff, 16, 4)
plot(FSRMCD$mahdist.rw, ics.dist.scree, col = colPoints, pch = pchPoints,
     cex.lab = 0.7, cex.axis = 0.7, cex.main = 0.7,
     main = "ICS vs FSRMCD", ylab = "ICS Distances", xlab = "FSRMCD")
points(FSRMCD$mahdist.rw[outliers], ics.dist.scree[outliers], pch = 5)
text(FSRMCD$mahdist.rw[outliers], ics.dist.scree[outliers], labels = outliers, pos = 2,
     cex = 0.7)
abline(h = ics.cutOff, v = FSRMCD.cutoffs, lty = "dashed")

# Comparison ICS vs SIGN1
plot(SIGN1$x.dist, ics.dist.scree,  col = colPoints, pch = pchPoints,
     cex.lab = 0.7, cex.axis = 0.7, cex.main = 0.7,
     main = "ICS vs SIGN1", ylab = "ICS Distances", xlab = "SIGN1")
points(SIGN1$x.dist[outliers], ics.dist.scree[outliers], pch = 5)
text(SIGN1$x.dist[outliers], ics.dist.scree[outliers], labels = outliers, pos = 2,
     cex = 0.7)
abline(h = ics.cutOff, v = SIGN1$const, lty = "dashed")
```

As illustrated in Figure 5, the cut-offs used for the FSRMCD and SIGN1 algorithms plotted on the x-axis lead to too many false positives. Even when focusing only on the rank of the outlying observations it is clear that the ICS method outperforms the other methods. The outlier labelled 619 is obviously outlying with all methods, but the outlier labelled 581 has the 4th highest squared ICS distance compared to 15th and 12th for the distances computed with FSRMCD and SIGN1 algorithms. Note that the data set contains highly collinear variables and it is well-known that high-breakdown point methods are not optimal in this case.

Archimbaud et al. (2018) contains a more detailed analysis of this dataset comparing ICS with other methods and also gives the impact of the choice of different scatter matrices. For the present data set the default scatter combination used here seems the best. In this analysis, it is also interesting to notice that the first component highlights only the two defective parts. The other two observations are

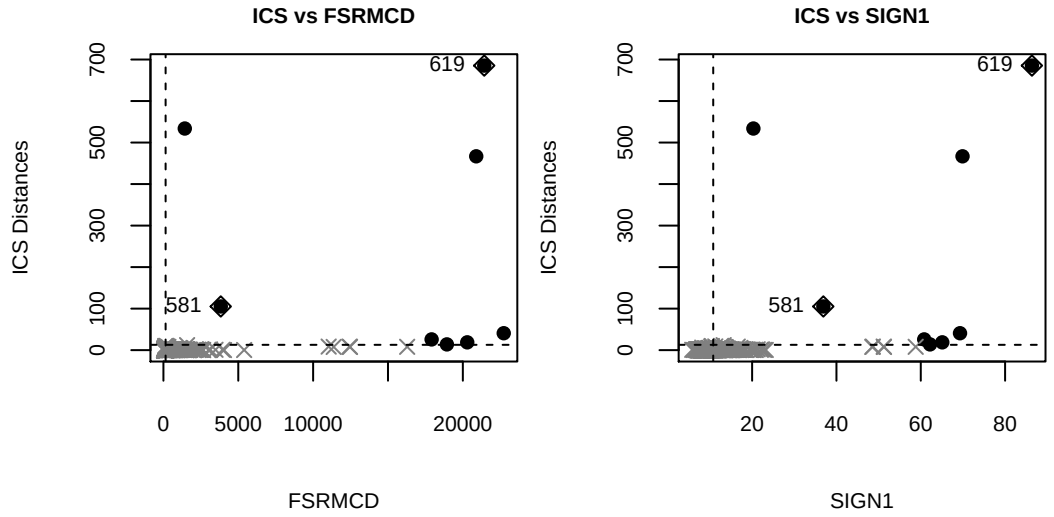


Figure 5: Squared ICS distances with 3 components against FSRMCD (left panel) and SIGN1 (right panel) measures of outlierness for HTP data with cut-offs at level of 2.5%.

detected on the second and the third components of ICS and as far as we know they are not defective parts but false positives.

Reliability data set

The Reliability data set comes also from an industrial context and is part the of **REPPlab** package. It contains 55 variables measured during the production process of 520 units. The challenge for this data is to identify the produced items with a fault not detected by the marginal tests. According to Fischer et al. (2016a), the observations numbered 414 and 512 are most likely outliers.

A special feature of this data set is that for example variable 24 is nearly constant which makes this data set hard to use with high-breakdown methods. For example the MCD cannot be computed for this data set when all variables are used. The use of ICS and other distance-based outlier detection methods is also detailed in Archimbaud et al. (2018).

Let us now illustrate the use of other scatter matrices than the default one together with the way to pass arguments to some of the functions. This time the invariant components are selected using simulations (Parallel analysis) with an initial decision level of 5%. These results are then compared to two methods not included in Archimbaud et al. (2018), namely the Local Outlier Factor (LOF, (Breunig et al., 2000)) and the Angle-Based Outlier Factor (Kriegel et al., 2008) outlier detection methods. First, we apply the LOF method with the `lof` function from the **Rlof** package which parallelizes the computation of the local outlier factor using l neighbours for each observation. The number of neighbours is determined based on the guidelines from Breunig et al. (2000): we compute the outlierness measures for l between 5 and 50 and we aggregate the results by taking the maximum of the local outlier factor for each observation. However, the distribution of the LOF values (and of the aggregated indices) is not known and therefore, there exists no theoretical cut-off to identify the outlying observations. In practice, observations with LOF values much higher than one are considered as potentially outlying. Second, we calculate the angle-based outlier factor for each observation through the `abod` function from the **abodOutlier** package on a random sample of 10% (the default) of the data (not on the entire data set because it is too time-consuming). As for the LOF method, the distribution of the outlyingness indices is unknown. But for this method, a potentially outlying observation is indicated by a small index value.

The location and scatter combinations we use here are the regular mean vector and covariance matrix and the joint maximum likelihood estimation of location and scatter of a t-distribution with one degree of freedom, also known as Cauchy MLE estimate. The function `MeanCov` returns the mean vector and regular covariance matrix as required for `ics2` and for the Cauchy MLE we use the function `tM` where the degree of freedom is specified using the argument `df`. As the Cauchy MLE is considered the more robust estimator it should be specified in `ics2` as `S1`.

```
# ReliabilityData example: the observations 414 and 512 are suspected to be outliers
library("ICSOutlier")
library("REPPlab")
```

```
set.seed(123)
data(ReliabilityData)
outliers <- c(414, 512)

# ICS with MLE Cauchy and the Mean-Cov
icsReliabilityData <- ics2(ReliabilityData, S1 = tM, S2 = MeanCov,
                           S1args = list(df = 1))

# Outlier detection with selection of components based on simulations
# it can take quite long as mEig = 5000 and mDist = 5000, so we choose
# to use all but one available cores to parallelize the simulations.
icsOutlierPA <- ics.outlier(icsReliabilityData, method = "simulation",
                            level.test = 0.05, mEig = 5000,
                            level.dist = 0.01, mDist = 5000,
                            ncores = detectCores()-1, iseed = 123)
icsOutlierPA

R> [1] "39 components were selected and 86 outliers were detected."

# LOF: Local Outlier Factor
library("Rlof", quietly = TRUE)
X.lof <- lof(ReliabilityData, 5:50, cores = 2)
X.lof.max <- apply(X.lof, 1, max)

# ABOD: Angle-Based Outlier Factor
library("abodOutlier", quietly = TRUE)
X.abod <- abod(ReliabilityData, method = "randomized")

# Code for the Figure 6
par(mfrow = c(1, 2))
par(mar = c(4, 4, 2, 0.2))

# Comparison ICS vs LOF
plot(X.lof.max, icsOutlierPA@ics.distances, cex.lab = 0.7, cex.axis = 0.7, pch = 4,
     cex.main = 0.7, main = "ICS vs LOF", ylab = "ICS Distances", xlab = "LOF")
text(X.lof.max[outliers], icsOutlierPA@ics.distances[outliers], labels = outliers,
     pos = 4, cex = 0.7)

# Comparison ICS vs ABOD
plot(X.abod, icsOutlierPA@ics.distances, cex.lab = 0.7, cex.axis = 0.7, cex.main = 0.7,
     main = "ICS vs ABOD", ylab = "ICS Distances", xlab = "ABOD", pch = 4)
text(X.abod[outliers], icsOutlierPA@ics.distances[outliers], labels = outliers,
     pos = 4, cex = 0.7)
```

This analysis shows that ICS selects here quite many components and suggests 86 outliers at the level of 1%. For their part, the LOF and ABOD methods do not provide cut-offs for identifying outlying observations. So we only consider the ranks of the observations. When looking at the ICS distances on Figure 6, the observations 414 and 512 are clearly separated from the main bulk of the data while the two observations do not differ at all from the rest of the data using the LOF and ABOD methods (see also Archimbaud et al. (2018) for more details on this exmaple).

HBK data set

The `hbk` data set is included in the **robustbase** package (Rousseeuw et al., 2017). It is an artificial data set created by Hawkins et al. (1984) for illustrating the so-called masking effect of outliers. It contains two groups of outliers with observations 1-10 in the first group and observations 11-14 in the second group, over the 75 observations characterized by the three explanatory variables. Usually with non-robust methods only the observations 12, 13 and 14 are identified as outliers. For this example the percentage of outliers is around 19% and so is larger than the percentage we recommend for the current version of the package. Our aim is to illustrate that in such a situation, the choice of the scatter matrices may have a large impact on the results.

First, the combination of the highly robust MCD estimates with the mean vector and regular covariance matrix is studied. Note that in order to be able to use the MCD via the function `CovMcd` a wrapper needs to be written around the function so that it returns the proper list as expected by `ics2`. Using then the D'Agostino test for skewness for selecting the invariant components leads to select two

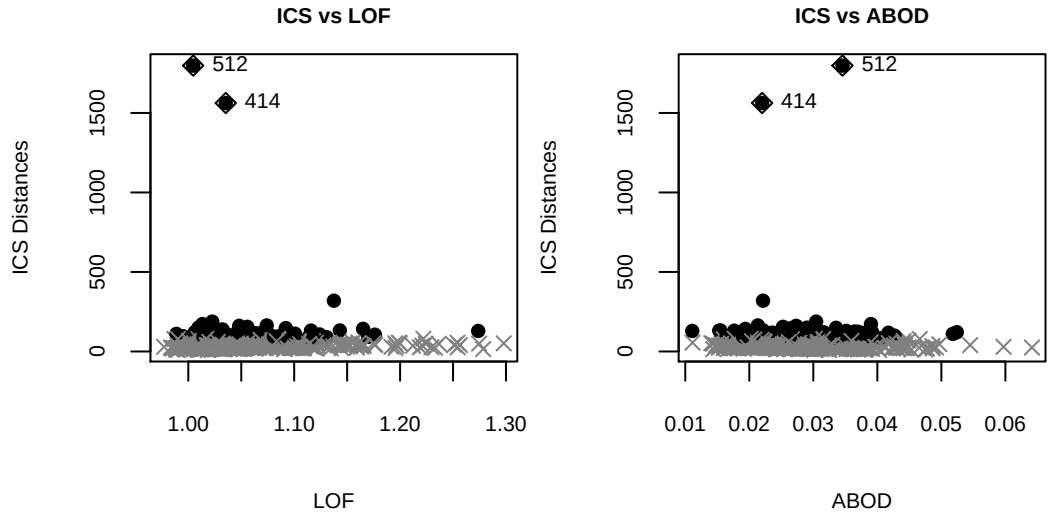


Figure 6: Squared ICS distances with 39 components against LOF (left panel) and ABOD (right panel) measures of outlierness for Reliability data.

components at the default level of 5%. The level of the quantile used for deriving the cut-off for outlier identification is the default, 2.5%. For this combination, ICS performs equally well as the MCD-based Mahalanobis distances, as illustrated in the first three plots of the Figure 7. All outliers are correctly identified, no false positive is detected and the two groups of outliers are clearly separated.

Next, we consider the combination of two non-robust location vectors and scatter matrices which are the default in ICS, with the same parameters for selecting the components and for identifying the outliers as previously. The results are presented on the fourth plot of Figure 7.

```
library("rrcov")
set.seed(123)
# HBK data set
data(hbk)

# ICS with MCD estimates and the usual estimates
# Need to create a wrapper for the CovMcd function to return first the location estimate
# and the scatter estimate secondly.
myMCD <- function(x,...){
  mcd <- CovMcd(x,...)
  return(list(location = mcd@center, scatter = mcd@cov))
}
icsHBK_mcd <- ics2(hbk[, 1:3], S1 = myMCD, S2 = MeanCov, S1args = list(alpha = 0.75))

# Outlier detection with selection of components based on the D'Agostino test for
# skewness. It can take quite long as mEig = 10000 and mDist = 10000, so we choose
# to use all but one available cores to parallelize the simulations.
library(parallel)
icsOutlierDA.MCD <- ics.outlier(icsHBK_mcd, mEig = 10000, level.dist = 0.025,
                                mDist = 10000,
                                ncores = detectCores()-1, iseed = 123,
                                pkg = c("ICSOutlier", "rrcov"))
icsOutlierDA.MCD

R> [1] "2 components were selected and 14 outliers were detected."

# Robust Mahalanobis distance with MCD estimates with a breakdown point of 25%
MCD <-  covMcd(hbk[, 1:3], alpha = 0.75)
RD <- mahalanobis(hbk[, 1:3],  MCD$center, MCD$cov)
cutoff.chi.sq <- qchisq(0.975, df = ncol(hbk[, 1:3]))

# Cut-off based Green and Martin (2017)
library("CerioliOutlierDetection")
cutoff.GM <- hr05CutoffMvnormal(n.obs = nrow(hbk[, 1:3]), p.dim = ncol(hbk[, 1:3]),
```

```
                                mcd.alpha = 0.75, signif.alpha = 0.025, method = "GM14",
                                use.consistency.correction = TRUE)$cutoff.asy

# ICS with non robust estimates
icsHBK <- ics2(hbk[,1:3], S1 = MeanCov, S2 = Mean3Cov4)

# Outlier detection with selection of components based on the D'Agostino test for
# skewness. It can take quite long as mEig = 10000 and mDist = 10000, so we choose
# to use all but one available cores to parallelize the simulations.
icsOutlierDA <- ics.outlier(icsHBK, mEig = 10000, level.dist = 0.025,
                                mDist = 10000,
                                ncores = detectCores()-1, iseed = 123,
                                pkg = c("ICSOutlier", "rrcov"))
icsOutlierDA

R> [1] "2 components were selected and 2 outliers were detected."

# Code for the Figure 7
par(mfrow = c(1, 3))
par(mar = c(4, 2, 2, 0.2))

# Robust Mahalanobis distance with MCD estimates
colPoints <- ifelse(RD >= cutoff.chi.sq, 1, grey(0.5))
pchPoints <- ifelse(RD >= cutoff.chi.sq, 16, 4)
plot(RD, col = colPoints, pch = pchPoints, xlab = "Observation Number",
     cex.lab = 0.7, cex.axis = 0.7, main = "Robust MD", cex.main = 0.7)
abline(h = c(cutoff.chi.sq, cutoff.GM), lty = 1:2)
legend("topright", lty = 1:2, cex = 0.7, bty = "n",
        legend = c(expression(paste(chi[p]**2, " cut-off")), "GM cut-off"))

# ICS with MCD estimates and regular covariance
plot(icsOutlierDA.MCD,  cex.lab = 0.7, cex.axis = 0.7, main = "ICS MCD-COV",
     cex.main = 0.7)

# ICS with default non-robust estimates
plot(icsOutlierDA, cex.lab = 0.7, cex.axis = 0.7, main = "ICS COV-COV4",
     cex.main = 0.7)
```

Here the default `ics2` scatter combination does not perform well. Only three outliers from the second group are detected, although the distance plot indicates three distance levels but the cut-off value is not good and the separation not clear especially when compared with the MCD-COV combination. However, when looking at the invariant coordinates on Figure 8, we can see that the default scatter combination reveals the 14 outliers and the masking effect appears only when computing the ICS distances.

```
# Code for the Figure 8
plot(icsHBK, col = rep(3:1, c(10, 4, 61)))
```

Conclusion and future developments

The use of **ICSOutlier** for outlier detection has been illustrated on several examples. The results are already pretty good with the default parameters as soon as the percentage of outliers is small. Note however that the components selection procedure can be improved sometimes by a visual inspection of the screeplot and the cut-off for outlier identification can be adjusted by looking at the ICS distances plot. The ability of the method to check for the absence of outliers is an advantage compared with methods such as the Mahalanobis distance. In the same vein, the number of false positives, i.e the number of non-outliers declared as outliers, is restricted thanks to the use of the cut-off derived from simulations. Moreover, the real HTP data set included in the package shows that the ICS method is useful for outlier detection in the context of quality control, among possible applications. Finally the ICS method competes with common approaches based on distances (the Mahalanobis distance, its robust version, the PCA methods, its variants and improvements) as well as other methods based on the density (LOF) or on angles (ABOD).

The computation time for cut-offs for the selection of the non-Gaussian invariant components, on the one hand, and for the threshold for labeling the outliers, on the other hand, can be reduced

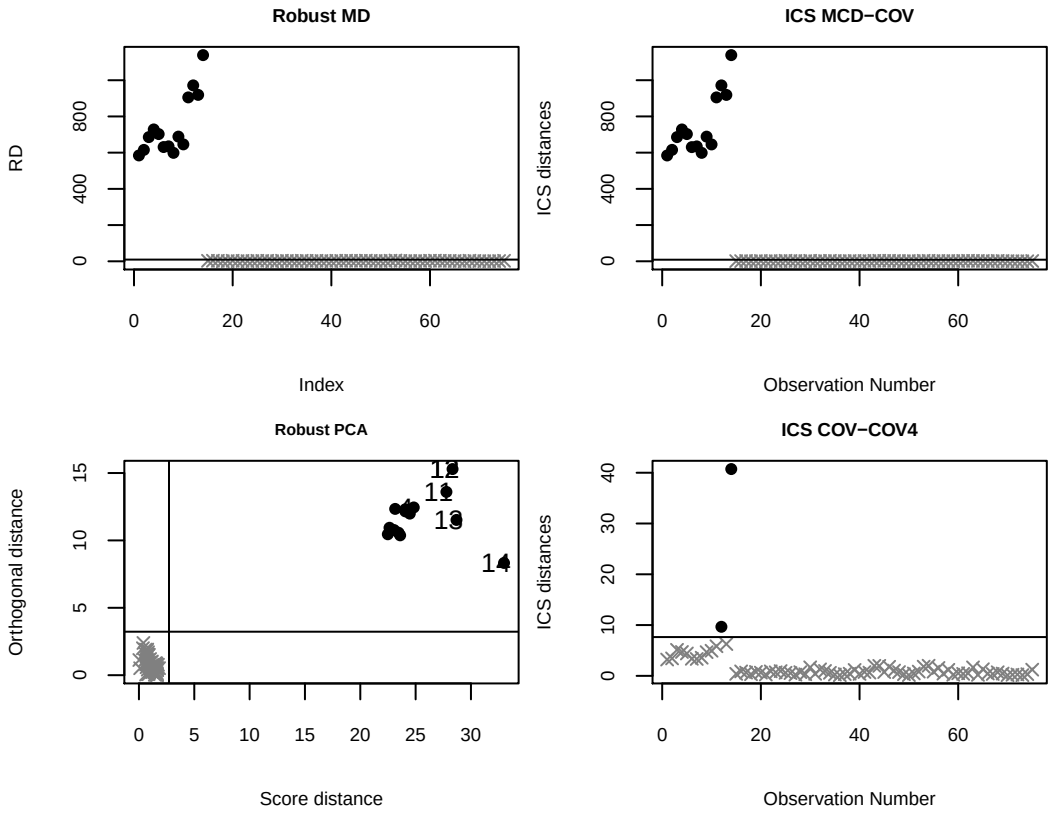


Figure 7: Squared robust Mahalanobis distance based on the MCD (left panel), squared ICS distances with MCD and Mean-Cov (middle panel), and squared ICS distances with default scatters (right panel). All cut-offs values are at level of 2.5%

by using the parallelizing option. Moreover, for now, the function can only select the first invariant components which is relevant when the percentage of outliers is small. For a large percentage, the invariant components associated with the smallest eigenvalues may also be of interest and this is a current perspective of the present package. A generalization of the ICS method in case of perfectly collinear variables is also a work in progress.

Acknowledgments

The article is based upon work from CRoNoS COST Action IC1408, supported by COST (European Cooperation in Science and Technology).

The authors wish to thank the editor and the two reviewers for their comments and suggestions which helped improve not only the paper but also the R package.

Bibliography

C. C. Aggarwal. *Outlier Analysis, 2nd Edition*. Springer-Verlag, 2017. ISBN 978-3-319-47577-6. URL https://doi.org/10.1007/978-3-319-47578-3. [p234]

A. Archimbaud, K. Nordhausen, and A. Ruiz-Gazen. *ICSOutlier: Outlier Detection Using Invariant Coordinate Selection*, 2016. URL https://CRAN.R-project.org/package=ICSOutlier. R package version 0.3-0. [p235]

A. Archimbaud, K. Nordhausen, and A. Ruiz-Gazen. Ics for multivariate outlier detection with application to quality control. *Computational Statistics & Data Analysis*, 128:184–199, 2018. ISSN 0167-9473. URL https://doi.org/10.1016/j.csda.2018.06.011. [p235, 236, 237, 239, 240, 242, 243, 244]

D. G. Bonett and E. Seier. A test of normality with high uniform power. *Computational Statistics & Data Analysis*, 40(3):435–445, 2002. URL https://doi.org/10.1016/S0167-9473(02)00074-9. [p236]

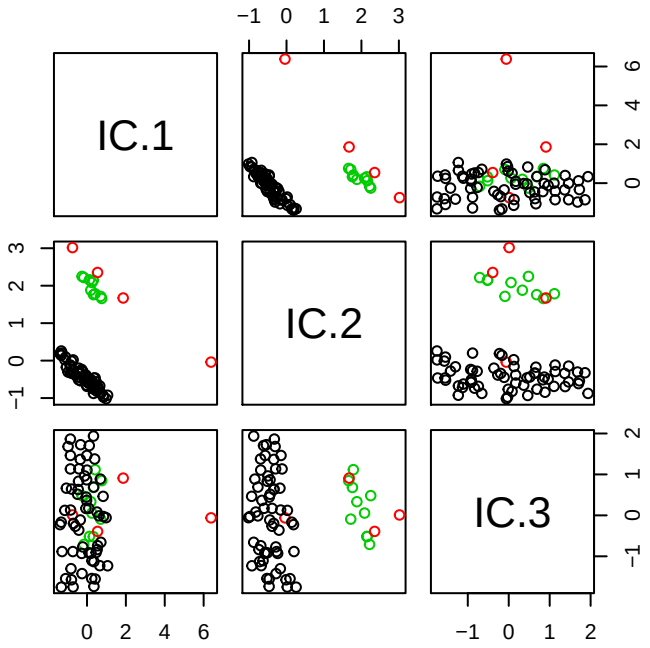


Figure 8: Scatter plot of the invariant coordinates of the hbk data for the default scatter combinations. The two outlier groups are marked with different colors.

M. M. Breunig, H.-P. Kriegel, R. T. Ng, and J. Sander. LOF: Identifying density-based local outliers. In *ACM Sigmod Record*, volume 29, pages 93–104. ACM, 2000. URL https://doi.org/10.1145/335191.335388. [p243]

A. Cerioli. Multivariate outlier detection with high-breakdown estimators. *Journal of the American Statistical Association*, 105(489):147–156, 2010. URL https://doi.org/10.1198/jasa.2009.tm09147. [p234, 241]

R. P. Chalmers and D. B. Flora. Faoutlier: An R package for detecting influential cases in exploratory and confirmatory factor analysis. *Applied Psychological Measurement*, 39(7):573–574, 2015. URL https://doi.org/10.1177/0146621615597894. [p234]

S. Dray. On the number of principal components: A test of dimensionality based on measurements of similarity between matrices. *Computational Statistics & Data Analysis*, 52(4):2228 – 2237, 2008. URL https://doi.org/10.1016/j.csda.2007.07.015. [p236]

S.-H. Eo and H. Cho. *OutlierDC: Outlier Detection Using Quantile Regression for Censored Data*, 2014. URL https://CRAN.R-project.org/package=OutlierDC. R package version 0.3-0. [p234]

C. Fan. *HighDimOut: Outlier Detection Algorithms for High-Dimensional Data*, 2015. URL https://CRAN.R-project.org/package=HighDimOut. R package version 1.0.0. [p234]

P. Filzmoser and M. Gschwandtner. *Mvoutlier: Multivariate Outlier Detection Based on Robust Methods*, 2017. URL https://CRAN.R-project.org/package=mvoutlier. R package version 2.0.8. [p234]

P. Filzmoser and V. Todorov. Robust tools for the imperfect world. *Information Sciences*, 245:4–20, 2013. URL https://doi.org/10.1016/j.ins.2012.10.017. [p234]

P. Filzmoser, R. G. Garrett, and C. Reimann. Multivariate outlier detection in exploration geochemistry. *Computers & Geosciences*, 31(5):579–587, 2005. URL https://doi.org/10.1016/j.cageo.2004.11.013. [p234]

P. Filzmoser, R. Maronna, and M. Werner. Outlier identification in high dimensions. *Computational Statistics & Data Analysis*, 52(3):1694–1711, 2008. URL https://doi.org/10.1016/j.csda.2007.05.018. [p241]

D. Fischer, A. Berro, K. Nordhausen, and A. Ruiz-Gazen. REPPlab: An R package for detecting clusters and outliers using exploratory projection pursuit. arXiv preprint arXiv:1612.06518, 2016a. URL https://arxiv.org/abs/1612.06518. [p243]

D. Fischer, A. Berro, K. Nordhausen, and A. Ruiz-Gazen. *REPPlab: R Interface to 'EPP-Lab', a Java Program for Exploratory Projection Pursuit*, 2016b. URL https://CRAN.R-project.org/package=REPPlab. R package version 0.9.4. [p234]

C. Fraley. *HDoutliers: Leland Wilkinson's Algorithm for Detecting Multidimensional Outliers*, 2016. URL https://CRAN.R-project.org/package=HDoutliers. R package version 0.15. [p234]

M. Genest, J.-C. Masse, and J.-F. Plante. *Depth: Nonparametric Depth Functions for Multivariate Analysis*, 2017. URL https://CRAN.R-project.org/package=depth. R package version 2.1-1. [p234]

C. G. Green and D. Martin. *CerioliOutlierDetection: Outlier Detection Using the Iterated RMCD Method of Cerioli (2010)*, 2017a. URL https://CRAN.R-project.org/package=CerioliOutlierDetection. R package version 1.1.9. [p234]

C. G. Green and R. D. Martin. An extension of a method of Hardin and Rocke, with an application to multivariate outlier detection via the IRMCD method of Cerioli. Technical report, Working Paper, 2017b. URL http://christopherggreen.github.io/papers/hr05_extension.pdf. [p239]

A. S. Hadi, A. Imon, and M. Werner. Detection of outliers. *Wiley Interdisciplinary Reviews: Computational Statistics*, 1(1):57–70, 2009. URL https://doi.org/10.1002/wics.6. [p234]

D. M. Hawkins, D. Bradu, and G. V. Kass. Location of several outliers in multiple-regression data using elemental sets. *Technometrics*, 26(3):197–208, 1984. URL https://doi.org/10.1080/00401706.1984.10487956. [p244]

V. J. Hodge and J. Austin. A survey of outlier detection methodologies. *Artificial Intelligence Review*, 22 (2):85–126, 2004. URL https://doi.org/10.1007/s10462-004-4304-y. [p234]

Y. Hu, W. Murray, Y. Shan, and Australia. *Rlof: R Parallel Implementation of Local Outlier Factor (LOF)*, 2015. URL https://CRAN.R-project.org/package=Rlof. R package version 1.1.1. [p234]

J. Jimenez. *abodOutlier: Angle-Based Outlier Detection*, 2015. URL https://CRAN.R-project.org/package=abodOutlier. R package version 0.1. [p234]

L. Komsta. *Outliers: Tests for Outliers*, 2011. URL https://CRAN.R-project.org/package=outliers. R package version 0.14. [p234]

H.-P. Kriegel, A. Zimek, and others. Angle-based outlier detection in high-dimensional data. In *Proceedings of the 14th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 444–452. ACM, 2008. URL https://doi.org/10.1145/1401890.1401946. [p243]

K. Luu and M. Blum. *Pcadapt: Fast Principal Component Analysis for Outlier Detection*, 2017. URL https://CRAN.R-project.org/package=pcadapt. R package version 3.0.4. [p234]

K. Nordhausen and D. E. Tyler. A cautionary note on robust covariance plug-in methods. *Biometrika*, 102(3):573–588, 2015. URL https://doi.org/10.1093/biomet/asv022. [p235]

K. Nordhausen, H. Oja, and D. E. Tyler. Tools for exploring multivariate data: The package ICS. *Journal of Statistical Software*, 28(6):1–31, 2008. URL https://doi.org/10.18637/jss.v028.i06. [p235, 237]

P. R. Peres-Neto, D. A. Jackson, and K. M. Somers. How many principal components? stopping rules for determining the number of non-trivial axes revisited. *Computational Statistics & Data Analysis*, 49 (4):974 – 997, 2005. URL https://doi.org/10.1016/j.csda.2004.06.015. [p236]

A. Rehage and S. Kuhnt. *alphaOutlier: Obtain Alpha-Outlier Regions for Well-Known Probability Distributions*, 2016. URL https://CRAN.R-project.org/package=alphaOutlier. R package version 1.2.0. [p234]

P. Rousseeuw and M. Hubert. High-breakdown estimators of multivariate location and scatter. In *Robustness and Complex Data Structures*, pages 49–66. Springer-Verlag, 2013. URL https://doi.org/10.1007/978-3-642-35494-6_4. [p235]

P. Rousseeuw, C. Croux, V. Todorov, A. Ruckstuhl, M. Salibian-Barrera, T. Verbeke, M. Koller, and and Martin Mächler. *robustbase: Basic Robust Statistics*, 2017. URL http://CRAN.R-project.org/package=robustbase. 0.92-8. [p244]

V. Todorov. *rrcovHD: Robust Multivariate Methods for High Dimensional Data*, 2016. URL https://CRAN.R-project.org/package=rrcovHD. R package version 0.2-5. [p234]

V. Todorov and P. Filzmoser. An object-oriented framework for robust multivariate analysis. *Journal of Statistical Software*, 32(3):1–47, 2009. URL https://doi.org/10.18637/jss.v032.i03. [p234]

L. Torgo. *Data Mining with R, Learning with Case Studies, 2nd Edition*. Chapman and Hall/CRC, 2016. URL http://ltorgo.github.io/DMwR2. [p234]

D. E. Tyler, F. Critchley, L. Dümbgen, and H. Oja. Invariant coordinate selection. *Journal of the Royal Statistical Society B*, 71(3):549–592, 2009. URL https://doi.org/10.1111/j.1467-9868.2009.00706.x. [p235, 236, 241]

M. van der Loo. *Extremevalues, an R Package for Outlier Detection in Univariate Data*, 2010. URL http://www.github.com/markvanderloo/extremevalues. R package version 2.3. [p234]

K. Williams. *Ldbod: Local Density-Based Outlier Detection*, 2017. URL https://CRAN.R-project.org/package=ldbod. R package version 0.1.2. [p234]

B. Yazici and S. Yolacan. A comparison of various tests of normality. *Journal of Statistical Computation and Simulation*, 77(2):175–183, 2007. URL https://doi.org/10.1080/10629360600678310. [p236]

Aurore Archimbaud
TSE-R, University Toulouse 1 Capitole
21 allée de Brienne, 31000 Toulouse
France
aurore.archimbaud@ut-capitole.fr

Klaus Nordhausen
CSTAT - Computational Statistics, Institute of Statistics & Mathematical Methods in Economics
Vienna University of Technology, Wiedner Hauptstr. 7, A-1040 Vienna
Austria
(0000-0002-3758-8501)
klaus.nordhausen@tuwien.ac.at

Anne Ruiz-Gazen
TSE-R, University Toulouse 1 Capitole
21 allée de Brienne, 31000 Toulouse
France
anne.ruiz-gazen@tse-fr.eu

rpostgis: Linking R with a PostGIS Spatial Database

by David Bucklin and Mathieu Basille

Abstract With the proliferation of sensors and the ease of data collection from online sources, large datasets have become the norm in many scientific disciplines, and efficient data storage, management, and retrival is imperative for large research projects. Relational databases provide a solution, but in order to be useful, must be able to be linked to analysis and visualization tools, such as R. Here, we present a package intended to facilitate integration of R with the open-source database software PostgreSQL, with a focus on its spatial extension, PostGIS. The package **rpostgis** (version 1.4.1) provides methods for spatial data handling (vector and raster) between PostGIS-enabled databases and R, methods for R `"data.frame"`s storage in PostgreSQL, and a set of convenient wrappers for common database procedures. We thus expect **rpostgis** to be useful for both (1) existing users of spatial data in R and/or PostGIS, and (2) R users who have yet to adopt relational databases for their projects.

Introduction

R has become an important tool for manipulating, analyzing, and displaying spatial (vector and raster) datasets. R has a growing set of contributed packages (see the CRAN *Spatial* task view). R can already import and export stand-alone vector (e.g, ESRI shapefiles) with relative ease using **rgdal** (Bivand et al., 2017) or **maptools** (Bivand and Lewin-Koh, 2017), and raster datasets using **raster** (Hijmans, 2016). With the proliferation and availability of geographic data from deployed sensors, open-GIS web data sources, and remotely-sensed environmental datasets (to name just a few), users are increasingly taking advantage of Database Management Systems (DBMS) with spatial database extenders, which store spatial data as an object in a database table column. DBMS are especially useful to researchers and scientists managing project datasets, due to the ability to store large amounts of data efficiently, specify data relationships among tables, maintain data integrity using constraints, ensure long-term storage, manage multiple users, and add, update, and retrieve data efficiently.

PostgreSQL (The PostgreSQL Global Development Group, 2017) is an advanced relational DBMS, and it is also free and open-source, making it popular and accessible to a large base of users. The PostgreSQL extension **PostGIS** (PostGIS Project Steering Committee, 2017) allows PostgreSQL to handle spatial data by introducing spatial data types (Geometry, Geography, Raster, and Topology), along with a library of functions which operate on objects of these types. For R users, the package **RPostgreSQL** (Conway et al., 2017), a backend for the generic R database interface package **DBI** (R Special Interest Group on Databases (R-SIG-DB) et al., 2017), provides the driver and methods to connect and interface with a PostgreSQL database, to do a variety of data import and export tasks, and to query the database. However, not all data types supported by PostgreSQL can be imported into equivalent R data types using **RPostgreSQL**; among these are the PostGIS spatial data types. In addition, some data types in R possess attributes (e.g., POSIX* time zones), which are not handled by **RPostgreSQL** reading and writing. Finally, while **RPostgreSQL** is an essential tool for users of PostgreSQL databases and R, it is not designed to facilitate new database users, who may not be familiar with database techniques, terminology, or Structured Query Language (SQL), the language used to interact with a DBMS.

Noting these gaps, we designed **rpostgis** as a general-purpose extension package for **RPostgreSQL**. The primary goal of **rpostgis** is to provide spatial data handling between R and PostGIS, and as such, it includes methods for bi-directional transfer of both vector and raster spatial datasets between R and PostGIS, spatial projection handling, and other PostGIS-related tools. In addition, the package also features methods for users who wish to work primarily (or exclusively) in R, including saving of R `"data.frame"`s (including data types and attributes) to PostgreSQL, and "SQL wrapper" functions which execute common database procedures. The latter provide not only convenient, script-able access to these procedures through R, but also can function as learning tools for new users of DBMS and SQL. In this paper, we explain and demonstrate the purpose and usage of **rpostgis** functions within a general workflow, following the three focal areas described above (PostGIS-specific, `"data.frame"` handling, and SQL wrappers).

Background

PostgreSQL and rpostgis basics

While it is far beyond the scope of this paper to provide a complete introduction to PostgreSQL, it is important to introduce several basic features of the DBMS, and the common terms we use to refer to them, especially concerning their usage in **rpostgis**.

A PostgreSQL install creates a PostgreSQL *server*, which is the *host* of one or more *databases*. Databases within a server are self contained (data cannot be shared across databases). Users interact with the database server using Structured Query Language (*SQL*) to write *queries*. Within a database, there are one or more named *schema*s, which are collections of database objects. In PostgreSQL, on new database creation, a default schema is created in the database with the name **public**.

Schemas can be used to organize *object*s in a database. Schemas can contain many different objects, including functions, data types, and sequences; the objects which users of **rpostgis** should be most aware of are *table*s and *view*s. Tables store data, while standard views only store an SQL query, that when executed, creates a virtual table (in memory), using data from other tables or views. There is no functional difference between how tables and views (which both fall under the umbrella term *relation*) are referred to within a query, with the convention being '`schema_name.relation_name`'. Relations typically contain one or more *column*s, and each column has a defined data *type* (e.g., integer, character, timestamp with time zone), and optionally a defined *sub-type*, that specifies what type of data the column can contain. A column can also store records representing a spatial type, which is described in the next section. Relations have 0 or more *record*s, or rows, which contain the data.

In **rpostgis**, most functions perform an action on a relation. Following **RPostgreSQL** convention, the relation name is supplied to the argument `name`, which can be given either a length-one or length-two character vector. One-length character elements are interpreted as a relation name, and looked for in the default user schema(s) (by default, '`public`'). Length-two character vectors are interpreted as schema and relation name, e.g., `name = c("schema_name","table_name")`. It is generally recommended to provide both schema and relation name, since identical relation names can be re-used across schemas. For ease of reading, from this point on we refer to relations (tables and views) generally as "tables" throughout this paper.

This brief summary should provide the new user with enough terminology to understand this paper, and begin working with databases using **rpostgis**. New users are encouraged to learn more about PostgreSQL from its excellent online documentation (`https://www.postgresql.org/docs/manuals/`).

Spatial objects in PostGIS and R

Vector spatial data in PostGIS tables are stored as **GEOMETRY** (planar) or **GEOGRAPHY** (spherical) types in a column, and usually have a specified sub-type, such as **POINT**, **LINESTRING**, or **POLYGON**. Each record of a GEOMETRY/GEOGRAPHY column represents one spatial *feature*, which represents one or more geometric objects. Records containing multiple geometric objects in one feature are specified as a **MULTI** sub-type (e.g., **MULTIPOLYGON**). Geometry and geography columns can also store a spatial reference identifier (**SRID**) attribute, which is an integer value referring to the spatial reference system (the "projection") of the spatial data. Since spherical data are represented using geographic longitude/latitude coordinates on the sphere (specifically, the spheroid defined by WGS 84), the Geography data type is restricted to '`SRID = 4326`'.

Raster spatial data are stored in PostGIS using the **RASTER** data type. Raster columns also can store an SRID attribute defining the projection of the raster. One raster record can store one or more **bands** (rasters with identical spatial coverage but different data). In PostGIS, it is common to store one raster dataset in one PostGIS table, but split the raster into multiple **tiles**. A raster tile is one rectangular spatial subset, or block, of a raster dataset. One tile of a PostGIS raster type corresponds with one record in the corresponding table. Storing rasters in this way allows for more efficient spatial queries on the raster, by working on subsets of the raster.

Geometry, geography, and raster data types can all be exported from PostGIS using a variety of supplied export functions. In addition, the PostGIS library contains a large set of GIS functions to manage, construct, and edit Geometry objects, as well as measure spatial relationships between geometries, making it a full-featured GIS system. However, PostgreSQL/PostGIS is not packaged with a native software to visualize spatial data, so most users employ 3rd-party software to access the database whenever visualization is needed. For spatial data, desktop GIS systems like **ArcGIS** (ESRI, 2017) and **QGIS** (QGIS Development Team, 2017) can load spatial data directly as layers in the mapping environment, where users can take advantage of tools available in those software.

Because of its advanced statistical, processing, and visualization capabilities, many users employ R as a front-end for their database systems. In R, the long-time standard for handling vector spatial objects are "Spatial*" classes (e.g., "SpatialPoints", "SpatialMultiPoints", "SpatialLines", and "SpatialPolygons" and their "Spatial*DataFrame" variants) provided by the package **sp** (Pebesma and Bivand, 2005). Likewise the **raster** package provides standard methods for handling raster datasets in R, in "Raster*" classes (e.g., "RasterLayer" for single-band, "RasterBrick" or "RasterStack" for multi-band rasters). There are several notable packages and utilities that assist in transfer of spatial data between PostGIS and **sp** "Spatial*"-objects or "Raster*"-objects in R:

1. **rgdal** : The R package providing bindings to GDAL (Geospatial Data Abstraction Library) provides the functions readOGR and writeOGR to read, and write, respectively, PostGIS tables with 'GEOMETRY' columns and R "Spatial*" objects. A limitation of readOGR is that it cannot query the database table to obtain only a subset of the table; similarly writeOGR can only write new tables (or overwrite existing ones); it does not allow writing to existing database tables. The **rgdal** package also provides methods for reading rasters from PostGIS tables with readGDAL, though GDAL writing of rasters to PostGIS tables is (as of writing) not available, but potentially still under development (https://trac.osgeo.org/gdal/wiki/frmts_wtkraster.html). One major drawback of this solution is that the PostGIS driver for **rgdal** is not included by default on Windows operating systems, meaning most Windows users cannot use these methods without manually installing **rgdal** from source.
2. **rgeos** (Bivand and Rundel, 2017): The functions readWKT and writeWKT provide conversion between the WKT format and "Spatial*"-objects. WKT format "well-known text" is a standardized, text-based version of a vector geometry, which can be written and read using PostGIS functions.
3. **wkb** (TIBCO Software Inc., 2016): The R package offers the functions readWKB and writeWKB functions, which convert vector geometries to and from the WKB format ("well-Known binary").
4. Alternatively, it is possible to use system calls to command-line utilities such as ogr2ogr, shp2pgsql or raster2pgsql, which brings the full power of these utilities at the expense of additional complexity, especially for scripting: This solution mixes two syntaxes (R syntax and the one of the command-line utility) and requires the passing of passwords at every call.

Read and write functions from both **rgeos** and **wkb** are utilized in **rpostgis** import/export functions to provide the translations of geometries from and to R's "Spatial*" objects. The **rpostgis** functions wrap additional functionality around these low-level functions in the vector data export (pgInsert) and import (pgGetGeom) functions, to ease data transferability and managed data stored alongside vector geometries in R "data.frame"s or PostGIS tables.

Also of importance is the recently developed package **sf** (Pebesma, 2017), which provides Simple Features access in R, for vector spatial data. Simple Features is an Open Geospatial Consortium (OGC) and International Organization for Standardization (ISO) standard for storing and accessing geometry objects, and is used by many RDBMS, including PostGIS. Like **sp**, **sf** provides standardized, comprehensive spatial data handling in R, notably using a "data.frame" with a geometry-list column to store Simple Feature objects, instead of the various "Spatial*" classes utilized in **sp**. The **sf** package also provides its own functions for reading and writing geometries from PostGIS databases, both through GDAL (using st_read/st_write) and directly (using st_read_db/st_write_db). At the time of writing, **rpostgis** reads from and writes to **sp**-class "Spatial*" objects, though we anticipate a migration to **sf**-class objects as they become the new standard for vector spatial data in R.

A note on permissions and privileges

A proper administration of a database, especially in a multi-user context, requires a consideration of the permissions given to users to access, write, or modify the database and its various objects. While it is beyond the scope of this paper to provide an exhaustive presentation of the issue, it is worth introducing the basic concepts from a PostgreSQL perspective: permissions on a database object are called "privileges," and there are several different ones, allowing to access ('SELECT') or modify them ('INSERT', 'UPDATE'), or even to create or delete them ('CREATE' and 'DELETE', respectively), among others. By default, the owner of an object (the user who created the object) and superusers (generally the DBMS administrator(s)) can do anything with it: they are granted all privileges on this object. PostgreSQL uses the functions 'GRANT' and 'REVOKE' to assign or revoke privileges of a certain user on a database object. We refer the interested reader (and the database administrator) to the official PostgreSQL documentation for more details about privileges.

Database privileges for a given user are not different by accessing the database through a connection from R, although it adds a layer of complexity as there will be R users and PostgreSQL users mixed in a session (as well as system users, which are distinct from database users, and we do not

consider here purposely). In other words, an R user could have different privileges on the database and its objects depending on the PostgreSQL user they use to connect to the database. In this paper, we will use the default PostgreSQL superuser 'postgres', which thus comes with all privileges granted on all database objects. This approach is fine for many use cases, especially single-user databases, but will be limited on multi-user databases, where privileges are set up at a finer grain. In general, all **rpostgis** functions that create, modify, or drop an object (e.g., `dbIndex`, `dbAsDate`, `dbDrop`) require ownership of the object being modified (so they require the appropriate 'CREATE' privileges). In addition, we specifically indicate special privileges that are necessary for the different functions of **rpostgis** when relevant.

PostGIS-specific functions

In the following two sections, we explain the main functionality and usage of the **rpostgis** package. Note that functions within **rpostgis** have one of two prefixes: "`pg*`" (for PostGIS-oriented functions, described in this section) and "`db*`" (for general PostgreSQL database functions, described in the next section), emulating the pattern established by **DBI**. The general presentation of the functions consist of (1) a code block presenting the generic version of the function(s) with all arguments and defaults listed, as well as specific privileges required, (2) descriptive text about the function(s), and in most cases (3) example calls of the function(s).

In this section, we use the term "PostGIS" generically to refer to the PostgreSQL/PostGIS DBMS; that is, PostgreSQL with the PostGIS extension installed. All usage of PostGIS within R begins with a connection to a database server and a particular database: here, using `RPostgreSQL::dbConnect`, we connect to the database 'rpostgis' which exists on the local computer/server ('localhost'):

```
> library(rpostgis)
> conn <- dbConnect(drv = "PostgreSQL", host = "localhost", dbname = "rpostgis",
+     user = "postgres", password = "postgres_password")
```

It is not the purpose of this article to detail connection details, so we refer to documentation from **DBI** and **RPostgreSQL** for this aspect.

PostGIS management

Check and create PostGIS extension: `pgPostGIS`

```
> pgPostGIS(conn, topology = FALSE, tiger = FALSE, sfcgal = FALSE,
+     display = TRUE, exec = TRUE)
```

Special privileges: `pgPostGIS` requires ownership of the database (or superuser role) to install the extension ('CREATE EXTENSION') the first time.

The "starter" function for new users of **rpostgis** is `pgPostGIS`. This function installs and/or checks the version of PostGIS currently available on the database. This is the first example of a function in **rpostgis** which implements the `display` and `exec` arguments. These respectively control printing (to the R console) of the constructed SQL query, and execution of the query on the database. When PostGIS is installed and ready to use, `pgPostGIS` returns `TRUE`. A standard in **rpostgis**: is when a function does not return an object, the function returns `TRUE` if the action was successfully executed in the database. Note that you can also enable the Topology, Tiger Geocoder, and SFCGAL extensions, with `topology`, `tiger`, and `sfcgal` arguments set to `TRUE`.

List geometries/rasters: `pgListGeom` and `pgListRast`

```
> pgListGeom(conn, geog = TRUE)
> pgListRast(conn)
```

These `pgList*` functions return information on GEOMETRY/GEOGRAPHY and RASTER columns stored in any database table. In a PostGIS database, these information are stored in three views ('geometry_columns', 'geography_columns', and 'raster_columns', respectively), all of which are created during the installation of PostGIS in the 'public' schema of the database.

Find (or create) PostGIS SRID based on CRS object: `pgSRID`

```
> pgSRID(conn, crs, create.srid = FALSE, new.srid = NULL)
```

Special privileges: pgSRID requires 'INSERT' privilege on the 'spatial_ref_sys' table with create.srid = TRUE.

On installation of PostGIS, a new table 'spatial_ref_sys' is created in the 'public' schema, which stores a large set of spatial reference systems, each of which has a unique (integer) spatial reference identifier (SRID), along with specifications of the projection in *PROJ.4* and *WKT*. The function pgSRID allows users to check if a projection they are using in R (stored as a "sp::CRS" object, which contains the *PROJ.4* representation of the projection) has matching SRID(s) in PostGIS. If there is no match and create.srid = TRUE, then pgSRID adds it to the 'spatial_ref_sys' table. In these cases, the user can also specify a desired SRID with new.srid; otherwise, **rpostgis** uses the next available value between 880001 and 889999. In the following example, we demonstrate how to find the SRID for the common WGS 1984 latitude/longitude projection:

```
> crs <- sp::CRS("+proj=longlat +datum=WGS84 +no_defs +ellps=WGS84 +towgs84=0,0,0")
> pgSRID(conn, crs)

[1] 4326
```

In some cases, pgSRID may return multiple SRIDs, since there are equivalent projections stored in multiple SRIDs in the 'spatial_ref_sys' table. Note that pgSRID is used in spatial data writing functions in **rpostgis**, with create.srid = TRUE.

Spatial data transfer

The following section describes the functions in **rpostgis** that transfer spatial data between R and PostGIS databases (Table 1 contains a summary of all data transfer functions, including for non-spatial data). Outside of conn and name (discussed previously), there are several other arguments re-used across functions, which have consistent default values:

- geom: this is the column name in the PostGIS table containing a Geometry or Geography data type. Defaults to "geom".
- rast: this is the column name in the PostGIS table containing a Raster data type. Defaults to "rast".
- overwrite: in writing to database functions, this defaults to FALSE; overwrite = TRUE allows the user to delete ('DROP') the existing table and create a new one.

In this section we begin to use example datasets. We first load the well-known meuse dataset containing information on environmental observations from sample points along the Meuse river in the Netherlands. After loading the data, we create a "SpatialPointsDataFrame" object, setting its associated projection (oblique stereographic for the Netherlands, i.e. EPSG 28992) as the proj4string attribute, which is a standardized character representation of the projection of class "CRS":

```
> library(sp)
> data("meuse")
> meuse <- SpatialPointsDataFrame(meuse[, 1:2], data = meuse[,
+     3:length(meuse)], proj4string = sp::CRS("+init=epsg:28992"))
> class(meuse)

[1] "SpatialPointsDataFrame"
attr(,"package")
[1] "sp"

> head(meuse@data)

  cadmium copper lead zinc  elev       dist   om ffreq soil lime landuse
1    11.7     85  299 1022 7.909 0.00135803 13.6     1    1    1      Ah
2     8.6     81  277 1141 6.983 0.01222430 14.0     1    1    1      Ah
3     6.5     68  199  640 7.800 0.10302900 13.0     1    1    1      Ah
4     2.6     81  116  257 7.655 0.19009400  8.0     1    2    0      Ga
5     2.8     48  117  269 7.480 0.27709000  8.7     1    2    0      Ah
6     3.0     61  137  281 7.791 0.36406700  7.8     1    2    0      Ga
  dist.m
1     50
2     30
3    150
4    270
5    380
6    470
```

Function	Reads from:	Writes to:
pgInsert	"Spatial*", "Spatial*DataFrame", "data.frame"	new or existing database table (with GEOMETRY or GEOGRAPHY column for spatial objects)
pgWriteRast	"RasterLayer", "RasterBrick", "RasterStack", "SpatialPixels", "SpatialPixelsDataFrame", "SpatialGrid", "SpatialGridDataFrame"	new database table
dbWriteDataFrame	"data.frame"	new database table
pgGetGeom	database table/view with GEOMETRY or GEOGRAPHY column	"Spatial*", "Spatial*DataFrame"
pgGetRast	database table/view with RASTER column	"RasterLayer", "RasterBrick", "SpatialPixels", "SpatialPixelsDataFrame", "SpatialGrid", "SpatialGridDataFrame"
pgGetBoundary	database table/view with GEOMETRY, GEOGRAPHY, or RASTER column	"SpatialPolygons"
dbReadDataFrame	database table/view	"data.frame"

Table 1: Functions for data transfer between R and PostgreSQL/PostGIS in **rpostgis**.

Vector spatial data: `pgInsert` **and** `pgGetGeom`

```
> pgInsert(conn, name, data.obj, geom = "geom", df.mode = FALSE,
+     partial.match = FALSE, overwrite = FALSE, new.id = NULL,
+     row.names = FALSE, upsert.using = NULL, alter.names = FALSE,
+     encoding = NULL, return.pgi = FALSE, df.geom = NULL, geog = FALSE)
```

Special privileges: `pgInsert` requires 'CREATE' privilege in schema for new tables, 'INSERT' privilege on existing tables, and 'UPDATE' privilege when using `upsert.using` argument.

As evidenced by the large number of possible arguments, `pgInsert` is a flexible function that aims to provide a variety of methods for PostGIS 'INSERT's, from data originating in R as an object of class "Spatial*", "Spatial*DataFrame", or "data.frame". The most basic usage requires just a PostgreSQL connection, a database table name to insert into, and a data object of one of the three types. In our example, we also utilize the `new.id` argument, providing a new column name for a sequential ID ("gid"):

```
> pgInsert(conn, "meuse", meuse, new.id = "gid")

Creating new table...

Using writeWKB from wkb package...

Data inserted into table "public"."meuse"

[1] TRUE
```

The data was inserted into the new database table 'meuse_sp' in the 'public' schema, printing out several informative messages, including an indication that `wkb::writeWKB` was used to convert spatial objects to well-known binary in order to write to the database (if the optional package **wkb** is not installed, `rgeos::writeWKT` is used instead). So that is clear to users how and where data is written to the database, `pgInsert` is a fairly verbose function—however, these messages can be hidden by wrapping the function call within `suppressMessages(...)`.

Since pgInsert can also insert into existing tables, a partial.match argument allows specifying if all columns in the R object must be present in the database in order to do the insert. The default is partial.match = FALSE, meaning if any column is found in the R object but not in the database table, an error is printed and no data is written in the database. Note that the check is uni-directional—database table columns not found in the R data object do not affect the insert, except if the database column definition requires that it have assigned values (e.g., 'NOT NULL' with no assigned 'DEFAULT' value).

```
> pgGetGeom(conn, name, geom = "geom", gid = NULL, other.cols = TRUE,
+     clauses = NULL, boundary = NULL, query = NULL)
```

Special privileges: pgGetGeom requires 'SELECT' on the table, and 'CREATE' in the schema to create a new view when both query and name are not null

pgGetGeom returns vector spatial data from a PostGIS table storing a Geometry or Geography—the column name of either type can be supplied to the geom column. We can retrieve the full 'meuse' table which we just wrote to the database, demonstrating the most basic use of the function:

```
> meuse.db <- pgGetGeom(conn, "meuse")

Returning Point types in SpatialPoints*-class.
```

A message prints the type of "Spatial*" class that is created in R. The other.cols argument can be a vector of character names of database table columns, indicating which (if any) columns to return with the spatial data. The default is the full table (all columns), but other.cols = FALSE returns the spatial data only (i.e., a "Spatial*" object). The clauses argument can take additional SQL to modify the data to take from the table. For instance, using a 'WHERE' clause allows filtering results based on specific filter, as long as it is valid SQL (in this case, indexing the filtering column will significantly speed up the query; see the section "Create an index: dbIndex" below). Conversely, a full SQL query can be provided using the query argument; in this case the names argument can be NULL, or set to the name of a (new) view to create in the database using the specified query. This is the method to use when it is necessary to combine multiple database tables and return a geometry, or use PostGIS functions within a query, which highlights the potential of **rpostgis** for complex GIS operations in the database.

For example, we can use the PostGIS function 'ST_Buffer' to create 100-m buffers around points in our 'meuse' table, and then transform the layer to another projection using 'ST_Transform', with 'SRID = 4326' (WGS 1984, lat-lon). Note that we specified the output geometry in the SQL query 'AS geom', which is the default name for GEOMETRY columns for pgGetGeom. This query returns only the original points ID (gid) and a GEOMETRY column that can be directly loaded into a "SpatialPolygons" object using pgGetGeom. Since we specify gid as the unique ID for meuse.buff (using gid = "gid"), and there are no other columns of data to import, there is no data frame associated with this object. We can then plot the result to highlight the "flat" shape of the buffers after reprojection (Figure 1):

```
> query <- "SELECT gid, ST_Transform(ST_Buffer(geom, 100), 4326) AS geom FROM meuse;"
> meuse.buff <- pgGetGeom(conn, name = "meuse_buff", query = query,
+     gid = "gid")

Returning Polygon types in SpatialPolygons*-class.

Created view "public"."meuse_buff".

> par(mar = c(2, 2, 2, 2))
> plot(meuse.buff, axes = TRUE, asp = 1)
```

Raster spatial data: pgWriteRast and pgGetRast

```
> pgWriteRast(conn, name, raster, bit.depth = NULL, blocks = NULL,
+     constraints = TRUE, overwrite = FALSE)
```

Special privileges: pgWriteRast requires 'CREATE' privilege in schema.

The pgWriteRast function sends R rasters to a new database table. We can demonstrate this loading the meuse.grid data set into a "SpatialPixelsDataFrame" object:

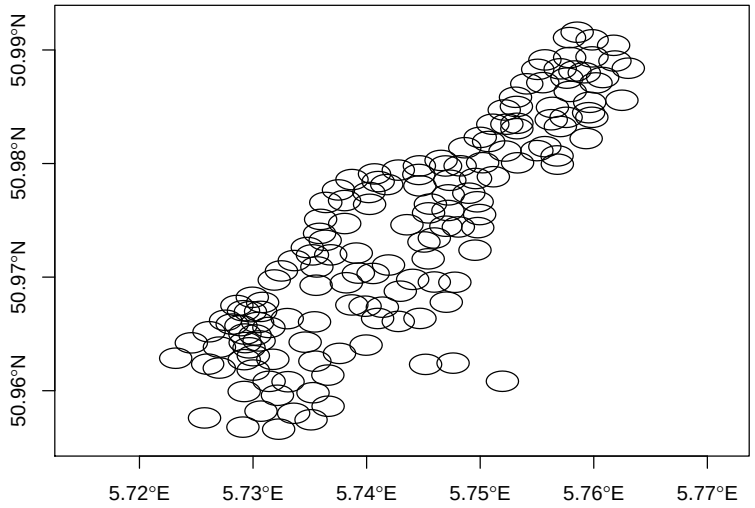


Figure 1: Buffered points (as polygons) from meuse dataset, created and loaded from a PostGIS query.

```
> data("meuse.grid")
> meuse.grid <- SpatialPointsDataFrame(meuse.grid[, 1:2], data = meuse.grid[,
+     3:length(meuse.grid)], proj4string = sp::CRS("+init=epsg:28992"))
> gridded(meuse.grid) <- TRUE
> class(meuse.grid)

[1] "SpatialPixelsDataFrame"
attr(,"package")
[1] "sp"
```

The **rpostgis** package works with `"Raster*"`-class or gridded `"Spatial*"` objects (i.e. `"SpatialPixels"`, `"SpatialGrid"`, and their `"*DataFrame"` counterparts). We can create directly send `meuse.grid`, a `"SpatialPixelsDataFrame"` object, to a new raster table in the database using `pgWriteRast`. Its five variables are converted into raster "bands" in the PostGIS data structure:

```
> pgWriteRast(conn, "meuse_rast", meuse.grid)

Splitting 5 band(s) into 1 x 1 blocks...

[1] TRUE
```

Since `pgWriteRast` is only for new database tables (or overwrites), the column storing the raster in the database is fixed to `"rast"`. The function automatically splits the raster into tiles (called "blocks" in the **raster** package), with one tile for each record in the table, and adds automatic raster constraints (when `constraints = TRUE`, the default). Setting raster constraints in PostGIS allows the raster overview data to be registered correctly in the 'raster_columns' view, and include attributes of the raster such as scale, extent, SRID, and checks on alignment between blocks in the raster table.

In this example case, we see that the raster only includes one block (1×1), since `meuse.grid` is relatively small (~8,000 cells); the default target block size in `pgWriteRast` is 10,000 cells (e.g., a 100×100 cell raster). Optionally, the user can specify the exact number of blocks to split the raster into using the `blocks` argument set to either a length-one or two integer vector (e.g., `blocks = 5` splits the raster into 5×5 blocks, and `blocks = c(2,4)` splits the raster into 2×4 blocks). Note that specifying a smaller number of blocks will reduce write time using `pgWriteRast`. Block size should largely be determined by usage within PostGIS. For example, with larger blocks, PostGIS queries that only summarize across blocks will be less precise (since more total raster cells will fall within these blocks), and these queries could also be slower as a result.

The function `pgWriteRast` uses only the R **raster** package and SQL queries to create and populate rasters, meaning it is not dependent on external software or command-line utilities. While this makes it particularly useful as a cross-platform solution, and for new users who are not familiar with existing command-line tools for loading rasters into PostGIS (e.g., `raster2pgsql`), it may be less memory-efficient and take more time to process rasters than these command-line options. Users who frequently work with very large rasters should keep this consideration in mind.

```
> pgGetRast(conn, name, rast = "rast", bands = 1, boundary = NULL)
```

Special privileges: pgGetRast requires 'SELECT' privilege on the table.

Loading rasters from the database can be done with pgGetRast, where the column to retrieve from defaults to "rast". Since we did not specify any bands, the default is to return the first band:

```
> lc.db <- pgGetRast(conn, "meuse_rast")
> summary(lc.db)

Object of class SpatialPixelsDataFrame
Coordinates:
     min    max
x 178440 181560
y 329600 333760
Is projected: TRUE
proj4string :
[+init=epsg:28992 +proj=sterea +lat_0=52.15616055555555
+lon_0=5.38763888888889 +k=0.9999079 +x_0=155000 +y_0=463000
+ellps=bessel
+towgs84=565.4171,50.3319,465.5524,-0.398957,0.343988,-1.87740,4.0725
+units=m +no_defs]
Number of points: 3103
Grid attributes:
   cellcentre.offset cellsize cells.dim
s1            178460       40        78
s2            329620       40       104
Data attributes:
     part.a
 Min.   :0.0000
 1st Qu.:0.0000
 Median :0.0000
 Mean   :0.3986
 3rd Qu.:1.0000
 Max.   :1.0000
```

Multi-band writing and reading of is also possible with pgWriteRast and pgGetRast, as illustrated in this example using the RGB RasterBrick of the R logo (available in the raster package as a raster of 77×101 pixels, with three layers for red, green, and blue values). We also demonstrate the boundary argument method of pgGetRast by providing an extent as four numbers in the raster's projection [top, bottom, right, left]. We finally plot the output in Figure 2:

```
> rlogo <- raster::brick(system.file("external/rlogo.grd", package = "raster"))
> pgWriteRast(conn, "rlogo", rlogo)

Splitting 3 band(s) into 1 x 1 blocks...

[1] TRUE

> just.r <- pgGetRast(conn, "rlogo", bands = TRUE, boundary = c(60, 0, 95, 35))
> just.r

class       : RasterBrick
dimensions  : 60, 60, 3600, 3  (nrow, ncol, ncell, nlayers)
resolution  : 1, 1  (x, y)
extent      : 35, 95, 0, 60  (xmin, xmax, ymin, ymax)
coord. ref. : +proj=merc +datum=WGS84 +ellps=WGS84 +towgs84=0,0,0
data source : in memory
names       : red, green, blue
min values  :   0,     0,    1
max values  : 255,   255,  255

> raster::plotRGB(just.r)
```

The R data class, proj4string attribute, and band/column names of R raster objects are also saved in the raster table, and are imported when pgGetRast re-creates the raster in R. The boundary argument can also be specified as a "Spatial*" object, where the raster is clipped to the bounding box of the object. This can greatly increase the speed of the import, and is recommended when working with large rasters.

Figure 2: (clipped) Logo of R.

Users should note that "Raster*"-class data types are handled natively within the **rpostgis** raster functions. "Spatial*" objects are converted to/from "Raster*"s within the function execution; as such, there may be a performance benefit to working with "Raster*" objects.

Other spatial functions

Bounding box retrieval: pgGetBoundary

```
> pgGetBoundary(conn, name, geom = "geom", clauses = NULL)
```

Special privileges: pgGetBoundary requires 'SELECT' privilege on the table.

The function pgGetBoundary can return the outer bounding box of all spatial data (or a subset using clauses) in a PostGIS table, and the geom argument can actually take the column name of a Geometry, Geography, or Raster column. A single feature in a "SpatialPolygons" object is returned in the original projection of the spatial data, as in this example using the 'meuse_dist' raster table previously created:

```
> (m.bound <- pgGetBoundary(conn, "meuse_rast", "rast"))
```

```
class       : SpatialPolygons
features    : 1
extent      : 178440, 181560, 329600, 333760  (xmin, xmax, ymin, ymax)
coord. ref. : +proj=sterea +lat_0=52.15616055555555 +lon_0=5.38763888888889
 +k=0.9999079 +x_0=155000 +y_0=463000 +ellps=bessel
 +towgs84=565.2369,50.0087,465.658,-0.406857,0.350733,-1.87035,4.0812
 +units=m +no_defs
```

Add a POINT or LINESTRING geometry field: pgMakePts and pgMakeStp

```
> pgMakePts(conn, name, colname = "geom", x = "x", y = "y", srid,
+     index = TRUE, display = TRUE, exec = TRUE)
>
> pgMakeStp(conn, name, colname = "geom", x = "x", y = "y", dx = "dx",
+     dy = "dy", srid, index = TRUE, display = TRUE, exec = TRUE)
```

These are specialized functions that can be used to apply a new Geometry column to an existing database table, and are of particular use for managing sequential location data tables (i.e., trajectories). While pgMakePts requires only columns storing x and y coordinates to create a 'POINT GEOMETRY' column, pgMakeStp also needs increments along the x/y dimensions to the next point, and builds the line segments connecting those points into a 'LINESTRING GEOMETRY' column.

PostgreSQL functions and wrappers

Storing R "data.frame"s in PostgreSQL: dbWriteDataFrame and dbReadDataFrame

```
> dbWriteDataFrame(conn, name, df, overwrite = FALSE, only.defs = FALSE)
> dbReadDataFrame(conn, name, df = NULL)
```

Special privileges: dbWriteDataFrame requires 'CREATE' privilege in schema.

Data frame columns have attributes which store important information about the class of data in the column, as well information specific to that class. However, when writing a `"data.frame"` object to databases (e.g, using `RPostgreSQL::dbWriteTable`), these attributes are not carried over. For users who want to work in R, store data in a database, and then wish to retrieve it in the same state that it was prior to storing, **rpostgis** introduces a "data frame mode" for its data writing and reading functions.

The first time it is used in a schema, data frame mode creates a new database table named `'.R_df_defs'` to store column types and attributes. Within the data table itself, data frame mode adds a column to store the `"data.frame"` row names (in `'.R_rownames'`), and another to act as the database table's primary key (in `'.df_pkid'`), and provide sorting for the `"data.frame"`. Note that since `'.R_df_defs'` and `'.df_pkid'` are only intended for internal **rpostgis** usage, we begin them with the non-standard "." to function as a flag to users of the database. It is not recommended to name tables commonly referred to by users in this way, since non-standard naming of tables and column identifiers will require that they be double-quoted whenever they are used in a query.

When the table is imported into R using **rpostgis** reading functions (`dbReadDataFrame` or `pgGetGeom`), these metadata are applied to the resulting data frame. Currently (**rpostgis** version 1.4.0), the only data type attributes which are stored and re-created using data frame mode are time zones of `"POSIXct"` objects, and levels of factors (including ordered factors). However, handling of specific attributes for other data types can be added in a modular way to the main data frame mode functions (`dbWriteDataFrame` and `dbReadDataFrame`).

We can demonstrate the utility of data frame mode with a simple example on the dataset `meuse`, this time as a data frame. To further demonstrate the utility of data frame mode, we first add a date-time (`"POSIXct"`) column to `meuse`:

```
> data("meuse")
> meuse$example_time <- seq(as.POSIXct("2010-01-01 12:00", tz = "Europe/Amsterdam"),
+     by = "1 day", length.out = nrow(meuse))
> head(meuse$example_time)

[1] "2010-01-01 12:00:00 CET" "2010-01-02 12:00:00 CET"
[3] "2010-01-03 12:00:00 CET" "2010-01-04 12:00:00 CET"
[5] "2010-01-05 12:00:00 CET" "2010-01-06 12:00:00 CET"
```

First, we can demonstrate the base writing and reading of this table using **RPostgreSQL**, in particular the functions `dbWriteTable` and `dbReadTable`:

```
> RPostgreSQL::dbWriteTable(conn, "meuse_base", meuse)

[1] TRUE

> meuse.base <- RPostgreSQL::dbReadTable(conn, "meuse_base")
> all.equal(meuse, meuse.base)

[1] "Component \"ffreq\": 'current' is not a factor"
[2] "Component \"soil\": 'current' is not a factor"
[3] "Component \"lime\": 'current' is not a factor"
[4] "Component \"landuse\": 'current' is not a factor"
[5] "Component \"example_time\": Mean absolute difference: 21274.84"
```

The comparison with `all.equal` indicates that four of the columns that were factors in `meuse` are no longer factors, and there is a time difference between the `example_time` columns in the two data frames. In this case, times in `meuse` were set to the time zone "Europe/Amsterdam", while our database (and R base) time zone is "America/New_York". The functions `dbWriteTable` and `dbReadTable` ignore time zone specifications in a `"data.frame"`, assuming all timestamps are in the local computer's time zone. In case of mismatch between the data and the user time zone, this results in incorrect timestamps in both the database table (`'meuse_base'`), and the recreated R data frame (`meuse.base`), as is the case here.

Using data frame mode methods, `meuse` can be sent to the database, and re-created in R, with no alterations:

```
> dbWriteDataFrame(conn, "meuse_df", meuse)

New R data frame definitions table created ("public".".R_df_defs").

[1] TRUE
```

```
> meuse.df <- dbReadDataFrame(conn, "meuse_df")
> all.equal(meuse, meuse.df)

[1] TRUE
```

Since data frame mode writes column attributes (e.g., factor levels) based on the data exactly as it appears in an R "data.frame", dbWriteDataFrame can only be used for new database tables (or overwrite of existing ones). Data frame mode is also available for "Spatial" data using pgInsert, with df.mode = TRUE (the default is set to FALSE). pgGetGeom will automatically use "data.frame" definitions (when they exist) for importing "Spatial*" datasets. For example, we can again remake meuse as a "SpatialPointsDataFrame", and send it to the database, overwriting the previous 'meuse_df' table:

```
> meuse <- SpatialPointsDataFrame(meuse[, 1:2], data = meuse[,
+     3:length(meuse)], proj4string = sp::CRS("+init=epsg:28992"))
> pgInsert(conn, "meuse_df", meuse, df.mode = TRUE, overwrite = TRUE)

Using writeWKB from wkb package...

Query executed:

DROP TABLE IF EXISTS "public"."meuse_df";

Data inserted into table "public"."meuse_df"

[1] TRUE

> meuse_df <- pgGetGeom(conn, "meuse_df")

Returning Point types in SpatialPoints*-class.

> all.equal(meuse, meuse_df)

[1] TRUE
```

Using "data frame mode" with a "Spatial*" class object with pgInsert also saves the projection (as a proj4string attribute) of the spatial object in the database. This allows for equivalent re-creation of the meuse dataset in this example, since pgGetGeom re-applies this proj4string when importing the table 'meuse_df'.

SQL wrappers for database management

A collection of "db"-prefixed functions are available to add, remove, alter, and manage objects in the database, and SQL users will be familiar with the keywords in the function and attribute names. These functions share several common features:

- The previously discussed conn, name, display, and exec arguments;
- Each returns TRUE when the statement was successfully executed on the database.

In the following section, we present a short discussion of each function's purpose and a usage example.

Check and create schema: dbSchema

```
> dbSchema(conn, name, display = TRUE, exec = TRUE)
```

Special privileges: dbSchema requires 'CREATE' privileges in the database.

As previously mentioned, schemas are named storage partitions of a database, containing collections of other database objects. The function dbSchema creates a new schema name, or checks if it exists in the database:

```
> dbSchema(conn, "rpostgis_demo")

Query executed:

CREATE SCHEMA "rpostgis_demo";

[1] TRUE
```

Note that all schema, table, and column names are applied in the database exactly as they are given in the **rpostgis** functions, including capitalization, special characters, digits, etc. In PostgreSQL, however, object identifiers need to start with a lowercase letter (a-z) or an underscore (_), and contain letters, underscores, and digits (0-9). Any object that does not follow these rules must be quoted (using double quotation marks) when used in SQL queries. For that reason, all object names used in **rpostgis** (as well as **RPostgreSQL**) are quoted by default in the SQL queries that it builds. Despite this, it is good practice to use object names that do not require quoting, including column names in any data frames sent to the database; using this approach facilitates SQL query writing.

To demonstrate the use of schemas, we insert `meuse` into the '`rpsostgis_demo`' schema, only with a slight modification of the time variable: `example_time` is converted to character as to avoid incorrect specification of the time zone, and is stored as text type in the database.

To work in the '`rpostgis_demo`' schema, we need to specify the schema name as the first argument in `name`. We can also utilize the `new.id` argument to add a sequential number column '`meuse_id`' to the new table, and `alter.names` to make sure all column names do not require quoting:

```
> meuse$example_time <- as.character(meuse$example_time)
> pgInsert(conn, c("rpostgis_demo", "meuse"), meuse, new.id = "meuse_id",
+     alter.names = TRUE)

Creating new table...

Making column names DB-compliant (lowercase and replacing special characters with '_').

Using writeWKB from wkb package...

Data inserted into table "rpostgis_demo"."meuse"

[1] TRUE
```

Comment on table/view/schema: `dbComment`

```
> dbComment(conn, name, comment, type = c("table", "view", "schema"),
+     display = TRUE, exec = TRUE)
```

Commenting is essential to maintaining a well-documented database, and comments can be applied to any database object. Here we apply comments to both the new schema and table we just created:

```
> dbComment(conn, "rpostgis_demo",
+     comment = "Schema storing example data for the 'rpostgis' paper.",
+     type = "schema")

Query executed:

COMMENT ON SCHEMA "rpostgis_demo" IS 'Schema storing example data for the ''rpostgis'' paper.';

[1] TRUE

> dbComment(conn, c("rpostgis_demo", "meuse"),
+     comment = "Meuse river example dataset from R 'sp' package.",
+     type = "table")

Query executed:

COMMENT ON TABLE "rpostgis_demo"."meuse" IS 'Meuse river example dataset from R ''sp'' package.';

[1] TRUE
```

Add or remove a column: `dbColumn`

```
> dbColumn(conn, name, colname, action = c("add", "drop"), coltype = "integer",
+     cascade = FALSE, display = TRUE, exec = TRUE)
```

Adding or dropping a table column can be achieved with the `dbColumn` function: to demonstrate, we drop the '`dist`' column, so as not to confuse it with '`dist_m`':

```
> dbColumn(conn, c("rpostgis_demo", "meuse"), "dist", action = "drop")

Query executed:

ALTER TABLE "rpostgis_demo"."meuse" DROP COLUMN "dist" ;

[1] TRUE
```

Add a primary or foreign key: `dbAddKey`

```
> dbAddKey(conn, name, colname, type = c("primary", "foreign"),
+     reference, colref, display = TRUE, exec = TRUE)
```

Keys are an important element of a database, and are part of a broader group of elements used to maintain integrity of table data called *constraints*. Using `dbAddKey`, we can add a *primary key* (specifying a non-null column acting as the unique identifier of a single table) or a *foreign key* (specifying a column referencing a column in another "foreign" table, where all values must have a match). Note that `dbAddKey` can build keys on multiple columns (or refer to foreign keys on multiple columns) using the form `colname = c("id1","id2","id3")`. Since we applied a new sequential ID column when we inserted `meuse`, we can now designate it as a primary key:

```
> dbAddKey(conn, c("rpostgis_demo", "meuse"), "meuse_id", type = "primary")

Query executed:

ALTER TABLE "rpostgis_demo"."meuse" ADD PRIMARY KEY ("meuse_id");

[1] TRUE
```

Create an index: `dbIndex`

```
> dbIndex(conn, name, colname, idxname, unique = FALSE, method = c("btree",
+     "hash", "rtree", "gist"), display = TRUE, exec = TRUE)
```

Indexes are another important database element, as they can store information which helps queries run efficiently on data which are commonly referenced (e.g., using a foreign key), sorted (e.g., timestamps), or compared to other columns (e.g., geometries). As one would expect, they are especially important for large tables, and geometry types (e.g., polygons representing complex boundaries or lines), and are essential for to efficiently retrieve data with `pgGetGeom` and a `'WHERE'` clause. Note that, similarly to keys, `dbIndex` can build indexes on multiple columns. We can add an index on our point geometry using the preferred indexing method for `'GEOMETRY'` in PostGIS: GIST (generic index structure):

```
> dbIndex(conn, c("rpostgis_demo", "meuse"), "geom", method = "gist")

Query executed:

CREATE INDEX "meuse_geom_idx" ON "rpostgis_demo"."meuse" USING GIST ("geom");

--

[1] TRUE
```

Convert to timestamp: `dbAsDate`

```
> dbAsDate(conn, name, date = "date", tz = NULL, display = TRUE,
+     exec = TRUE)
```

The function `dbAsDate` allows us to convert a time stored as `'TEXT'` column into a `'TIMESTAMP WITH TIME ZONE'` type. We can demonstrate this with the `'example_time'` column, which we know was added in R to represent times in the "Europe/Amsterdam" time zone:

```
> dbAsDate(conn, c("rpostgis_demo", "meuse"), "example_time", tz = "Europe/Amsterdam")

Query executed:
```

```
ALTER TABLE "rpostgis_demo"."meuse"
    ALTER COLUMN "example_time" TYPE timestamptz
    USING
        "example_time"::timestamp AT TIME ZONE 'Europe/Amsterdam';

[1] TRUE
```

Garbage-collect and analyze a database (VACUUM): dbVacuum

```
> dbVacuum(conn, name, full = FALSE, verbose = FALSE, analyze = TRUE,
+     display = TRUE, exec = TRUE)
```

Vacuuming a table is a maintenance operation that cleans out unused space in the database structure, generally due to recent changes to a table (adding or deleting rows, updating data, etc.). Another operation (analyze) is often performed along with a vacuum, and is used to update statistics about the table, which PostgreSQL uses to optimize queries. Both vacuum and analyze are optional operations, but recommended for tables that are frequently modified, as they can greatly improve query speeds. Note that PostgreSQL automatically vacuums (and analyze) databases with the autovacuum daemon. It is generally preferable to tune the daemon for automatic use, but `dbVacuum` provides a way to run the operation manually. The default behavior for `dbVacuum` is to both vacuum and analyze a table:

```
> dbVacuum(conn, c("rpostgis_demo", "meuse"))

Query executed:

VACUUM ANALYZE "rpostgis_demo"."meuse";

[1] TRUE
```

Drop table/view/schema: dbDrop

```
> dbDrop(conn, name, type = c("table", "schema", "view", "materialized view"),
+     ifexists = FALSE, cascade = FALSE, display = TRUE, exec = TRUE)
```

The `dbDrop` function can be used to remove tables or schemas—just remember that this process is irreversible! Here we drop a table from the 'public' schema, which we created in previous examples:

```
> dbDrop(conn, "meuse_base")

Query executed:

DROP TABLE "public"."meuse_base";

[1] TRUE
```

Get information about table columns: (dbTableInfo)

```
> dbTableInfo(conn, name, allinfo = FALSE)
```

Finally, another "db*" function, `dbTableInfo`, returns a set of descriptive information about an existing database table, notably column names and types. This may be useful for reference prior to importing a table into R:

```
> dbTableInfo(conn, c("rpostgis_demo", "meuse"))

    column_name               data_type is_nullable
1     meuse_id                 integer          NO
2      cadmium        double precision         YES
3       copper        double precision         YES
4         lead        double precision         YES
5         zinc        double precision         YES
6         elev        double precision         YES
7           om        double precision         YES
8        ffreq                    text         YES
9         soil                    text         YES
```

```
10         lime                      text         YES
11      landuse                      text         YES
12       dist_m         double precision         YES
13 example_time timestamp with time zone         YES
14         geom             USER-DEFINED         YES
   character_maximum_length
1                        NA
2                        NA
3                        NA
4                        NA
5                        NA
6                        NA
7                        NA
8                        NA
9                        NA
10                       NA
11                       NA
12                       NA
13                       NA
14                       NA
```

Summary

Linking R with data storage services such as relational databases has become essential for sound data and project management. However, some users may be hesitant to adopt them for their projects, as databases (and SQL) can present a significant learning curve, translating data between R and databases is not always straightforward, or users may simply be unaware of free and open-source software such as PostgreSQL. In this paper, we presented **rpostgis**, an extension of the package **RPostgreSQL**, which we hope will provide a gateway to databases for current users of R, especially those working with spatial datasets. To summarize, **rpostgis** provides methods for transferring spatial data (vector and raster) between R and PostgreSQL/PostGIS, introduces a "data frame mode" for storing attributes of `"data.frame"` columns, and provides "SQL wrapper" functions for database management and maintenance, which can be also useful as learning tools for users unfamiliar with SQL and databases.

While we consider the base functionalities of **rpostgis** to be complete, we do foresee further development and evolution in several areas, and welcome collaboration to develop these. Providing a user-friendly R interface to the large library of PostGIS functions would be one area of great usefulness for spatial analysis. The "data frame mode" implementation is another area where progress could be made: for example, handling of R type attributes beyond `"factor"` and `"POSIX*"` types could be implemented in a modular fashion in `dbWriteDataFrame` and `dbReadDataFrame`, and requests and contributions are welcome. Likewise, more specific translations between R data types and PostgreSQL data types would be helpful improvement on current methods (e.g., translating R `"integer"` or `"numeric"` types to any of 10 PostgreSQL numeric data types); note that this work may be better suited for a generic package such as **RPostgreSQL**.

We also encourage users working with customized data classes for potentially large data sets in R to consider writing translations to a corresponding database model. For example, a related "extension" package of **rpostgis** is **rpostgisLT** (Dukai et al., 2017). This package specifically translates animal trajectory data objects stored as `"ltraj"` from the package **adehabitatLT** (Calenge, 2006) into a customized data model (`"pgtraj"`) built and managed by **rpostgisLT**. This not only provides a useful storage option for potentially large animal tracking datasets; it also opens up the world of PostgreSQL/PostGIS to scientists working with trajectory datasets, who may previously have only used R and/or desktop GIS.

These solutions also allow R users to better share data with other users or applications, since databases provide a common data location and format for interoperabilty with other software. While it is beyond the scope of this paper, there are many other attractive aspects of databases (functions/triggers, automated backup and restore, multi-user control, user-friendly data sharing and visualizations through front-end server applications) which not only can make R users more efficient, but help broaden the reach of their projects. Finally, for scientists interested in reproducible science and research, databases provide a framework to script and automate a variety of data cleaning, summary, and analysis processes, as well as a stable, long-term storage solution.

To conclude this paper, we can let our example database speak, to show all GEOMETRY and RASTER columns created in the examples, and finally close the database connection by using `RPostgreSQL::dbDisconnect`, as it is good practice to discard all pending work and free up database

resources for other users:

```
> pgListGeom(conn)

    schema_name table_name geom_column geometry_type     type
1        public      meuse        geom         POINT GEOMETRY
2        public meuse_buff        geom      GEOMETRY GEOMETRY
3        public   meuse_df        geom         POINT GEOMETRY
4 rpostgis_demo      meuse        geom         POINT GEOMETRY

> pgListRast(conn)

  schema_name table_name raster_column
1      public meuse_rast          rast
2      public      rlogo          rast

> RPostgreSQL::dbDisconnect(conn)

[1] TRUE
```

Bibliography

R. Bivand and N. Lewin-Koh. *maptools: Tools for Reading and Handling Spatial Objects*, 2017. URL https://CRAN.R-project.org/package=maptools. R package version 0.9-2. [p251]

R. Bivand and C. Rundel. *rgeos: Interface to Geometry Engine - Open Source (GEOS)*, 2017. URL https://CRAN.R-project.org/package=rgeos. R package version 0.3-23. [p253]

R. Bivand, T. Keitt, and B. Rowlingson. *rgdal: Bindings for the Geospatial Data Abstraction Library*, 2017. URL https://CRAN.R-project.org/package=rgdal. R package version 1.2-10. [p251]

C. Calenge. The package adehabitat for the R software: tool for the analysis of space and habitat use by animals. *Ecological Modelling*, 197:1035, 2006. URL https://doi.org/10.1016/j.ecolmodel.2006.03.017. [p266]

J. Conway, D. Eddelbuettel, T. Nishiyama, S. K. Prayaga, and N. Tiffin. *RPostgreSQL: R interface to the PostgreSQL database system*, 2017. URL https://CRAN.R-project.org/package=RPostgreSQL. R package version 0.6-2. [p251]

B. Dukai, M. Basille, D. Bucklin, and C. Calenge. *rpostgisLT: Managing Animal Movement Data with 'PostGIS' and R*, 2017. URL https://CRAN.R-project.org/package=rpostgisLT. R package version 0.5.0. [p266]

ESRI. ArcGIS desktop, 2017. URL https://www.arcgis.com. [p252]

R. J. Hijmans. *raster: Geographic Data Analysis and Modeling*, 2016. URL https://CRAN.R-project.org/package=raster. R package version 2.5-8. [p251]

E. Pebesma. *sf: Simple Features for R*, 2017. URL https://CRAN.R-project.org/package=sf. R package version 0.5-5. [p253]

E. J. Pebesma and R. S. Bivand. Classes and methods for spatial data in R. *R News*, 5(2):9–13, November 2005. URL https://CRAN.R-project.org/doc/Rnews/. [p253]

PostGIS Project Steering Committee. PostGIS, 2017. URL https://postgis.net. [p251]

QGIS Development Team. *QGIS Geographic Information System*. Open Source Geospatial Foundation, 2017. URL http://qgis.osgeo.org. [p252]

R Special Interest Group on Databases (R-SIG-DB), H. Wickham, and K. Müller. *DBI: R Database Interface*, 2017. URL https://CRAN.R-project.org/package=DBI. R package version 0.7. [p251]

The PostgreSQL Global Development Group. PostgreSQL, 2017. URL https://www.postgresql.org. [p251]

TIBCO Software Inc. *wkb: Convert Between Spatial Objects and Well-Known Binary Geometry*, 2016. URL https://CRAN.R-project.org/package=wkb. R package version 0.3-0. [p253]

David Bucklin
Fort Lauderdale Research and Education Center
Department of Wildlife Ecology and Conservation
Institute of Food and Agricultural Sciences
University of Florida
3205 College Avenue
Davie, FL 33314, USA
david.bucklin@gmail.com

Mathieu Basille
Fort Lauderdale Research and Education Center
Department of Wildlife Ecology and Conservation
Institute of Food and Agricultural Sciences
University of Florida
3205 College Avenue
Davie, FL 33314, USA
ORCiD: https://orcid.org/0000-0001-9366-7127
basille@ufl.edu

lba: An R Package for Latent Budget Analysis

by Enio G. Jelihovschi and Ivan Bezerra Allaman

Abstract The latent budget model is a mixture model for compositional data sets in which the entries, a contingency table, may be either realizations from a product multinomial distribution or distribution free. Based on this model, the latent budget analysis considers the interactions of two variables; the explanatory (row) and the response (column) variables. The package **lba** uses expectation-maximization and active constraints method (ACM) to carry out, respectively, the maximum likelihood and the least squares estimation of the model parameters. It contains three main functions, `lba` which performs the analysis, `goodnessfit` for model selection and goodness of fit and the plotting functions `plotcorr` and `plotlba` used as a help in the interpretation of the results.

Introduction

The idea of latent budget was first proposed by Goodman (1974) in which he wanted to show a method to analyse the relationship between a set of qualitative variables, when some of them are manifested variables and other are non observable or latent variables. These ideas were later elaborated by Clogg (1981) by interpreting a simple latent class model in an asymmetric way. Independently, de Leeuw and van der Heijden (1988) introduced the model and named it latent budget analysis because they used it to analyse time-budget data. The model was also introduced independently in geology by Renner (1988), where it is known as the endmember model.

LBA is an analysis method for compositional data which is basically an $I \times J$ matrix where the row variable with I categories is called the explanatory variable and the column variable with J categories is called the response variable. In compositional data each row is considered a J dimensional vector of conditional probabilities so that, for every row, they add up to one. LBA is used to understand the relationship between those two variables.

LBA allows us to find out which categories of the response variables are related to different groups of the explanatory categories. If the table has a product multinomial distribution we can understand the latent budget model (LBM) as explaining the relationship between the explanatory and the response variables. It is done by assuming that conditioned on the latent variable they are independent. In that sense, the latent budgets, which are categories of a latent variable, are hidden values which explain the relationship between the explanatory and response variables. LBA reduces the dimensionality of the original problem, thus making it easier to understand its hidden relations.

Examples of latent budget models in sociological research, political sciences and other areas where categorical variables are used can be found in Van der Ark (1999a). Generalizations of the LBA method may be found in Siciliano and Heijden (1994), Siciliano and Mooijaart (2001) and Aria (2008). However, we found few articles in our literature search that used LBA in their data analyses. Larrosa (2005) shows how to use LBA in applications to economics. Tambrea and Siciliano (1999) proposed an LBA approach for three-way tables in a business field. We can also cite Aquilia et al. (2015) in geology, Ros-Freixedes and Estany (2014) in biology, and Aria et al. (2003) in food engineering.

In our point of view, the reason why the use of LBA is not more widespread is due to the lack of available software. The software "A freeware computer program to perform latent budget analysis" written by L. Andries van der Ark (Van der Ark, 1999b) in Borland Pascal 7.0 only runs under MS-DOS. Unfortunately, it is no longer available at `http://come.to/lba/software`. The only software we could find that performs LBA analysis is CoDaPack `http://ima.udg.edu/codapack/`, which until recently only ran in the Windows operational system. Therefore, because of the importance LBA has in categorical data analysis, the authors decided to write the **lba** package.

This is the first package for this type of analysis in R. The package is available from both the Comprehensive R Archive Network at `http://cran.r-project.org/web/packages/lba/index.html` and the **lba** project web site at `https://github.com/ivanalaman/lba`.

Latent budget model

LBM is a mixture model for compositional data. A row of compositional data is called a *composition* or a *budget* and its elements are the *components*. We will follow the notation of Van der Ark (1999a). He says: "by performing LBA we approximate I observed budgets, which may represent persons, groups or objects by a small number of latent budgets, consisting of typical characteristics of the sample." See

also de Leeuw et al. (1990) and van der Heijden et al. (1992).

Terminology and model definition

The original contingency table is $N(I, J)$. Let us also define:

- row total: $n_{i+} = \sum_j n_{ij}$.
- column total: $n_{+j} = \sum_i n_{ij}$.
- total: $n = \sum_j \sum_i n_{ij}$.

The compositional data matrix $\mathbf{P}$ is formed by dividing the raw data by their corresponding row total. Let us call the observed components $p_{i|j} (i = 1, \ldots, I; j = 1, \ldots, J)$ then, $p_{i|j} = \frac{n_{ij}}{n_{i+}}$, $p_{i+} = \frac{n_{i+}}{n}$ and $p_{+j} = \frac{n_{+j}}{n}$. Each row vector $\mathbf{p}_i$ of $\mathbf{P}$ is called an observed budget and is approximated by the expected budget $\boldsymbol{\pi}_i$ which is a mixture of K, $(K \leq min(I, J))$ latent budgets.

The row vectors $\pi_i, (i = 1, \ldots, I)$ form the expected matrix π which has a lower rank and, in LBM, approximates $\mathbf{P}$.

The latent budgets are represented by $\boldsymbol{\beta}_k, (k = 1, \ldots, K)$ and the model is written as

$$\boldsymbol{\pi}_i = \alpha_{1|i}\boldsymbol{\beta}_1 + \ldots + \alpha_{k|i}\boldsymbol{\beta}_k + \ldots + \alpha_{K|i}\boldsymbol{\beta}_K,$$

where $\alpha_{k|i}$ are the mixing parameters.

The elements of $\boldsymbol{\pi}$ are $\pi_{j|i}$ and are called *expected components*. The elements of $\boldsymbol{\beta}_k$, $\beta_{j|k}$ are called *latent components*. In scalar notation, $\pi_{j|i} = \sum_{k=1}^{K} \alpha_{k|i}\beta_{j|k}$, and in matrix notation $\Pi = \mathbf{A}\mathbf{B}^\top$ where Π is an $I \times J$ matrix whose rows are the expected budgets. $\mathbf{A}$ is an $I \times K$ matrix of mixing parameters and $\mathbf{B}$ is a $J \times K$ matrix whose columns are the latent budgets. LBM(K) is then the latent budget model with K latent budgets. Similar to the observed components, the parameters of LBM are subject to the sum constraints $\sum_{j=1}^{J} \pi_{j|i} = \sum_{k=1}^{K} \alpha_{k|i} = \sum_{j=1}^{J} \beta_{j|k} = 1$ and the non-negativity constraints $0 \leq \pi_{j|i}, \alpha_{k|i}, \beta_{j|k} \leq 1$. In this way, all parameters are proportions that further facilitate the interpretation of the model.

Quoting Van der Ark (1999a)

> The latent budgets can be characterized by being compared to the latent budgets of LBM(1). LBM(1) is the independence model with $\alpha_{1|i} = 1$ and $\beta_{j|1} = p_{+j}$, in this case $\boldsymbol{\pi}_i = \boldsymbol{\beta}_1$. Hence, if latent component $\beta_{j|k} \geq p_{+j}$, then $\boldsymbol{\beta}_k$ is characterized by the j-th category. On the other hand, if $\beta_{j|k} \leq p_{+j}$, then the j-th category is of lesser importance. The relative importance of each latent budget, in terms of how much of the expected data they account for, is expressed by the budget proportions $\pi_k = \sum_i p_{i+}\alpha_{k|i}$.
>
> The π_k parameter also denotes the probability of latent budget k when there is no information about the level of the row variable. To understand how the expected budgets are constructed to form the latent budgets, we must compare the mixing parameters to π_k. If $\alpha_{k|i} \geq \pi_k$ then the expected budget $\boldsymbol{\pi}$ is characterized more than average by latent budget $\boldsymbol{\beta}_k$, otherwise, if $\alpha_{k|i} \leq \pi_k$ then the expected budget $\boldsymbol{\pi}$ is characterized less than average by latent budget $\boldsymbol{\beta}_k$. In practice, the mixture model interpretation is easier to carry out when we first characterize the latent budgets and then interpret the expected budgets in terms of them.

Compositional data that follows the product multinomial sampling scheme may be estimated by the maximum likelihood estimation method (MLE) which is estimated by using the EM algorithm (Dempster et al., 1977). On the other hand, if the data cannot be assumed to follow that distribution, using MLE is not recommended. Following Van der Ark (1999a) we opted to estimate by using weighted least squares (WLS) estimators since it is a distribution free method, that is, one which does not assume any probability configuration on the data, see Mooijaart et al. (1999).

Unconstrained parameter estimation: maximum likelihood estimator (MLE)

In de Leeuw et al. (1990) they describe the MLE for compositional data under a product-multinomial distribution. The log likelihood is:

$$\ln L(\pi_{j|i}; p_{j|i}, n_{ij}) = \sum_{i=1}^{I} n_{i+} \sum_{j=1}^{J} p_{j|i} \ln(\pi_{j|i}) + C. \tag{1}$$

In this case, $\pi_{j|i}$ is dependent on a latent variable with K categories, where the observations on that latent variable are missing. Therefore the complete data loglikelihood function is:

$$\ln L(\pi_{ijk}; n_{++}) = \sum_{i=1}^{I}\sum_{j=1}^{J}\sum_{k=1}^{K} n_{ijk} \ln\left(\frac{\pi_{ijk}}{\pi_{i+}}\right) + C. \tag{2}$$

The π_{ijk} parameters are the unknown joint probabilities of the two manifest variables and the latent variable where $\sum_{i=1}^{I}\sum_{j=1}^{J}\sum_{k=1}^{K}\pi_{ijk} = 1$ and $\frac{\pi_{ijk}}{\pi_{i+}} = \pi_{jk|i} = \alpha_{K|i}\beta_{j|k}$. n_{ijk} are the unknown joint frequencies of the three variables where $n_{ijk} = \pi_{ijk}n_{++}$. In that paper, de Leeuw et al. (1990) use an EM-algorithm to maximize the complete data loglikelihood.

The EM algorithm proceeds iteratively. Beginning with arbitrary initial values of $\alpha_{k|i}$ and $\beta_{j|k}$, which may be set either by the user or randomly generated by the package function, and labelling them $\hat{\alpha}^0_{k|i}$ and $\hat{\beta}^0_{j|k} \forall i, j, k$ to meet the sum and the no-negativity constraints. In the expectation (E) step, calculate $\hat{\pi}^0_{ijk}$ from the initial estimates and find the estimator for n_{ijk} which is:

$$\hat{n}^0_{ijk} = n_{ij}\frac{\hat{\pi}^0_{ijk}}{\sum_k \hat{\pi}^0_{ijk}}. \tag{3}$$

In the maximization (M) step, given $\hat{\pi}^{new}_{ijk}$ and $\hat{n}^{new}_{ijk}$, the complete data loglikelihood is maximized. In de Leeuw et al. (1990) they show that this yields the following estimates for the next iteration:

$$\hat{\alpha}^{new}_{k|i} = \frac{\sum_j \hat{n}^0_{ijk}}{\sum_{j,k}\hat{n}^0_{ijk}} = \frac{\hat{n}^0_{i+k}}{\hat{n}^0_{i++}} \tag{4}$$

and

$$\hat{\beta}^{new}_{j|k} = \frac{\sum_i \hat{n}^0_{ijk}}{\sum_{i,j}\hat{n}^0_{ijk}} = \frac{\hat{n}^0_{+jk}}{\hat{n}^0_{++k}}. \tag{5}$$

Unconstrained parameter estimation: weighted least squares (WLS)

The WLS function to be minimized is:

$$L_{\alpha_{k|i},\beta_{j|k}} = \sum_{i,j}(v_i w_j)(p_{j|i} - \sum_k \alpha_{k|i}\beta_{j|k})^2. \tag{6}$$

The weights $v_i w_j$ may be chosen freely. If they are chosen to equal one, the WLS becomes the ordinary least squares (OLS).

The **lba** package follows the algorithm completely described in Van der Ark (1999a) and Mooijaart et al. (1999) called the active constraints method (ACM). The method is a minimization with constraints. The equality constraints are the row sums of matrix **A** and column sums of matrix **B**, and the inequality constraints are the values of all elements of **A** and which must be greater than zero.

Identifiability: $K = 2$

In general, the LBM is not identifiable, meaning that there could be various sets of parameters yielding the same goodness of fit (van der Ark et al., 1999). In fact, as de Leeuw et al. (1990) show, $\mathbf{\Pi} = \mathbf{A}\mathbf{B}^\top = \mathbf{A}\mathbf{T}^{-1}\mathbf{T}\mathbf{B}^\top = \mathbf{A}^*\mathbf{B}^{*\top}$ where $\mathbf{A}^* = \mathbf{A}\mathbf{T}^{-1}$ and $\mathbf{B}^{*\top} = \mathbf{T}\mathbf{B}^\top$.

T is a matrix of order $K \times K$ with elements τ_{cd}. To ensure that the rows of $\mathbf{A}^*$ and the columns of $\mathbf{B}^*$ add up to one, de Leeuw et al. (1990) proved that **T** is subject to the constraints that all elements of **T** must be nonnegative and

$$\sum_d \tau_{cd}, \quad c = 1, \ldots, K.$$

We follow van der Ark et al. (1999) in which he shows that **T** can be chosen such that the latent budget solution is optimal in a specific sense.

The **lba** package follows the solutions proposed by de Leeuw et al. (1990) for LBM(2), as described in Van der Ark (1999a) Chapter 2, when discussing the geometry of LBM(2).

In LBM(2), the unidentified latent budgets β_1 and β_2 can be viewed as two vectors

in a J-dimensional space. The heads of any two vectors can be connected by a line segment, denoted by V, which is a subset of a line S. The expected budgets $\pi_1, \ldots, \pi_I$ are J-dimensional vectors and convex combinations of β_1 and β_2. Therefore, the heads of $\pi_1, \ldots, \pi_I$ lie on V, and the relative distance from $\pi_1, \ldots, \pi_I$ to β_1 and β_2 is expressed by the mixing parameters. The unidentified latent budgets β_1 and β_2, collected in **B**, can be transformed into $\mathbf{B}^*$.

The region of budgets is denoted by **U**. The vectors that bound **U** are called outer extreme budgets, and have one component equal to zero. LBM(2) always has two outer extreme budgets. Not every $b \in \mathbf{U}$ is a feasible latent budget. A latent budget cannot lie between two expected budgets, because this would result in negative mixing parameters. Hence a latent budget cannot lie within the space spanned by the expected budgets that take the most extreme position on **S**. This space is denoted by **W**. The most extreme expected budgets are called inner extreme budgets.

The matrices **T** to get the transformations into $\mathbf{B}^*$ are found in both the outer extreme solution and the inner extreme solution. They are used to get the respective $\mathbf{A}^*$ matrices.

In the outer extreme solution the latent budgets are as different as possible, simplifying their interpretation in most cases. In the inner extreme solution, the latent budgets are as similar as possible. At the same time, the mixing parameters will be as different as possible.

Identifiability: $K \geq 3$

Van der Ark (1999a) uses the following criteria to identify the solutions: minimize $\sum_{q=1}^{Q} \delta_{\chi^2_q}$ for an identified inner extreme solution and maximize $1/\sum_{q=1}^{Q} \delta_{\chi^2_q}$ for an identified outer extreme solution.

Where $Q = \binom{n}{k}$, i.e., the number of distances among the K latent budgets, and

$$\delta_{\chi^2} = \sqrt{\sum_{j=1}^{J} \frac{(p_{j|i} - p_{j|i'})}{p_{j+}}}.$$

In order to find those minimal solutions, the **lba** package uses the `constrOptim.nl` function from the **alabama** package (Varadhan, 2015). In this case the "BFGS" algorithm is used .

Constrained parameter estimation

Parameters in LBM may be subject to optional constraints, which can be imposed by a researcher, either to test specific hypotheses about the model to facilitate its interpretation, or to build complex models.

There are three different types of optional constraints, namely *fixed value constraints*, *equality constraints*, and *multinomial logit constraints*.

Fixed value constraints have the form $\alpha_{k|i} = c$ or $\beta_{j|k} = c'$, where $0 \leq c \leq 1$ and $0 \leq c' \leq 1$ are constants.

Equality constraints have the form $\alpha_{k_1|i_1} = \alpha_{k_2|i_2} = \ldots = \alpha_{k_L|i_L}$ when equalities are placed on the mixing parameters and $\beta_{j_1|k_1} = \beta_{j_2|k_2} = \ldots = \beta_{j_M|k_M}$ when equalities are placed on the latent components.

The multinomial logit constraints were introduced in LBA by van der Heijden et al. (1992), and have the following form:

$$\alpha_{k|i} = \frac{\exp(\sum_{s=1}^{S} x_{is}\gamma_{sk})}{\sum_{n=1}^{K} \exp(\sum_{s=1}^{S} x_{is}\gamma_{sn})},$$

for the mixing parameters, where γ_{sk} are the multinomial logit parameters; and

$$\beta_{j|k} = \frac{\exp(\sum_{t=1}^{T} y_{jt}\psi_{tk})}{\sum_{n=1}^{J} \exp(\sum_{t=1}^{T} y_{nt}\psi_{tk})},$$

for the latent components, where ψ_{tk} are the multinomial logit parameters. For a detailed discussion see Van der Ark (1999a) Chapter 3.

The S variables that contain the additional information about the mixing parameters, are called *row covariates* and the $I \times S$ matrix **X** is called the *row design matrix*. Similarly, the T variables that contain the additional information about the latent components, are called *column covariates* and the

$J \times T$ matrix $\mathbf{Y}$ is called the *column design matrix*. Both the sum constraints and the non-negativity constraints are satisfied by the multinomial logit constraints.

The degrees of freedom, according to de Leeuw et al. (1990), is the number of independent cells minus the number of independent parameters, For compositional data the number of independent cells is always $I(J-1)$ due to the sum constraints on the observed budgets. For the unconstrained LBM(K), we have $I(K-1)$ free mixing parameters, and $K(J-1)$ free latent components. However, the model is not identifiable and $K(K-1)$ parameters should be fixed. Hence the number of degrees of freedom is $I(J-1) - I(K-1) - K(J-1) + K(K-1) = (I-K)(J-K)$.

The **lba** package only runs the identifiability function when there are no optional constraints in the model. This is due to the fact that it is not possible to maintain the constraints while running that function. Therefore users who use optional constraints should use $K(K-1)$ fixed parameters or an adequate number of other constraints in order to attain identifiability.

The maximum likelihood estimation is adjusted for fixed value constraint according to van der Heijden et al. (1992).

Let $\alpha_{l|r} = c$, then the new adjusted value of Equation 4 for the free mixing parameters is:

$$\hat{\alpha}_{k|i}^{new} = \frac{\sum_j \hat{n}_{ijk}^0}{\sum_{j,k\neq l} \hat{n}_{ijk}^0} = \frac{\hat{n}_{i+k}^0}{(\hat{n}_{i++}^0 - \hat{n}_{i+l}^0)}$$

and, if $\beta_{s|r} = c'$, then the new adjusted value of Equation 5 for the free latent components is:

$$\hat{\beta}_{j|k}^{new} = \frac{\sum_i \hat{n}_{ijk}^0}{\sum_{i,j\neq s} \hat{n}_{ijk}^0} = \frac{\hat{n}_{+jk}^0}{(\hat{n}_{++k}^0 - \hat{n}_{+sk}^0)}.$$

Optional equality constraints for parameter estimates obtained with the EM algorithm are described in Mooijaart and van der Heijden (1992); see also van der Heijden et al. (1992). For the mixing parameters, if $\alpha_{k|i} = \alpha_{k'|i'}$ the new adjusted values of Equation 4 are:

$$\hat{\alpha}_{k|i}^{new} = \hat{\alpha}_{k'|i'}^{new} = \frac{(\hat{n}_{i+k}^0 + \hat{n}_{i'+k'}^0)}{(\hat{n}_{i++}^0 + \hat{n}_{i'++}^0)}$$

and, for $\beta_{j|k} = \beta_{j'|k'}$, the new adjusted values of Equation 5 are:

$$\hat{\beta}_{j|k}^{new} = \hat{\beta}_{j'|k'}^{new} = \frac{(\hat{n}_{+jk}^0 + \hat{n}_{+j'k'}^0)}{(\hat{n}_{+j+}^0 + \hat{n}_{+j'+}^0)}.$$

The remaining parameters should be updated by using quations 4 and 5.

In van der Heijden et al. (1992) they warn that the estimation of optional equality constraints (in combination with fixed value constraints) by the EM-algorithm is not always correct. The **lba** package takes it into account automatically and uses the **alabama** package to estimate the parameters when necessary.

The estimation of the parameters under multinomial logit constraints is described in van der Heijden et al. (1992). The complete data log likelihood function can be split into two parts where one depends only on row covariates and the other only on the column covariates; therefore, the E-step of the EM algorithm is the same as in the unconstrained LBM. The M-step is implemented by making use of the `optim` function and the **alabama** package.

Depending on the values of the matrices $\mathbf{X}$ and $\mathbf{Y}$, the exponential values of the row and column covariates might become infinity. They are replaced by `1e6`. Also, whenever the values of the row or column covariates are not supplied, **lba** creates random values from the standard normal distribution.

Note:

- Depending on the starting parameters, all algorithms cited above may only locate a local, rather than global, maximum or minimum. This becomes more and more of a problem as K, the number of latent budgets, increases. It is therefore highly advisable to run **lba** a few times until you are relatively certain that you have located the global maximum log-likelihood, the global minimum least squares, or the identification minimization.
- Some times it a label switching may occur. Usually the interpretation remains the same but the label of the budgets might not be the same. The **lba** package does try to minimize those occurrences, nevertheless they still may occur.

Model selection and goodness of fit criteria

Latent budget analysis has a great variety of tools available for assessing model fit and determining an appropriate number of latent budgets K for a given data set. In some applications, the number of latent budgets will be selected for primarily theoretical reasons. In other cases, however, the analysis may be of a more exploratory nature, with the objective being to locate the best fitting or most parsimonious model. The researcher may then begin by fitting an independence LBM(1), and then iteratively increasing the number of latent budgets one by one until a suitable fit has been achieved.

Adding an additional budget to a latent budget model will increase the fit of the model, but at the risk of fitting too much noise, and at the expense of estimating further $I + J$ model parameters. Parsimony criteria seeks to strike a balance between over and under-fitting the model to the data by penalizing the log-likelihood for a function of the number of parameters being estimated. Usually, the researcher must take into account that parsimony is the best help in order to achieve a good interpretation of the model, that means a close resemblance between the observed and expected data, with as few parameters as possible. See de Leeuw and van der Heijden (1988), de Leeuw et al. (1990), and Van der Ark (1999a).

The most used criteria can be found in the Table 1.

Statistics	Formula
Likelihood ratio statistic	$G^2 = 2\sum_{i,j} n_{ij} \log\left(\frac{n_{ij}}{\pi_{j\|i} n_{i+}}\right)$
Pearson chi-squared statistic	$X^2 = \sum_{i,j}\left(\frac{(n_{ij}-\pi_{j\|i} n_{i+})^2}{\pi_{j\|i} n_{i+}}\right)$
Residual sum of squares	$RSS = \sum_{i,j}(\pi_{j\|i} - p_{j\|i})^2$
Weighted residual sum of squares	$wRSS = \sum_{i,j} v_i^2 w_j^2 (\pi_{j\|i} - p_{j\|i})^2$
Akaike information criterion	$AIC = G^2 - 2df$
Corrected AIC	$CAIC = G^2 - df \times [\log(N) + 1]$
Bayesian information criterion	$BIC = G^2 - df \times \log(N)$

Table 1: Goodness of fit calculated by the **lba** package.

RSS and $wRSS$ must be compared to the RSS's of other models in order to be meaningful.

The **lba** package calculates a great variety of goodness of fit statistics (GFS). Some can only be used with the data following the product multinomial distribution; others, on the other hand, may be used with distribution free data. A few have an asymptotic chi-square distribution, called exact GFS, but most have an unknown distribution.

Computation time as a function of the data matrix dimension and the number of latent budgets

Every numerical method performing an estimation of many parameters which depends on a optimization algorithm is going to be computationally time consuming. The **lba** package in particular uses a maximization with constraints in order to identify the parameters being estimated. The **alabama** package is used in this case and furthermore the restrictions, which are row sums of matrix A and column sums of matrix B must be one, is programmed in **alabama** in a very complex way. All those conditions tend to make **alabama** somewhat computationally time consuming, nevertheless, in our experience, those times are not too long. Figure 1 shows that the most important time consuming parameter is the number of latent budgets, K, the size of the data matrix is not very significant in increasing computation time while keeping K constant.

The lba package

The main function of package **lba** is `lba`. This function input may be an object of class `"formula"`, `"matrix"` or `"table"`. The `lba` function can be called by:

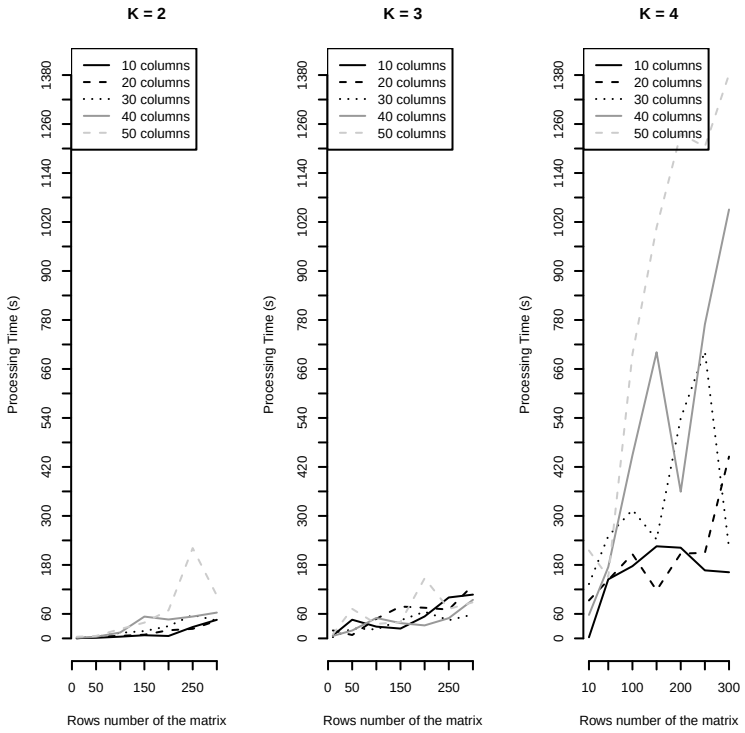


Figure 1: Computation time as a function of the data matrix dimensions and the number of latent budgets.

`lba(obj, ...)`

If the object is from class `"formula"`, the method `lba.formula` will be called:

```
lba(formula, data, A = NULL, B = NULL, K = 1L, cA = NULL,
    cB = NULL, logitA = NULL, logitB = NULL, omsk = NULL, psitk = NULL,
    S = NULL, T = NULL, row.weights = NULL, col.weights = NULL,
    tolG = 1e-10, tolA = 1e-05, tolB = 1e-05, itmax.unide = 1000,
    itmax.ide = 1000, trace.lba = TRUE, toltype = "all", method = c("ls",
        "mle"), what = c("inner", "outer"), ...)
```

Objects of class `"formula"` follow the same logic of linear models, that is, dependent variables as a function of independent variables. The argument `data` must have objects of class `"data.frame"` as input. Objects of class `matrix` are executed by method `lba.matrix`:

```
lba.matrix(obj, A = NULL, B = NULL, K = 1L, cA = NULL, cB = NULL,
    logitA = NULL, logitB = NULL, omsk = NULL, psitk = NULL,
    S = NULL, T = NULL, row.weights = NULL, col.weights = NULL,
    tolG = 1e-10, tolA = 1e-05, tolB = 1e-05, itmax.unide = 1000,
    itmax.ide = 1000, trace.lba = TRUE, toltype = "all", method = c("ls",
        "mle"), what = c("inner", "outer"), ...)
```

Objects of class `"table"` are executed by the method `lba.table`:

```
lba.table(obj, A = NULL, B = NULL, K = 1L, cA = NULL, cB = NULL,
    logitA = NULL, logitB = NULL, omsk = NULL, psitk = NULL,
    S = NULL, T = NULL, row.weights = NULL, col.weights = NULL,
    tolG = 1e-10, tolA = 1e-05, tolB = 1e-05, itmax.unide = 1000,
    itmax.ide = 1000, trace.lba = TRUE, toltype = "all", method = c("ls",
        "mle"), what = c("inner", "outer"), ...)
```

The default method of estimation is *weighted least squares* with the row weights ($\sqrt{n_{i+}/n_{++}}$) and column weights ($1/\sqrt{n_{j+}/n_{++}}$). The user who wants to use *weighted least squares* differently from the default should give values to either one or both parameters `row.weights` and `col.weights`. If all those

values equal one, then we get the *ordinary least squares*. The other available method is the *maximum likelihood estimator*.

The arguments `A` and `B` are used whenever the user wants to set the initial values of the *mixing parameters* or *latent components* respectively. If the user has no initial values to set, those matrices are randomly set by using a *Dirichlet* distribution. For matrix `A` the distribution parameters are I and *alphavec* where I is the row number of the compositional data matrix and *alphavec* is randomly generated from a uniform distribution with parameter K *(number of latent budgets)* as for `B` the parameters are K and *alphavec* which is randomly generated from a uniform distribution with parameter J where J is the column number of the compositional data matrix.

The arguments `cA, cB` must be used whenever the estimation process is done with constraints on the parameters α or β respectively. For fixed value constraints, they must give values between zero and one. For equality constraints, they must be integers greater or equal to two where the parameters with the same value will be considered equal.

Use `help(lba)` for the remaining parameters.

The **lba** package can produce plots of the mixing parameters and latent components matrices. For $K = 3$, **lba** performs two different kinds of plots. One is the triangular (or ternary) coordinate system, suggested by Van der Ark (1999a) and de Leeuw et al. (1990), the other is the correspondence analysis (CA) plot suggested by Jelihovschi et al. (2011), which can also be made for any $K \geq 3$. It is important to note that the CA plots are applied to the indentified mixing parameters and identified latent components matrices; in **lba** they are respectively the matrices `Aoi` and `Boi`. For $K = 2$, **lba** does, as suggested by Van der Ark (1999a) page 41, also use the correspondence analysis result. The function which creates the correspondence analysis plot is called `plotcorr`, the other one is called `plotlba`.

It should be noted that when plotting the latent components using the triangular coordinate system, **lba** uses the rescaled latent components matrix whose values are: $\beta_{k|j} = \frac{\beta_{jlk}\pi_k}{p_{+j}}$. Note that in this case the row sums equal one, not column sums as in the latent components matrix.

The functions `plotlba` and `plotcorr` use the generic functions `plot`, `axis`, `text`, `points`, `segments` and `legend` for `K = 2`. For `K = 3`, the function `plotlba` uses the functions `triax.plot`, `triax.points` and `thigmophobe.labels` from package **plotrix** and also `segments` and `legend`. The function `plotcorr` uses the generic functions for `K = 2`. Whenever `K` $\geq$ `4`, only the function `plotcorr` is used. In this case the function `scatterplot3d` from package **scatterplot3d** is internally called. Finally, if the argument `rgl = TRUE` is used, then the function `plot3d` from package **rgl** will be called.

The goodness of fit results can be obtained by making use of the function `goodnessfit(obj,...)` where `obj` is an object of class `"lba"`.

Examples

Main et al. (2015) studied pregnancy related maternal deaths in California. They examined five distinct clinical conditions that account for nearly 70 of all pregnancy related deaths,

- cardiovascular diseases, CVD
- preeclampsia/eclampsia, Pre.E
- obstetric hemorrhage, OH
- deep vein thrombosis - pulmonary embolism, DVTPE
- amniotic fluid embolism, AFE

together, they also collected data about

- maternal age,
- parity,
- gestational age at delivery,
- maternal race and country of birth,
- body mass index - BMI.

which will be the rows or budgets of the data matrices.

Example 1; BMI

Table 2 shows the number of deaths related to the five conditions for women with BMI less than 30, between 30 and 40 and above 40.

The lba function was performed on data matrix `bmi`, as shown below.

	Pre.E	OH	CVD	DVTPE	AFE
<30b	29	14	28	8	18
30-40b	4	2	15	6	0
>40b	1	2	6	5	0

Table 2: Number of deaths related to the five conditions for women with BMI less than 30, between 30 and 40 and above 40

```
> library(lba)
> data(pregnancy)
> bmi <- pregnancy[5:7,]
> set.seed(1)
> bmilba <- lba(bmi, K = 2, method = "mle", what = 'outer',
+               trace.lba = FALSE)
```

Since all rows of the BMI matrix are independent, the product multinomial model can be used and the maximum likelihood estimation (MLE) method applies. We will also use $K = 2$ because the number of rows is 3 (and so, $K = 3$ is the saturated model). We used the function `set.seed` so that the user who wishes to replicate the analysis may get the same results as the ones shown below.

```
> goodnessfit(bmilba)
Likelihood ratio statistic:
         K budget Baseline
G2 value     1.809 2.95e+01
P-value      0.613 2.63e-04
```

The goodness of fit result shows that the likelihood ratio statistic (G2) used to test the model gave a p-value of 0.613 and thus accepting the model with $K = 2$. The `summary` of both `lba` and `goodnessfit` functions gives complete results. The only other possible model is the independence model, or baseline model, which has a p-value of 0.000263 and so, it is rejected. The interpretation of the model is easily done by using the correspondence analysis plot, which is created using the function `plotcorr`.

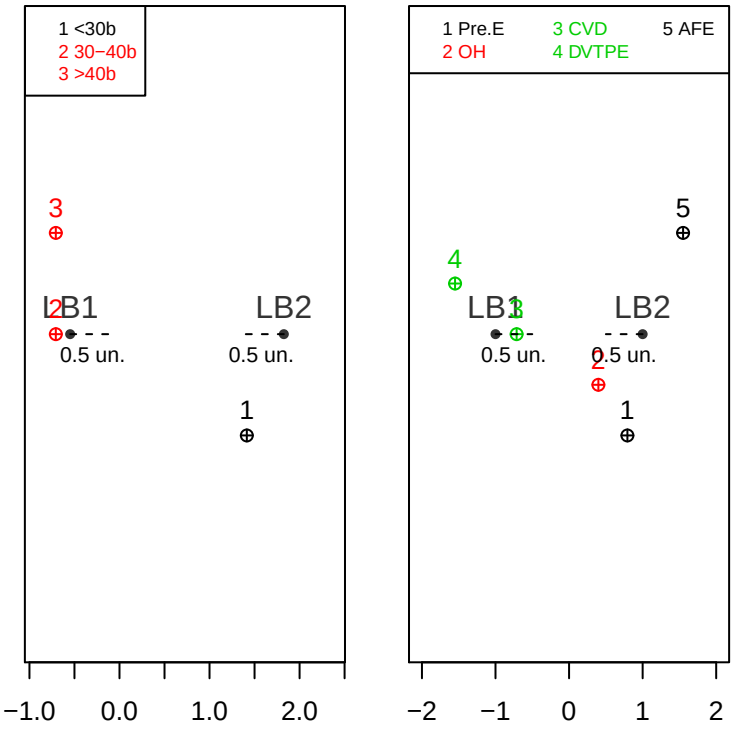


Figure 2: Example 1: Mixing parameters (left); latent components (right).

The plot has just one dimension, so that the points are spread only along the horizontal axis; see documentation of function `plotcorr`. The first latent budget (LB1) is composed of CVD and

DVTPE, which could be considered as pregnancy-related conditions; the second latent budget (LB2) is composed of AFE and Pre.E, which are more general conditions. The OH condition can be considered neutral. The mixing parameter BMI less than 30 is related to LB2 and the more obese women to LB1 as is expected since obese people are more affected by the general conditions.

Example 2; parity, maternal age, gestational age at delivery in weeks

In this example, we consider three other explanatory variables connected to pregnancy-related death: parity, maternal age and gestational age at delivery/fetal demise. The resulting data matrix is:

	Pre.E	OH	CVD	DVTPE	AFE
1	16	3	13	3	3
2-4	16	13	31	14	10
5+	4	4	5	3	5
<30a	12	5	25	11	4
30-40a	18	13	22	8	13
>40a	6	2	2	1	1
<32w	6	5	8	0	0
32-36w	16	5	8	8	1
>37w	14	10	33	12	17

Table 3: Relation between, parity, maternal age, gestational age at delivery with five cause of pregnancy related death

Unlike the Example 1, the matrix rows are not independent since the same women are counted in each one of the row variables. Therefore the MLE method does not apply here and we use the least squares method in order to estimate the model parameters. The functions to call are:

```
> mcd <- pregnancy[8:16,]
> set.seed(1)
> mcdlba  <- lba(mcd, K = 2, method = 'ls', what = 'outer', trace.lba = FALSE)
> set.seed(1)
> mcdlba1 <- lba(mcd, K = 3, method = 'ls', what = 'outer', trace.lba = FALSE)
> set.seed(1)
> mcdlba2 <- lba(mcd, K = 4, method = 'ls', what = 'outer', trace.lba = FALSE)
```

In order to get the values of Table 4, the function `goodnessfit` is used as follows.

```
> summary(goodnessfit(mcdlba))
> summary(goodnessfit(mcdlba1))
> summary(goodnessfit(mcdlba2))
```

Number of Latent budgets	df	*wRSS*	Actual decrease	Required decrease	Fit improved?
1	32	0.31	NA	NA	NA
2	21	0.14	0.17	0.11	yes
3	12	0.06	0.11	0.09	yes
4	05	0.02	0.04	0.07	no

Table 4: Goodness of fit of LBM using least squares and $K = 1, 2, 3$, and 4.

The weighted residual sum of squares between the observed components and the expected components (wRSS) were used as a goodness of fit statistic, and the independence model, LBM(1) as a baseline model.

We followed Van der Ark (1999a) for the following guidelines in order to make a decision on the number of latent budgets to be used:

- The proportion of lack of fit with respect to the baseline model should be the largest one.

- the improvement of adding an extra latent budget should be large enough to justify the increased effort of interpreting the extra set of parameter estimates. In order to achieve this we use the criterion that the average improvement of fit per degree of freedom as shown in the `summary(goodnessfit)` times the number of the difference of degrees of freedom between two values of K being calculated.
- The results should be interpretable.

Both the models with $K = 2$ and $K = 3$ improve the previous model. We decide for $K = 3$ and try to interpret it.

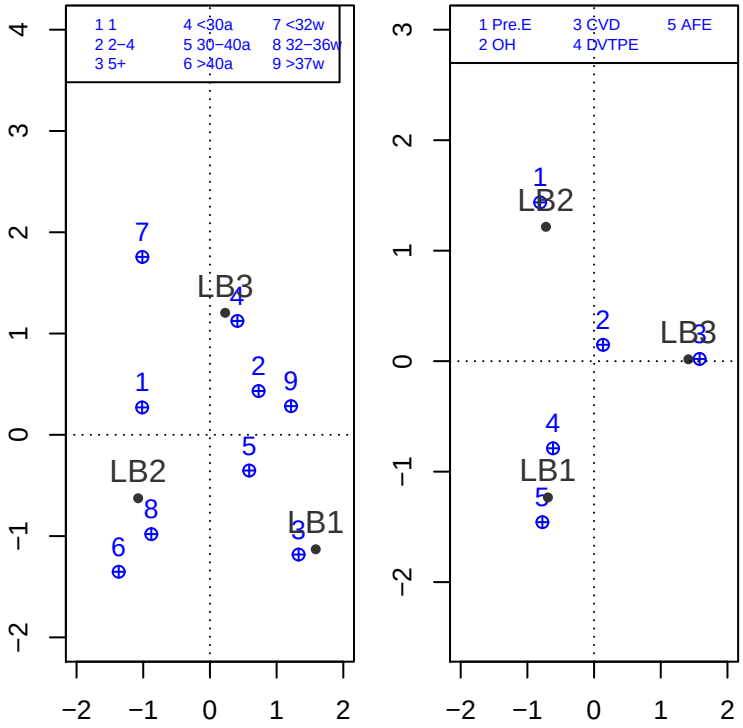


Figure 3: Example 2: Mixing parameters (left); latent components (right).

The interpretation of the latent budgets goes as follows: LB2 is explained by pre-eclampsia/eclampsia, which is a condition that occurs only in pregnant women, which is characterized by high blood pressure. LB1 has two conditions: AFE, which is a pregnancy condition; and DVTPE, which is a more general condition, and LB3 is explained by CVD. Unlike the first example, it does not put together CVD and DVTPE (Figure 3).

The mixing parameters connected to LB2 are gestational age at delivery of 32 to 36 weeks and maternal age older than 40 years. Pre.E may occur any time after the twentieth week. The LB1 is connected to women with more than 5 children, and age from 30 to 40 years. LB3 is connected to younger women with 2 to 4 children and early delivery (Figure 3).

Example 3; maternal race and country of birth

In this example we consider as explanatory variables, maternal age and country of birth in connection to pregnancy related death. The resulting data matrix is in Table 5.

This matrix has independent rows so that the product multinomial model and the MLE method are used to estimate the mixing parameters and latent components.

Table 6 shows the results of goodness of fit. Both $K = 2$ and $K = 3$ are accepted. We interpret both. Figure 4 is the default plot of **lba** for $K = 2$.

```
> set.seed(1)
> mrd2 <- pregnancy[1:4,]
> rownames(mrd2) <- c("Hispanic,foreign-born", "Hispanic, us-born",
            "White, non-hispanic", "Black, non-hispanic")
```

	Pre.E	OH	CVD	DVTPE	AFE
Hispanic,foreign-born	18	5	8	4	5
Hispanic, us-born	6	4	9	5	1
White, non-hispanic	6	7	11	6	4
Black, non-hispanic	5	2	19	5	5

Table 5: Relation between maternal race and country of birth with five causes of pregnancy related death

Number of latent budgets	df	G^2	p-value
1	12	20.5	0.06
2	6	6.8	0.34
3	2	1.61	0.45

Table 6: Goodness of fit of LBM using MLE and K= 1, 2 and 3.

```
> mrd2lbaa <- lba(mrd2, K = 2, method = "mle", what = 'outer', trace.lba = FALSE)

> par(mfrow = c(1,2))
> plotcorr(mrd2lbaa, pch.points = 20, xlim = c(-3,2.5),
+          labels.points = rownames(mrd2lbaa$Aoi), col.budget = 'gray20',
+          args.legend = list(plot = FALSE))
> plotcorr(mrd2lbaa, with.ml = 'lat', pch.points = 20,
+          labels.points = rownames(mrd2lbaa$Boi), col.budget = 'gray20',
+          args.legend = list(plot = FALSE))
```

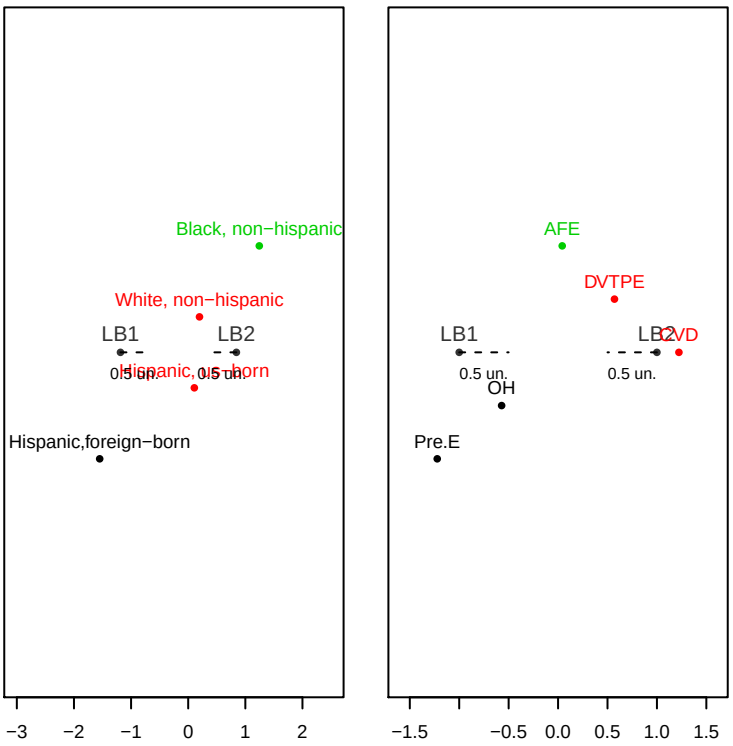


Figure 4: Mixing parameters (left); latent components (right).

The Figure 4 graphs are both one dimensional. What is important is the position at the horizontal axis. The default of the `plotcorr` function lays the numbered points out of the line so that they do not overlap, which often happens when the number of parameters increases. The latent components

show that the LB2 is composed of CVD and DVTPE which are the more general conditions, and LB1 is composed of Pre.E and OH, which are conditions more specific to pregnancy. AFE is around the origin and does not affect either one of the budgets. It should be noted that CVD and Pre.E are the ones with the strongest influence on their respective budgets because they are farther away from the origin. Looking at the mixing parameters we can see that Black women belong to LB2,that is Black women have a strong connection to more general conditions, and Hispanic foreign-born women are mostly affected by specific pregnancy conditions.

```
> set.seed(1)
> mrd2lba <- lba(mrd2, K = 3, method = "mle", what = 'outer', trace.lba = FALSE)

> par(mfrow = c(1,2))
> plotcorr(mrd2lba, with.ml = 'mix', xlim = c(-4, 3.5), ylim = c(-1.5, 2),
+          pch.points = 20, col.points = 4, pos.points = c(3, 2, 4, 3),
+          labels.points = rownames(mrd2lba$Aoi),
+          col.budget = 'gray20',
+          args.legend = list(plot = FALSE))
> plotcorr(mrd2lba, with.ml = 'lat', xlim = c(-2,2), ylim = c(-1.5,2.5),
+          pch.points = 20, col.points = 4, pos.points = c(1, 3, 3, 3, 3),
+          labels.points = rownames(mrd2lba$Boi),
+          col.budget = 'gray20',
+          args.legend = list(plot = FALSE))
```

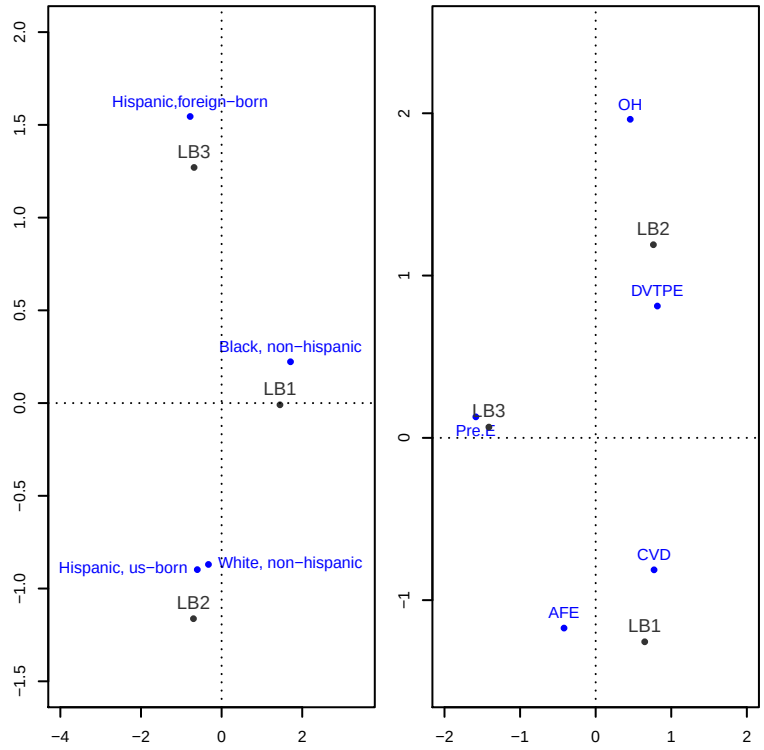


Figure 5: Example 4: Mixing parameters (left); latent components (right).

The three budgets model is shown in Figure 5. CVD and Pre.E each form a budget, LB1 and LB3 respectively; LB2 is composed of DVTPE and OH. Most important is to see that Hispanic foreign born women are strongly connected to Pre.E and Black women to CVD. This gives more emphasis to the $K = 2$ model results.

Example 4; post-materialism data

One theory of post-materialism states that political values change with the industrial and economic growth of a society. It says that people can be classified into two major groups with respect to their political values, namely *materialistic*, who seek security and materialistic supply, and *post-materialistic*, who try to bring about idealistic goals. Those two views, according to the theory, could be regarded as

the extremes of a continuum. The dataset consists of seven categories ranking from materialism (m..) to post-materialism (pm..) from a survey across Europe coded in a contingency table with 13 countries (rows) and the 7 levels of the *in depth post-materialism index*.

The countries included in the survey are:

- B=Belgium, D=Germany, DK=Denmark, E=Spain, F=France, GB=Great Britain,
- GR=Greece,I=Italy, IRL=Ireland, L=Luxembourg, NIRL=Northern Ireland, NL=Netherlands, P=Portugal.

The complete table is part of the `postmater` dataset contained in package **lba**. In order to find out how many typical societies are needed to explain the data, four models for $K = 1, 2, 3$, and 4 were estimated using the MLE method and the outer extreme solution so that the latent budgets representing the materialistic and post-materialistic concepts become as clear as possible. The goodness of fit results by using the `goodnessfit` using the G^2 function are displayed in Table 7.

Number of latent budgets	df	G^2	p-value
1	72	855.6	0.00
2	55	93.7	0.00
3	40	66.3	0.01
4	27	36.9	0.03

Table 7: Goodness of fit of LBM using MLE and $K = 1, 2, 3$, and 4.

We will now discuss the graphical results from the function `plotcorr`.

```
> data(postmater)
> new_post <- as.matrix(postmater[,-1])
> row.names(new_post) <- postmater[,1]
>
> set.seed(1)
> ex4 <- lba(new_post, method = "mle", what = 'outer', K = 4, tolG = 1e-5,
+             itmax.unide = 1e4, trace.lba = FALSE)

> par(mfrow = c(1,2))
> plotcorr(ex4, main = "Mixing Parameters", ylim = c(-1.5,2.5), zlim = c(-3,3),
+          pch.points = 20, col.points = 4, labels.points = rownames(ex4$Aoi),
+          col.budget = 'gray20', args.legend = list(plot = FALSE))
> plotcorr(ex4, with.ml = "lat", main = "Latent Components", pch.points = 20,
+           col.points = 4, labels.points = rownames(ex4$Boi), col.budget = 'gray20',
+           args.legend = list(plot = FALSE))
```

In the first graph shown in Figure 6, $K = 4$ and the plot has 3 dimensions. In both plots, mixing parameters and latent components (the third dimension principal component) explain a negligible percentage of the inertia. This means that a two dimensional plot explains as much of the total information as the three dimensional one.

Should the user wish to use a dynamic visualization of 3D graphics, the function `plotcorr` has the argument `rgl = TRUE`.

```
> tex4 <- ex4
> class(tex4) <- c("lba.2d", "lba.mle", "lba.matrix", "lba")
> par(mfrow=c(1,2))
> plotcorr(tex4, main = "Mixing Parameters", xlim = c(-2,2), ylim = c(-1.5,4),
+           pch.points = 20, col.points = 4, labels.points = rownames(tex4$Aoi),
+           col.budget = 'gray20', args.legend = list(plot = FALSE))
> plotcorr(tex4, with.ml = "lat", main = "Latent Components", xlim = c(-1.5,2.5),
+           ylim = c(-2,2.5), pch.points = 20, col.points = 4,
+           labels.points = rownames(tex4$Boi), col.budget = 'gray20',
+           args.legend = list(plot = FALSE))
```

The graph shown in Figure 7 shows, for $K = 4$, a plot with 2 dimensions where the two principal components explain almost all the information contained in the mixing parameters and latent components matrices. By inspecting the latent components matrix we can readily see that LB1 and

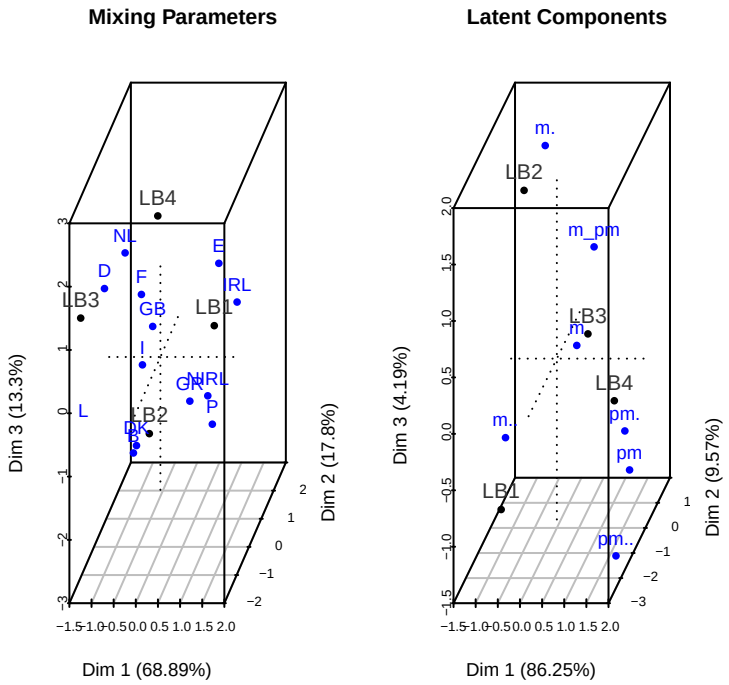


Figure 6: Example 5: Mixing parameters (left); latent components (right).

LB2 represent exactly the same budget, therefore there is no need for four budgets, which makes sense since the rule of parsimony of LBM is required to better explain the model. Considering that, we continue by interpreting the model with three latent budgets. As we did above we look at the correspondence analysis plot of the results.

```
> set.seed(1)
> ex3 <- lba(new_post, method = "mle", what = 'outer', K = 3, tolG = 1e-5,
+             itmax.unide = 1e4, trace.lba = FALSE)

> par(mfrow = c(1,2))
> plotcorr(ex3, xlim = c(-2.5,2), ylim = c(-2,2), main = "Mixing Parameters",
+           pch.points = 20, col.points = 4, labels.points = rownames(ex3$Aoi),
+           col.budget = 'gray20', args.legend = list(plot = FALSE))
> plotcorr(ex3, with.ml = "lat", main = "Latent Components", xlim = c(-2,2.5),
+           ylim = c(-1.5,2.5), pch.points = 20, col.points = 4,
+           labels.points = rownames(ex3$Boi), col.budget = 'gray20',
+           args.legend = list(plot = FALSE))
```

In figure 8 we have two graphs. Looking at the latent component part we clearly see three latent budgets. These are:

- LB1 consisting of m.. that means the clearly materialistic countries.
- LB2 consisting of pm, pm. and pm.. which means the most post-materialistic countries.
- LB3 consisting of m., m and m_{pm} which means materialistic countries leaning to post-materialism.

The mixing parameters show that:

- The materialistic countries, belonging to LB1, are: Greece, Northern Ireland and Ireland.
- Those in midway, belonging to LB3, are: Belgium and Italy.
- The post-materialistic countries, belonging to LB2, are: France, Germany and Netherlands.
- We could say that Great Britain and Luxembourg are in a group and Spain is also in a group apart. Portugal is midway between LB1 and LB3.

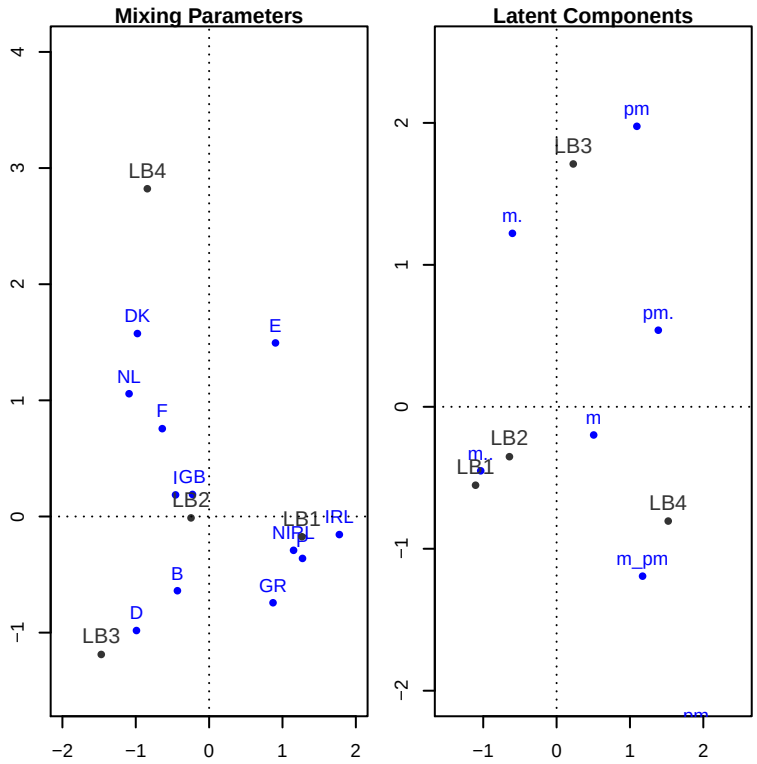


Figure 7: Example 6: Mixing parameters (left); latent components (right).

Finally, it becomes very interesting when we look at Figure 9, only the mixing parameters for $K = 2$.

In this case there are two budgets; LB1 representing the post-materialism and LB2 representing the materialism. The graph shows a clear continuum from materialistic to post-materialistic countries where some groups, as we go from one end to another, become clear. They are:

- Greece, Ireland, Northern Ireland, and Portugal the most materialistic,
- Belgium and Spain,
- Great Britain, Italy, and Luxembourg
- Denmark, France, and Germany,
- Netherlands the most post-materialistic.

```
> set.seed(1)
> ex2 <- lba(new_post, method = "mle", what = 'outer', tolG = 1e-5,
+             itmax.unide = 1e4, K = 2, trace.lba = FALSE)

> plotcorr(ex2, pch.points = 20, labels.points = rownames(ex2$Aoi),
+           col.budget = 'gray20', args.legend = list(plot = FALSE))
```

For more details see Van der Ark (1999a) page 172.

The **lba** package permits different approaches in latent budget analysis, much more than we could possibly bring to this article, and we strongly suggest the reading of Van der Ark (1999a) to get a full idea of them.

Conclusion

We have presented the **lba** package for latent budget analysis, which is derived from "A freeware computer program to perform latent budget analysis" (Van der Ark, 1999b). All unconstraint and constraint methods found in Van der Ark (1999a) were implemented.

We added some new features, such as the possibility to assign any value between zero and one as a fixed value constraint for both mixing parameters and latent components, and the implementation of two types of plots, which greatly facilitates the interpretation of the model.

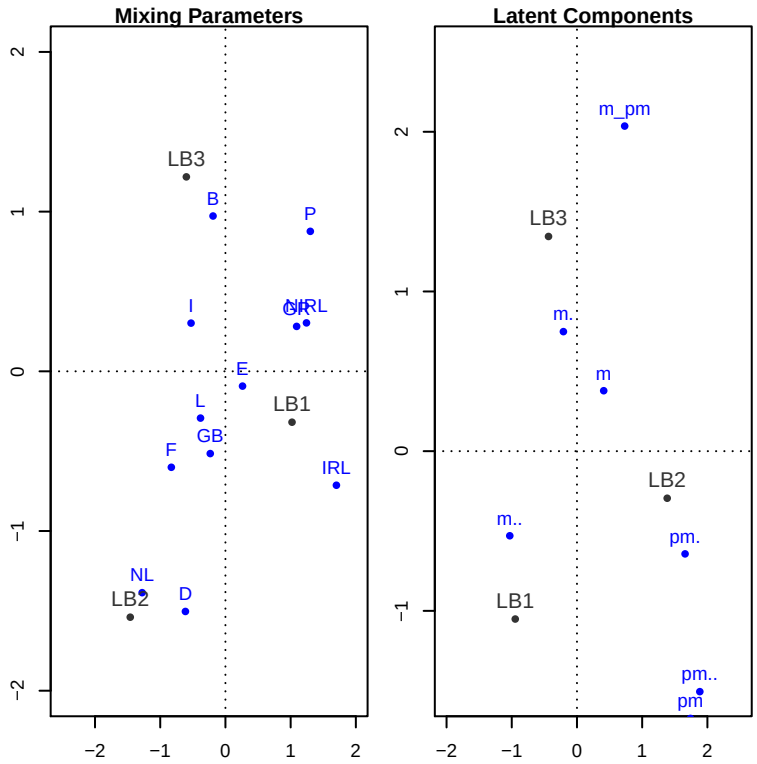


Figure 8: Example 7: Mixing parameters (left); latent components (right).

The lba package does not have the capability to analyze longitudinal data and neural networks yet. It is part of our plan to add those features to the package's capabilities.

Our next goal is to implement a new algorithm to perform identification to to replace the **alabama** package. This might be less computationally time consuming.

All the programming in R was done through the Tinn-R interface (Faria et al., 2015).

The package depends on the packages; **MASS** (Venables and Ripley, 2002), **alabama** (Varadhan, 2015), **plotrix** (Lemon, 2006), **scatterplot3d** (Ligges and Mächler, 2003), and **rgl** (Adler et al., 2016).

Bibliography

D. Adler, D. Murdoch, and others. *Rgl: 3D Visualization Using OpenGL*, 2016. URL https://CRAN.R-project.org/package=rgl. R package version 0.96.0. [p285]

E. Aquilia, G. Barone, P. Mazzoleni, S. Raneri, and G. Lamagna. Petro-archaeometric characterization of potteries from a kiln in Adrano, Sicily. *Heritage Science*, 3(1):1–9, 2015. [p269]

M. Aria. Parallel networks for compositional longitudinal data. *Statistica Applicata*, 20(1):155–179, 2008. [p269]

M. Aria, A. Mooijaart, and R. Siciliano. Neural budget networks of sensorial data. In M. Schader, W. Gaul, and M. Vichi, editors, *Between Data Science and Applied Data Analysis: Proceedings of the 26th Annual Conference of the Gesellschaft für Klassifikation e.V., University of Mannheim, July 22–24, 2002*, pages 369–377. Springer-Verlag, Berlin, Heidelberg, 2003. ISBN 978-3-642-18991-3. URL https://doi.org/10.1007/978-3-642-18991-3_42. [p269]

C. C. Clogg. Latent structure models of mobility. *American Journal of Sociology*, 86:836–868, 1981. [p269]

J. de Leeuw and P. G. M. van der Heijden. The analysis of time-budgets with a latent time-budget model. In E. D. (eds.), editor, *Data Analysis and Informatics V*, pages 159–166. North-Holland, Amsterdan, 1988. [p269, 274]

J. de Leeuw, P. J. M. van der Heijden, and P. Verboon. A latent time budget model. *Statistica Neerlandica*, 44:1–21, 1990. [p270, 271, 273, 274, 276]

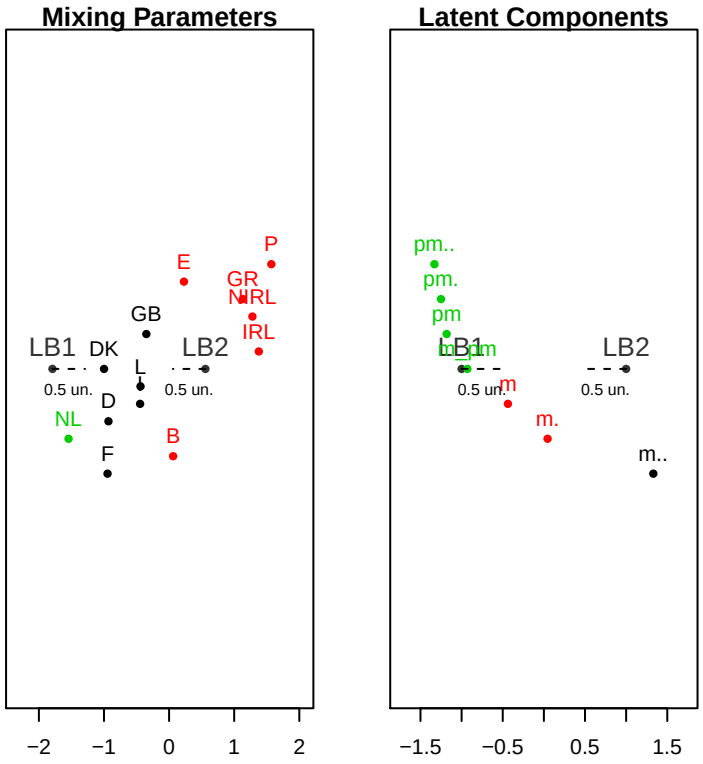


Figure 9: Example 8: Mixing parameters (left); latent components (right).

A. P. Dempster, N. M. Laird, and D. B. Rubin. Maximum likelihood from incomplete data via the EM algorithm. *Journal of the Royal Statistical Society B*, 39:1–38, 1977. [p270]

J. C. Faria, P. Grosjean, and E. Jelihovschi. Tinn-r - gui/editor for ǐanguage and environment statistical computing., 2015. URL http://sourceforge.net/projects/tinn-r. [p285]

L. A. Goodman. The analysis of systems of qualitative variables when some of the variables are unobservable. A modified latent structure approach. *American Journal of Sociology*, 79:1179–1259, 1974. [p269]

E. G. Jelihovschi, R. R. Alves, and F. M. Correa. Interacting latent budget analysis and correspondence analysis to analyze beauty salon management data. *Biometric Brazilian Journal*, 29:657–673, 2011. URL http://jaguar.fcav.unesp.br/RME/fasciculos/v29/v29_n4/A8_Enio.pdf. [p276]

J. Larrosa. A latent budget analysis approach to classification: Examples from economics. MPRA paper, University Library of Munich, Germany, 2005. URL http://EconPapers.repec.org/RePEc:pra:mprapa:12569. [p269]

J. Lemon. **Plotrix**: a package in the red light district of R. *R-News*, 6(4):8–12, 2006. [p285]

U. Ligges and M. Mächler. Scatterplot3d - an R package for visualizing multivariate data. *Journal of Statistical Software*, 8(11):1–20, 2003. URL http://www.jstatsoft.org. [p285]

E. K. Main, C. L. McCain, C. H. Morton, S. Holtby, and E. S. Lawton. Pregnancy-related mortality in California. *OBSTETRICS & GYNECOLOGY*, 4(125):938–947, 2015. [p276]

A. Mooijaart and P. G. M. van der Heijden. The EM algorithm for latent class analysis with equality constraints. *Psychometrika*, 57(2):261–269, 1992. [p273]

A. Mooijaart, P. G. M. van der Heijden, and L. A. van der Ark. A least squares algorithm for a mixture model for compositional data. *Computational Statistics & Data Analysis.*, 30:359–379, 1999. [p270, 271]

R. M. Renner. On the resolution of compositional datasets into convex combinations of extreme vectors. Technical report, Institute of Statistics and Operations Research Technical, 1988. [p269]

R. Ros-Freixedes and J. Estany. On the compositional analysis of fatty acids in pork. *Journal of Agricultural, Biological, and Environmental Statistics*, 19(1):136–155, 2014. [p269]

R. Siciliano and P. G. M. V. D. Heijden. Simultaneous latent budget analysis of a set of two way tables with constant row sum data. *Metron*, 53:5–20, 1994. [p269]

R. Siciliano and A. Mooijaart. Unconditional latent budget analysis: a neural network approach. In S. Borra, R. Rocci, M. Vichi, and M. Schader, editors, *Advances in Classification and Data Analysis*, pages 127–134. Springer-Verlag, Berlin, Heidelberg, 2001. ISBN 978-3-642-59471-7. URL https://doi.org/10.1007/978-3-642-59471-7_16. [p269]

N. Tambrea and R. Siciliano. Exploratory analysis of three-way data by simultaneous latent budget model. *Applied Stochastic Models in Business and Industry*, 4(15):469–484, 1999. [p269]

L. A. Van der Ark. *Contributions to Latent Budget Analysis*. PhD thesis, University of Utrecht, Utrecht, 1999a. [p269, 270, 271, 272, 274, 276, 278, 284]

L. A. Van der Ark. Latent budget analysis for two way tables, 1999b. [p269, 284]

L. A. van der Ark, P. G. M. van der Heijden, and D. Sikkel. On the identifiability in the latent budget model. *Journal of Classification*, 16:117–137, 1999. [p271]

P. J. M. van der Heijden, A. Mooijaart, and J. de Leeuw. Constrained latent budget analysis. *Sociological Methodology*, 22:279–320, 1992. [p270, 272, 273]

R. Varadhan. *alabama: Constrained Nonlinear Optimization*, 2015. URL http://CRAN.R-project.org/package=alabama. R package version 2015.3-1. [p272, 285]

W. N. Venables and B. D. Ripley. *Modern Applied Statistics with S*. Springer-Verlag, New York, 4th edition, 2002. URL http://www.stats.ox.ac.uk/pub/MASS4. ISBN 0-387-95457-0. [p285]

Enio Jelihovschi
Departamento de Ciências Exatas e Tecnológicas
Universidade Estadual de Santa Cruz
Cep 45650-000, Ilhéus, Bahia, Brazil
eniojelihovs@gmail.com

Ivan Bezerra Allaman
Departamento de Ciências Exatas e Tecnológicas
Universidade Estadual de Santa Cruz
Cep 45650-000, Ilhéus, Bahia, Brazil
ivanalaman@gmail.com

Semiparametric Generalized Linear Models with the gldrm Package

by Michael J. Wurm and Paul J. Rathouz

Abstract This paper introduces a new algorithm to estimate and perform inferences on a recently proposed and developed semiparametric generalized linear model (glm). Rather than selecting a particular parametric exponential family model, such as the Poisson distribution, this semiparametric glm assumes that the response is drawn from the more general exponential tilt family. The regression coefficients and unspecified reference distribution are estimated by maximizing a semiparametric likelihood. The new algorithm incorporates several computational stability and efficiency improvements over the algorithm originally proposed. In particular, the new algorithm performs well for either small or large support for the nonparametric response distribution. The algorithm is implemented in a new R package called **gldrm**.

Introduction

Rathouz and Gao (2009) introduced the generalized linear density ratio model (gldrm), which is a novel semiparametric formulation of the classical glm. Although Rathouz and Gao did not use the term gldrm, we refer to it as such because it is a natural extension of the density ratio model (see e.g. Lovric (2011)). Like a standard glm, the gldrm relates the conditional mean of the response to a linear function of the predictors through a known link function g. To be specific, let the random variable Y be a scalar response, and let X be a $p \times 1$ covariate vector for a particular observation. The model for the mean is

$$\mathrm{E}(Y|X = x) = g^{-1}(x^T\beta)\,. \tag{1}$$

Because Equation 1 holds for both gldrm and glm, the regression coefficients have the same interpretation for both models. The gldrm relaxes the standard glm assumption that the distribution of $Y|X$ comes from a particular exponential family model. Instead, assume that $Y|X$ comes from an exponential tilt model of the form

$$f(y|X = x) = f_0(y)\exp\{\theta y - b(\theta)\}\,, \tag{2}$$

where

$$b(\theta) = \log \int f_0(y)\exp(\theta y)\, d\lambda(y)\,, \tag{3}$$

and θ is defined implicitly as the solution to

$$g^{-1}(x^T\beta) = \int y\exp\{\theta y - b(\theta)\}\, d\lambda(y)\,. \tag{4}$$

Here f_0 is an unspecified probability density with respect to a measure λ. We call f_0 the *reference distribution*. Measure λ is Lebesgue if $Y|X$ is continuous, a counting measure if $Y|X$ is discrete, or a mixture of the two if $Y|X$ has a mixture distribution. Note that $\mathrm{E}(Y|X) = b'(\theta)$ and $\mathrm{Var}(Y|X) = b''(\theta)$, which are standard glm properties.

The regression coefficients β and reference distribution f_0 are estimated by maximizing a semiparametric likelihood function, which contains a nonparametric representation of f_0 that has point mass only at values of Y observed in the data. The quantity θ is not a parameter in this model, but rather a function of the free parameters β and f_0, as well as of X. This work was fully developed by Huang and Rathouz (2012), who first focused on the case where the covariate vector takes one of finitely many values, drawing on theoretical arguments advanced in the empirical likelihood literature (e.g. Owen et al. (2001)). Drawing on semiparametric profile likelihood methods, Huang (2014) went on to fully generalize the asymptotic arguments, proving consistency and asymptotic normality of the regression coefficient estimators, and deriving the asymptotic variance matrix using the profile likelihood. Huang also proved pointwise asymptotic normality of the reference cumulative distribution function estimator.

Despite these important theoretical advances, computation for this model has remained a practical challenge. The original algorithm in Rathouz and Gao (2009) is somewhat rudimentary and applies mostly to cases with finite support. It does not scale well, and stability and speed are challenges. This paper proposes a new algorithm to address these challenges and render the model more widely applicable.

In particular, the issue of optimizing the semiparametric likelihood over the f_0 point masses is improved with application of the Broyden-Fletcher-Goldfarb-Shanno (BFGS) technique (Broyden, 1970;

Fletcher, 1970; Goldfarb, 1970; Shanno, 1970). Because the number of parameters in the semiparametric likelihood associated with f_0 is equal to the number of unique points in the response vector, the rank of the Hessian matrix can become unwieldy, especially for continuous response data. The BFGS method uses an easily-computed rank-two approximate Hessian matrix that has an inverse which is much less computationally intensive to calculate.

The optimization techniques in this paper have been incorporated into the R package **gldrm** (Wurm and Rathouz, 2018), the first CRAN package for gldrm. This package estimates the gldrm model and provides coefficient standard errors. In addition, since the publication of Rathouz and Gao (2009), Huang (2014) developed likelihood ratio tests and likelihood ratio-based confidence intervals for regression coefficients and demonstrated their strong properties. The **gldrm** package provides utilities for likelihood ratio tests of nested models, as well as confidence intervals.

We present our optimization algorithm in Section 2. In Section 3, we present simulation experiments to check the finite sample performance of the gldrm estimators and the accuracy of estimated standard errors using this algorithm. In Section 4, we discuss methods for inference. Section 5, we present a simulation that benchmarks the computational time and how it scales with the number of observations and number of covariates. In Section 6, we present a summary of the functions in the **gldrm** package. In Section 7, we demonstrate the functionality of the **gldrm** package with an example in R. In Section 8, we make concluding remarks.

Optimization algorithm

Suppose we have an observed sample $(x_1, y_1), \ldots, (x_n, y_n)$ generated by the model specified in Equations 1-4. The x_i can be fixed or random, and the y_i are conditionally independent given the covariates. Let $\mathcal{S} = (s_1, \ldots, s_K)$ be the observed support, i.e. a vector containing the unique values in the set $\{y_1, \ldots, y_n\}$. If the response follows a continuous distribution, then K will equal n. Otherwise K may be less than n due to ties. Let the vector $\tilde{f}_0 = (f_1, \ldots, f_K)$ represent probability mass at the points $(s_1, \ldots, s_K)$. This is a nonparametric representation of f_0 because the true reference density may be continuous or have probability mass at values not contained in $\mathcal{S}$.

The semiparametric log-likelihood is

$$\ell(\beta, \tilde{f}_0) = \sum_{i=1}^{n} \left\{ \theta_i y_i - \log \sum_{k=1}^{K} f_k \exp(\theta_i s_k) + \sum_{k=1}^{K} I(y_i = s_k) \log f_k \right\}, \tag{5}$$

where each θ_i is defined implicitly as the solution to

$$g^{-1}(x_i^T \beta) = \frac{\sum_{k=1}^{K} s_k f_k \exp\{\theta_i s_k\}}{\sum_{k=1}^{K} f_k \exp\{\theta_i s_k\}}. \tag{6}$$

There exists a θ_i that satisfies Equation 6 as long as $g^{-1}(x_i^T \beta) \in (m, M)$ for all i, where $m \equiv \min(\mathcal{S})$ and $M \equiv \max(\mathcal{S})$.

We obtain parameter estimates from this likelihood as $\arg\max_{(\beta, \tilde{f}_0)} \ell(\beta, \tilde{f}_0)$, subject to the following constraints:

C1. $g^{-1}(x_i^T \beta) \in (m, M)$ for all i

C2. $f_k > 0$ for all $k = 1, \ldots, K$

C3. $\sum_{k=1}^{K} f_k = 1$

C4. $\sum_{k=1}^{K} s_k f_k = \mu_0$ for some chosen $\mu_0 \in (m, M)$

Constraint (C4) is necessary for identifiability because for any nonparametric density $\tilde{f}_0 = (f_1, \ldots, f_k)^T$, the "exponentially tilted" density $\tilde{f}_0^{(\alpha)} = (f_1 e^{\alpha s_1}, \ldots, f_K e^{\alpha s_K})^T / \sum_{k=1}^{K} f_k e^{\alpha s_k}$ has the same semiparametric log-likelihood for any $\alpha \in \mathbb{R}$, i.e. $\ell(\beta, \tilde{f}_0) = \ell(\beta, \tilde{f}_0^{(\alpha)})$ for all β. We can set μ_0 to be any value within the range of observed response values, but choosing a value too extreme can lead to numerical instability in estimating $\tilde{f}_0$. It usually works well to choose $\mu_0 = \frac{1}{n} \sum_{i=1}^{n} y_i$.

To perform the optimization, we take an iterative approach, alternating between optimizing over $\tilde{f}_0$ and over β (Rathouz and Gao, 2009). Each optimization step marginally optimizes the log-likelihood over one set of parameters, holding the other fixed. Neither optimization step has a closed form solution, so iterative procedures must be used. To update $\tilde{f}_0$, we propose using the BFGS technique. To update β, we use Fisher scoring, which is equivalent to iteratively re-weighted least squares (IRLS). We propose iterating BFGS until convergence for each $\tilde{f}_0$ update, while only using a single IRLS iteration to update β in between $\tilde{f}_0$ updates; this is tantamount to a profile likelihood approach to estimation of β.

Although the log-likelihood in Equation 5 is a function of $(\beta, \tilde{f}_0)$, we have expressed it in terms of $(\theta_1, \ldots, \theta_n)$; each θ_i is implicitly a function of $(\beta, \tilde{f}_0)$. The score function is also best expressed in terms of the θ_i's (see Equations 12 and 15). Consequently, our optimization algorithm requires the θ_i's to be computed each time β or $\tilde{f}_0$ is updated. To do this, we use an iterative Newton-Raphson procedure after each iteration of the β update or the $\tilde{f}_0$ update procedures.

In what follows, we detail updates of the θ_i's, of $\tilde{f}_0$, and of β. We use the notation $b(\theta)$ to denote the function defined in Equation 4 with respect to the discrete probability distribution specified by $\tilde{f}_0$, i.e. $b(\theta) = \log \sum_{k=1}^{K} f_k \exp(\theta s_k)$. We also define $\mu_i \equiv g^{-1}(x_i^T \beta)$.

θ update procedure

θ_i is defined implicitly as the solution to Equation 6, which can be written as

$$\mu_i = b'(\theta_i) \, . \tag{7}$$

To calculate θ_i, we use the Newton-Raphson procedure provided in Appendix C of Rathouz and Gao (2009). To satisfy equation 7, we need to find θ such that $\mu_i = b'(\theta)$, or equivalently $g_l(\mu_i) = g_l\{b'(\theta)\}$, where $g_l(s) = \text{logit}\left(\frac{s-m}{M-m}\right) = \log\left(\frac{s-m}{M-s}\right)$. (The transformation, g_l, stabilizes the solution.)

We use Newton-Raphson to find the root of $t(\theta) = g_l\{b'(\theta)\} - g_l(\mu_i)$. Let $\theta^{(r)}$ denote the approximate solution at the r^{th} Newton-Raphson iteration. The Newton-Raphson update is given by

$$\theta^{(r+1)} = \theta^{(r)} - \left\{ t'(\theta^{(r)}) \right\}^{-1} t(\theta^{(r)}) \, , \tag{8}$$

where

$$t'(\theta) = \frac{M-m}{\{b'(\theta)-m\}\{M-b'(\theta)\}} b''(\theta) \, , \tag{9}$$

and

$$b''(\theta) = \sum_{k=1}^{K} \{s_k - b'(\theta)\}^2 f_k \exp\{\theta s_k - b(\theta)\} \, . \tag{10}$$

We typically initialize $\theta^{(0)}$ to be the value obtained from the previous θ update procedure. The first time θ is updated, we initialize $\theta^{(0)}$ to zero for every observation. We define convergence when $|t(\theta^{(r)})| < \epsilon$, where ϵ is a small threshold such as 10^{-10}.

As $\mu_i \to M$ from the left, $\theta_i \to +\infty$. Likewise, as $\mu_i \to m$ from the right, $\theta_i \to -\infty$. To prevent numerical instability when μ_i is at or near these boundaries, we cap $|\theta_i|$ at a maximum value (500 by default). The appropriateness of this threshold would depend on the scale of response variable. Rather than adjust the threshold, we center and scale the response variable to the interval [-1, 1] (see the subsequent section on "response variable transformation").

$\tilde{f}_0$ optimization procedure

Holding β fixed at its current estimate, we need to marginally optimize the log-likelihood $\ell(\beta, \tilde{f}_0)$ over $\tilde{f}_0$, subject to constraints (C2)-(C4). The linear constraints (C3) and (C4) could be enforced using constrained optimization techniques such as Lagrange multipliers or reducing the dimension of the parameter space by two. Huang and Rathouz (2012) used the former technique, while Rathouz and Gao (2009) used the latter. We propose a different method, based on the BFGS technique, that is more computationally efficient. At each iteration, we apply a BFGS update to $\tilde{f}_0$ to improve the unconstrained log-likelihood and then transform $\tilde{f}_0$ to satisfy constraints (C3) and (C4). Application of the constraints does not affect the log-likelihood of the estimate, as the constraints are only required for identifiability (i.e. uniqueness of the optimal $\tilde{f}_0$). For any set of positive $\tilde{f}_0$ values, there exists a unique set of values with equal log-likelihood that satisfies both constraints (C3) and (C4).

We define the transformation $\tilde{g}_0 = (g_1, \ldots, g_k) = (\log f_1, \ldots, \log f_K)$ and consider the log-likelihood as a function of $\tilde{g}_0$ only, with β held fixed. Working on the log scale enforces constraint (C2)

and also improves numerical stability. Specifically, numerical stability is improved by working with the score and Hessian as a function of $\tilde{g}_0$ rather than $\tilde{f}_0$.

BFGS is a quasi-Newton procedure, which makes iterative updates using an approximate Hessian matrix along with the exact score function. Let $\tilde{g}_0^{(t)}$ be the estimate at the t^{th} iteration. The updates take the form

$$\tilde{g}_0^{(t+1)} \leftarrow \tilde{g}_0^{(t)} - H_t^{-1} S(\tilde{g}_0^{(t)}; \beta) \,. \tag{11}$$

Here, $S(\tilde{g}_0; \beta)$ is the score as a function of $\tilde{g}_0$ only, holding β fixed. It has k^{th} element

$$\{S(\tilde{g}_0; \beta)\}_k = \sum_{i=1}^{n} \left\{ I(y_i = s_k) - \frac{\exp(g_k + \theta_i s_k)}{\exp\{b(\theta_i)\}} - \frac{\exp(g_k + \theta_i s_k)}{\exp\{b(\theta_i)\}} \frac{s_k - \mu_i}{b''(\theta_i)} (y_i - \mu_i) \right\} \tag{12}$$

for $k = 1, \ldots, K$ (derivation in Appendix A). Note that this score function ignores constraints (C3) and (C4). The matrix H_t is an approximation to the Hessian of the log-likelihood as a function of $\tilde{g}_0$ only, holding β fixed. This estimate is updated with each iteration. Letting $u_t = \tilde{g}_0^{(t)} - \tilde{g}_0^{(t-1)}$ and $v_t = S(\tilde{g}_0^{(t)}; \beta) - S(\tilde{g}_0^{(t-1)}; \beta)$, we can write the BFGS estimate of the Hessian recursively as

$$H_t = H_{t-1} + \frac{v_t v_t^T}{u_t^T v_t} - \frac{H_{t-1} u_t u_t^T H_{t-1}}{u_t^T H_{t-1} u_t} \,. \tag{13}$$

H_t is a rank-2 update to H_{t-1} that satisfies the secant condition: $H_t u_t = v_t$. Furthermore, H_t is guaranteed to be symmetric and positive definite, even though the true Hessian is not full rank. (The true Hessian is not full rank because without imposing constraints (C3) and (C4), the log-likelihood does not have a unique optimum.) The BFGS update in Equation 11 requires the inverse of H_t, which can be calculated efficiently and directly, without calculating H_t. By the Sherman-Morrison-Woodbury formula,

$$H_t^{-1} = H_{t-1}^{-1} + \frac{(u_t^T v_t + v_t^T H_{t-1}^{-1} v_t)(u_t u_t^T)}{(u_t^T v_t)^2} - \frac{H_{t-1}^{-1} v_t u_t^T + u_t v_t^T H_{t-1}^{-1}}{u_t^T v_t} \,. \tag{14}$$

For an initial estimate, we propose $H_0^{-1} = \alpha I_K$, where I_K is the $K \times K$ identity matrix. We perform a line search along the gradient to choose an appropriate value for α such that $\ell(\beta, \tilde{f}_0^{(1)}) > \ell(\beta, \tilde{f}_0^{(0)})$.

As previously mentioned, constraints (C3) and (C4) are only required for identifiability. After each BFGS iteration, we impose these constraints on the $\tilde{f}_0$ estimate, which does not affect the log-likelihood of the estimate. Specifically, we apply (C3) by scaling our estimate of $\tilde{f}_0$ to sum to one. We then "exponentially tilt" the estimate to enforce constraint (C4). In other words, we compute θ such that $\sum_{j=1}^{K} s_j f_j e^{\theta s_j} / \sum_{j=1}^{K} f_j e^{\theta s_j} = \mu_0$, and set our final estimate for the iteration to be $f_k \leftarrow f_k e^{\theta s_k} / \sum_{j=1}^{K} f_j e^{\theta s_j}$ for all k.

We initialize $\tilde{g}_0^{(0)}$ to the log of the $\tilde{f}_0$ estimate obtained from the previous $\tilde{f}_0$ update procedure. We suggest using the empirical response distribution as an initial estimate of $\tilde{f}_0$. We define convergence using the relative change in the log-likelihood. Our default convergence threshold is 10^{-10}. If the log-likelihood decreases after any iteration, we backtrack by half steps, setting $\tilde{g}_0^{(t+1)} \leftarrow \frac{1}{2}\left(\tilde{g}_0^{(t+1)} + \tilde{g}_0^{(t)}\right)$ until the log-likelihood improves. In our experience, we have found that the log-likelihood improves after most iterations without taking half steps, but log-likelihood decreases can occur sporadically (both at early and late iterations).

β optimization procedure

Holding $\tilde{f}_0$ fixed at its current estimate, we could marginally optimize the log-likelihood over β using iteratively re-weighted least squares (IRLS). Rather than iterating until convergence, however, we propose using a single IRLS iteration to update β in between $\tilde{f}_0$ updates. The IRLS algorithm is simply the Newton-Raphson algorithm, but using the Fisher information in place of the negative Hessian matrix. This technique is commonly referred to as Fisher Scoring. As we now show, the Fisher Scoring update minimizes a weighted least squares expression, which is why we can refer to the algorithm as IRLS. The score function is given by

$$S(\beta; \tilde{f}_0) = \sum_{i=1}^{n} x_i \left(\frac{1}{g'(\mu_i)} \right) \left(\frac{1}{b''(\theta_i)} \right) (y_i - \mu_i) = \boldsymbol{X}^T \boldsymbol{W} \boldsymbol{r} \,, \tag{15}$$

(derivation in Appendix C) and the Fisher information is

$$\begin{aligned}\mathcal{I}(\beta;\tilde{f}_0) &= \mathrm{E}\left(S(\beta;\tilde{f}_0)S(\beta;\tilde{f}_0)^T \middle| X_1 = x_1,\ldots,X_n = x_n\right)\\ &= \sum_{i=1}^{n} x_i \left(\frac{1}{g'(\mu_i)}\right)^2 \left(\frac{1}{b''(\theta_i)}\right) x_i^T \\ &= \boldsymbol{X}^T\boldsymbol{W}\boldsymbol{X}\,,\end{aligned} \tag{16}$$

where $\boldsymbol{X}$ is an $n \times p$ matrix with rows x_i^T, $\boldsymbol{W}$ is an $n \times n$ diagonal matrix with entries $\left(\frac{1}{g'(\mu_i)}\right)^2 \frac{1}{b''(\theta_i)}$, and $\boldsymbol{r}$ is an $n \times 1$ vector with entries $g'(\mu_i)(Y_i - \mu_i)$.

Let $\beta^{(0)}$ be the estimate obtained from the previous β update procedure. The IRLS step between $\beta^{(0)}$ and the updated estimate, $\beta^{(1)}$, is

$$\begin{aligned}\beta^{(1)} - \beta^{(0)} &= \left\{\mathcal{I}(\beta^{(0)};\tilde{f}_0)\right\}^{-1} S(\beta^{(0)};\tilde{f}_0) \\ &= \left(\boldsymbol{X}^T\boldsymbol{W}\boldsymbol{X}\right)^{-1}\boldsymbol{X}^T\boldsymbol{W}\boldsymbol{r}\,.\end{aligned} \tag{17}$$

This is the solution to a weighted least squares expression, which can be computed efficiently using QR decomposition.

Response variable transformation

Numerical stability issues can occur if the $\exp\{\theta_i s_k\}$ terms become too large or small during optimization. To prevent these terms from exceeding R's default floating-point number range, we transform the response vector to the interval [-1, 1]. Specifically, the response values are transformed to $y_i^* = (y_i - \frac{m+M}{2}) \cdot \frac{2}{M-m}$. It turns out the semiparametric log-likelihood function with the transformed response and modified link function $g_*(\mu) = g(\frac{m+M}{2} + \frac{M-m}{2}\mu)$ is equivalent to the original log-likelihood. The parameters β and $\tilde{f}_0$ that optimize the modified log-likelihood also optimize the original log-likelihood (proof in Appendix D).

As mentioned in the "θ update procedure" section, we choose to cap $|\theta|$ for each observation at 500 by default, thereby restricting the rescaled $\theta_i s_k$ terms to the interval [-500, 500]. Note that the optimization function in the **gldrm** R package returns the estimated θ_i values for each observation, and these values are on the original scale of the response variable.

Inference

The **gldrm** R package (Wurm and Rathouz, 2018) can perform inference on the regression coefficients using likelihood ratio tests for nested models. It can also calculate likelihood ratio, Wald, or score confidence intervals. All three confidence intervals should yield similar results for large samples, but the Wald form may be preferred for its computational simplicity. Likelihood ratio and score confidence intervals are more computationally expensive to calculate. Huang (2014) recommends likelihood ratio tests for small samples.

The Wald asymptotic variance estimate for the β estimator can be obtained from the inverse of the information matrix given in Equation 16. Recall this information matrix is calculated with the $\tilde{f}_0$ parameters held fixed. Its inverse is a valid asymptotic variance matrix because the full information matrix is block diagonal; i.e., the β and $\tilde{f}_0$ estimators are asymptotically independent. The **gldrm** optimization function returns standard error estimates for each coefficient, which are displayed by the print method along with p-values. Wald confidence intervals are straightforward to calculate from the standard errors and can be obtained from the `gldrmCI` function. For these single-coefficient hypothesis tests and confidence intervals, we approximate the null distribution by a t-distribution with $n - p$ degrees of freedom, where n is the number of observations and p is the rank of the covariate matrix.

Likelihood ratio tests for nested models are based on the usual test statistic: 2 times the log-likelihood difference between the full and reduced model. Following Huang (2014), we approximate the null distribution by an F-distribution with q and $n - p$ degrees of freedom, where q is the difference in the number of parameters between the full and reduced models. This test can be performed by the `gldrmLRT` function.

Likelihood ratio and score confidence intervals for a single coefficient can be obtained from the `gldrmCI` function. These confidence intervals are more computationally expensive than Wald confidence intervals because an iterative method is required to search for the interval boundaries. We

use the following bisection method.

Suppose we want to obtain the lower boundary of a confidence level $1-\alpha$ interval for a single coefficient β^*, and that the gldrm estimate of this coefficient is $\hat{\beta}^*$. For the lower boundary, we need to find β^*_{lo} such that $\beta^*_{\text{lo}} < \hat{\beta}^*$ and the one-sided likelihood ratio test has a p-value equal to $\alpha/2$ under the null hypothesis $H_0 : \beta^* = \beta^*_{\text{lo}}$. As explained in the next paragraph, we can compute the p-value for any guess of β^*_{lo}. We begin by finding a guess that is too low (p-value less than $\alpha/2$) and a guess that is too high (p-value greater than $\alpha/2$). We then obtain the p-value of the midpoint. If the p-value is less than $\alpha/2$ then the midpoint becomes the new low guess, and if it is greater than $\alpha/2$ then the midpoint becomes the new high guess. We iterate this process, each time halving the distance between the low guess and high guess, until we have a guess of β^*_{lo} that has a p-value arbitrarily close to $\alpha/2$. The same procedure can be used to solve for β^*_{hi}.

Let β_0 be any constant. To calculate the likelihood ratio or score test p-value of $H_0 : \beta^* = \beta_0$, we need to optimize the log-likelihood over $\tilde{f}_0$ and all other β values, holding β^* fixed at β_0. This can be done by multiplying β_0 with the column of the $\boldsymbol{X}$ matrix corresponding to β^* and treating this vector as an offset to the linear predictor. In other words, this vector contains a fixed component of the linear predictor for each observation. The β^* column is dropped from $\boldsymbol{X}$, and the model is optimized with the offset term.

Goodness of fit

To check the model fit, we calculate the randomized probability inverse transform (Smith, 1985). This is done by evaluating the fitted conditional cdf (i.e., the cumulative distribution function, conditional on the covariates) of each observed response value. If the model fit is good, the probability inverse transform values should roughly follow a uniform distribution on the interval (0, 1). Because the gldrm fitted cdf is discrete, we use randomization to make the distribution continuous as follows.

Let $\hat{F}(y|X = x)$ denote the fitted cdf conditional on a covariate vector x evaluated at y, and let $y_i^- = \max\left(\{y_j : y_j < y_i\} \bigcup \{-\infty\}\right)$. For each observation, we draw a random value from a uniform distribution on the interval $\left(\hat{F}(y_i|X = x_i),\ \hat{F}(y_i^-|X = x_i)\right)$.

To illustrate a good gldrm fit, we generate 1,000 independent pairs (x, y) where $x \sim$ Normal(mean = 0, sd = 1) and $y|x \sim$ Normal(mean = x, sd = 1). The mean of $y|x$ is linear in x, and the data generating mechanism is an exponential tilt family, so we expect the gldrm fit to be good. We fit the model and then do a visual check by plotting the histogram and uniform QQ plot of the randomized probability inverse transform values (Figure 1).

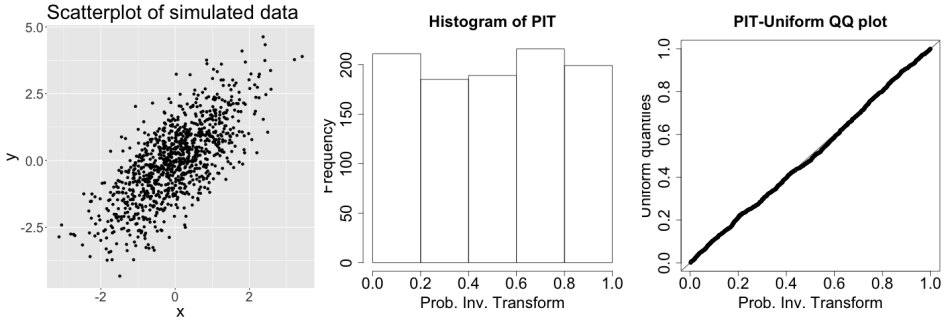


Figure 1: Scatterplot and probability inverse transform histogram and QQ plot for a good gldrm model fit.

To illustrate a poor gldrm fit, we generate 1,000 independent pairs (x, y) where $x \sim$ Normal(mean = 0, sd = 1) and $y|x \sim$ Normal(mean = x, sd = x^2). The mean of $y|x$ is again linear in x, but this data generating mechanism is not an exponential tilt family. The diagnostic plots (Figure 2) confirm that the gldrm fit is not ideal. The probability inverse transform values are concentrated near the center of the interval (0, 1) rather than being uniformly distributed. The gldrm still provides a good estimate of the regression coefficients and the mean of $y|x$, but it is not the correct model for the cdf of $y|x$. .

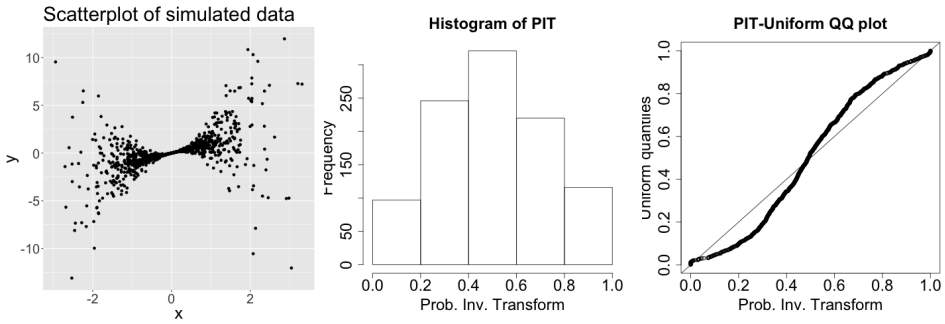


Figure 2: Scatterplot and probability inverse transform histogram and QQ plot for a poor gldrm model fit.

Simulation experiments

Under four different simulation scenarios, we check the finite sample performance of the regression coefficient estimators, $\hat{\beta}$, as well as the reference distribution cdf estimator $\hat{F}_0()$. For $\hat{\beta}$, we also check the accuracy of the estimated standard errors.

For each simulation, the data generating mechanism is a gamma glm. The covariates are the same under each simulation setting, and their values are fixed, not random. The covariate vector is $x = (x_0, x_1, x_2, x_3)$. Variable x_0 is an intercept column of ones. Variable x_1 is an indicator with exactly 20% ones in each sample (observations 5, 10, 15, 20, ... have $x_1 = 1$, and the rest have $x_1 = 0$). Variable x_2 is a continuous variable that follows a uniform sequence between zero and one in each sample (the first observation has $x_2 = 0$, and the last observation has $x_2 = 1$). Lastly, $x_3 = x_1 \cdot x_2$.

The coefficient values were $(\beta_1, \beta_2, \beta_3) = (1, 1, -2)$ for all simulation scenarios. The intercepts were $\beta_0 = 0$ for simulations 1 & 2, and $\beta_0 = 1$ for simulations 3 & 4. These values were selected so the mean response values were centered close to one and, for the identity link scenarios, guaranteed to be positive.

The log link was used for simulations 1 & 2, and the identity link was used for simulations 3 & 4. For all four simulations, the response was drawn from a gamma distribution with $\text{Var}(y|x) = \phi \text{E}(y|x)^2$, where ϕ varies by simulation to achieve different R^2 values. R^2 is defined as the squared correlation between the response and linear combination of predictors.

1. Simulation 1: log link and high R^2 ($\phi = 0.5$, $R^2 \approx 0.13$)
2. Simulation 2: log link and low R^2 ($\phi = 4$, $R^2 \approx 0.019$)
3. Simulation 3: identity link and high R^2 ($\phi = 0.25$, $R^2 \approx 0.13$)
4. Simulation 4: identity link and low R^2 ($\phi = 2$, $R^2 \approx 0.018$)

A gldrm was fit to each data set with correct link function and with $\tilde{f}_0$ constrained to have mean $\mu_0 = 1$. If we had followed our usual practice of choosing μ_0 to be the sample mean of the observed response values, then the reference distribution would have a different mean for each simulation replicate. By choosing $\mu_0 = 1$ for all replicates, the true f_0 always follows a gamma distribution with mean one and variance ϕ, where ϕ varies by simulation scenario. The value $\mu_0 = 1$ was chosen because, by design, it fell within the range of observed response values for all 2,000 replicates of each simulation scenario. The cumulative reference distribution estimate, denoted as $\hat{F}_0(\cdot)$, was computed at percentiles 0.1, 0.25, 0.5, 0.75, 0.9.

For each simulation scenario, we display four tables. The first table (Tables 1, 5, 9, and 13) contains the sample mean of each β estimator. For comparison, we also calculated the gamma glm coefficient estimates for each simulated data set and display the sample mean alongside in parentheses.

The second table (Tables 2, 6, 10, and 14) contains the root mean squared error (rmse) of each β estimator. The rmse is calculated as $\sqrt{\frac{1}{2000}\sum_{i=1}^{2000}(\hat{\beta}_i - \beta)^2}$, where $\hat{\beta}_i$ is the estimator of the i^{th} simulation replicate, and β is the true parameter value. For comparison, we also show the relative efficiency compared to the gamma glm estimator. Relative efficiency is calculated as $\text{mse}_{\text{glm}}/\text{mse}_{\text{gldrm}}$, where mse is the mean squared error (not rmse).

The third table (Tables 3, 7, 11, and 15) contains Wald confidence interval coverage rates for each β estimator. Confidence intervals were calculated based on a t-distribution with $n - 4$ degrees of freedom.

The fourth table (Tables 4, 8, 12, and 16) contains the mean of $\hat{F}_0()$, calculated at the true 10^{th}, 25^{th}, 50^{th}, 75^{th}, and, 90^{th} percentiles.

To summarize the results, the gldrm estimators perform as expected. In all four simulation scenarios, the bias of $\hat{\beta}$ and $\hat{F}_0()$ goes to zero as the sample size increases. For the high R^2 scenarios, there is very little bias even at $n = 25$. Also, the relative efficiency of the β estimators compared to the gamma glm estimators goes to one as the sample size increases. This is expected because, as Rathouz and Gao (2009) demonstrated, the β and f_0 estimators are asymptotically orthogonal, so gldrm and glm have equal asymptotic efficiency when the glm model is correctly specified. For the high R^2 scenarios, the relative efficiency is close to one even at $n = 25$. For the low R^2 scenarios, the gldrm estimator is actually more efficient than the glm estimator for small sample sizes.

The standard error estimates for gldrm β estimators are consistently low for small sample sizes, as demonstrated by low Wald confidence interval coverage rates. The coverage rates improve with increasing sample size, demonstrating good asymptotic properties of the standard error estimators. Likelihood ratio confidence intervals can be calculated with the **gldrm** R package and may have better small sample performance, as demonstrated by Huang (2014).

Simulation 1

Simulation 1 uses log link with $\beta = (0, 1, 1, -2)$ and $\phi = 0.5$. This results in an R^2 of approximately 0.13. The simulation was replicated 2,000 times.

	$\hat{\beta}$ mean (gamma glm mean)		
	n = 25	n = 100	n = 400
$\beta_0 = 0$	-0.02 (-0.02)	0.00 (0.00)	0.00 (0.00)
$\beta_1 = 1$	0.93 (0.92)	0.98 (0.99)	0.99 (0.99)
$\beta_2 = 1$	0.99 (1.00)	0.99 (1.00)	1.00 (1.00)
$\beta_3 = -2$	-2.00 (-2.00)	-2.00 (-2.01)	-1.99 (-1.99)

Table 1: Sample mean of $\hat{\beta}$. For comparison, the gamma glm sample mean is shown in parentheses.

	$\hat{\beta}$ rmse (relative efficiency)		
	n = 25	n = 100	n = 400
β_0	0.31 (0.99)	0.16 (1.00)	0.08 (1.00)
β_1	0.81 (0.96)	0.38 (1.01)	0.18 (1.00)
β_2	0.54 (1.00)	0.27 (1.00)	0.14 (0.99)
β_3	1.27 (0.96)	0.63 (1.00)	0.30 (1.00)

Table 2: Root mean squared error (rmse) of $\hat{\beta}$. In parentheses is the relative efficiency compared to the gamma glm estimator.

	80% C.I.			90% C.I.			95% C.I.		
	n = 25	n = 100	n = 400	n = 25	n = 100	n = 400	n = 25	n = 100	n = 400
β_0	0.755	0.799	0.805	0.870	0.896	0.899	0.920	0.945	0.944
β_1	0.677	0.764	0.800	0.782	0.860	0.900	0.837	0.919	0.949
β_2	0.742	0.784	0.810	0.853	0.883	0.896	0.910	0.940	0.950
β_3	0.681	0.767	0.806	0.800	0.869	0.907	0.867	0.920	0.948

Table 3: Wald confidence interval coverage rate for β.

		$\hat{F}_0(y)$ mean		
$F_0(y)$	y	n = 25	n = 100	n = 400
0.10	0.27	0.082	0.097	0.099
0.25	0.48	0.222	0.244	0.249
0.50	0.84	0.487	0.498	0.499
0.75	1.35	0.759	0.751	0.750
0.90	1.94	0.911	0.902	0.900

Table 4: Sample mean of $\hat{F}_0$ at selected true percentiles.

Simulation 2

Simulation 2 uses log link with $\beta = (0, 1, 1, -2)$ and $\phi = 4$. This results in an R^2 of approximately 0.019. The simulation was replicated 2,000 times.

	$\hat{\beta}$ mean (gamma glm mean)		
	n = 25	n = 100	n = 400
$\beta_0 = 0$	-0.16 (-0.19)	-0.05 (-0.06)	-0.02 (-0.02)
$\beta_1 = 1$	0.21 (0.29)	0.80 (0.84)	0.96 (0.97)
$\beta_2 = 1$	0.90 (0.96)	0.99 (1.01)	1.00 (1.00)
$\beta_3 = -2$	-1.74 (-1.98)	-1.93 (-2.01)	-1.99 (-2.00)

Table 5: Sample mean of $\hat{\beta}$. For comparison, the gamma glm sample mean is shown in parentheses.

	$\hat{\beta}$ rmse (relative efficiency)		
	n = 25	n = 100	n = 400
β_0	0.94 (1.14)	0.45 (1.04)	0.23 (1.00)
β_1	2.83 (1.43)	1.12 (1.09)	0.51 (1.02)
β_2	1.62 (1.19)	0.78 (1.05)	0.39 (1.01)
β_3	4.18 (1.59)	1.88 (1.14)	0.89 (1.02)

Table 6: Root mean squared error (rmse) of $\hat{\beta}$. In parentheses is the relative efficiency compared to the gamma glm estimator.

	80% C.I.			90% C.I.			95% C.I.		
	n = 25	n = 100	n = 400	n = 25	n = 100	n = 400	n = 25	n = 100	n = 400
β_0	0.704	0.770	0.782	0.806	0.877	0.888	0.869	0.933	0.945
β_1	0.626	0.724	0.781	0.732	0.832	0.883	0.794	0.897	0.935
β_2	0.680	0.755	0.774	0.795	0.862	0.881	0.869	0.925	0.939
β_3	0.633	0.722	0.762	0.750	0.829	0.879	0.819	0.895	0.935

Table 7: Wald confidence interval coverage rate for β.

		$\hat{F}_0(y)$ mean		
$F_0(y)$	y	n = 25	n = 100	n = 400
0.10	0.00	0.093	0.100	0.101
0.25	0.01	0.230	0.247	0.250
0.50	0.17	0.465	0.496	0.499
0.75	1.04	0.727	0.747	0.749
0.90	3.00	0.903	0.899	0.900

Table 8: Sample mean of $\hat{F}_0$ at selected true percentiles.

Simulation 3

Simulation 3 uses identity link with $\beta = (1, 1, 1, -2)$ and $\phi = 0.25$. This results in an R^2 of approximately 0.13. The simulation was replicated 2,000 times.

	$\hat{\beta}$ mean (gamma glm mean)		
	n = 25	n = 100	n = 400
$\beta_0 = 1$	1.00 (1.00)	0.99 (0.99)	1.00 (1.00)
$\beta_1 = 1$	1.01 (1.01)	1.00 (1.00)	0.99 (0.99)
$\beta_2 = 1$	1.01 (1.01)	1.01 (1.01)	1.00 (1.00)
$\beta_3 = -2$	-2.02 (-2.01)	-2.01 (-2.01)	-1.99 (-1.99)

Table 9: Sample mean of $\hat{\beta}$. For comparison, the gamma glm sample mean is shown in parentheses.

	$\hat{\beta}$ rmse (relative efficiency)		
	n = 25	n = 100	n = 400
β_0	0.25 (0.99)	0.13 (0.99)	0.07 (1.00)
β_1	0.88 (0.96)	0.40 (0.99)	0.20 (1.00)
β_2	0.55 (0.98)	0.28 (0.99)	0.14 (1.00)
β_3	1.26 (0.98)	0.63 (0.98)	0.32 (1.00)

Table 10: Root mean squared error (rmse) of $\hat{\beta}$. In parentheses is the relative efficiency compared to the gamma glm estimator.

	80% C.I.			90% C.I.			95% C.I.		
	n = 25	n = 100	n = 400	n = 25	n = 100	n = 400	n = 25	n = 100	n = 400
β_0	0.735	0.793	0.802	0.830	0.888	0.892	0.888	0.939	0.947
β_1	0.670	0.789	0.799	0.785	0.875	0.892	0.847	0.931	0.949
β_2	0.737	0.782	0.796	0.849	0.885	0.893	0.916	0.941	0.943
β_3	0.667	0.773	0.792	0.800	0.876	0.896	0.872	0.937	0.942

Table 11: Wald confidence interval coverage rate for β.

		$\hat{F}_0(y)$ mean		
$F_0(y)$	y	n = 25	n = 100	n = 400
0.10	0.44	0.077	0.095	0.098
0.25	0.63	0.219	0.244	0.249
0.50	0.92	0.492	0.498	0.499
0.75	1.28	0.769	0.753	0.751
0.90	1.67	0.913	0.903	0.901

Table 12: Sample mean of $\hat{F}_0$ at selected true percentiles.

Simulation 4

Simulation 4 uses identity link with $\beta = (1, 1, 1, -2)$ and $\phi = 2$. This results in an R^2 of approximately 0.018. The simulation was replicated 2,000 times.

	$\hat{\beta}$ mean (gamma glm mean)		
	n = 25	n = 100	n = 400
$\beta_0 = 1$	1.02 (1.02)	1.01 (1.01)	1.00 (1.00)
$\beta_1 = 1$	0.91 (0.89)	0.97 (1.00)	0.99 (0.99)
$\beta_2 = 1$	0.99 (1.02)	0.97 (0.98)	0.99 (1.00)
$\beta_3 = -2$	-1.86 (-1.85)	-1.92 (-1.95)	-1.98 (-1.99)

Table 13: Sample mean of $\hat{\beta}$. For comparison, the gamma glm sample mean is shown in parentheses.

	$\hat{\beta}$ rmse (relative efficiency)		
	n = 25	n = 100	n = 400
β_0	0.73 (1.17)	0.39 (1.02)	0.19 (1.00)
β_1	2.35 (1.05)	1.16 (1.09)	0.58 (1.02)
β_2	1.58 (1.18)	0.81 (1.03)	0.40 (1.00)
β_3	3.27 (1.18)	1.83 (1.11)	0.92 (1.02)

Table 14: Root mean squared error (rmse) of $\hat{\beta}$. In parentheses is the relative efficiency compared to the gamma glm estimator.

	80% C.I.			90% C.I.			95% C.I.		
	n = 25	n = 100	n = 400	n = 25	n = 100	n = 400	n = 25	n = 100	n = 400
β_0	0.661	0.761	0.794	0.754	0.845	0.893	0.802	0.893	0.938
β_1	0.595	0.718	0.767	0.692	0.822	0.868	0.741	0.870	0.921
β_2	0.665	0.769	0.779	0.777	0.864	0.895	0.847	0.910	0.943
β_3	0.666	0.711	0.777	0.789	0.824	0.877	0.871	0.892	0.932

Table 15: Wald confidence interval coverage rate for β.

		$\hat{F}_0(y)$ mean		
$F_0(y)$	y	n = 25	n = 100	n = 400
0.10	0.02	0.085	0.097	0.100
0.25	0.10	0.223	0.246	0.249
0.50	0.45	0.474	0.493	0.499
0.75	1.32	0.741	0.749	0.750
0.90	2.71	0.907	0.902	0.900

Table 16: Sample mean of $\hat{F}_0$ at selected true percentiles.

Likelihood ratio and score confidence intervals

To support our claim that likelihood ratio confidence intervals may have better small sample performance, we compared the Wald, likelihood ratio, and score 95% confidence interval coverage rates under the settings of Simulation 1 with $n = 25$. The coverage rates are shown in Table 17. The Wald coverage rates are similar to those in Table 3, but not identical because this experiment used a new set of 2,000 replicates.

While the Wald confidence intervals tend to be too narrow, the score confidence intervals tend to be a bit too wide. The likelihood ratio intervals have coverage rates much closer to 0.95 than the corresponding Wald or score intervals. This is consistent with the findings of Huang (2014).

	β_0	β_1	β_2	β_3
Wald	0.918	0.823	0.909	0.859
Likelihood ratio	0.969	0.939	0.958	0.944
Score	0.976	0.966	0.967	0.967

Table 17: Coverage rates for 95% confidence intervals under the settings of Simulation 1.

Computational time and scaling

The following simulation explores the computation time of **gldrm** and how it scales with the number of observations n and number of covariates p. For each value of n and p, we fit **gldrm** to 100 randomly generated data sets. A gldrm was fit once to each data set.

Covariates were drawn independently from a standard normal distribution, and coefficients were drawn independently from a uniform distribution on the interval (-1, 1). Simulation 1 used an identity link function with response drawn from a normal distribution with variance function $V(\mu) = 1$. Simulation 2 used a log link function with response drawn from an exponential distribution, i.e. a gamma distribution with variance function $V(\mu) = \mu^2$.

Iteration limits were set at 100 per θ update, 1,000 per f_0 update, and 100 for the outer loop (for which each iteration consists of a single-iteration β update, followed by an f_0 update). Convergence thresholds were set at 10^{-10} for the θ update, f_0 update, and outer loop update. This experiment was run using a 2.2 GHz AMD Opteron 6174 processor. Figures 3 and 4 show the average CPU seconds for Simulations 1 and 2, respectively.

We also repeated this experiment with the number of support points fixed at 25. To do this, we generated n response values from the model. The first 25 values were designated as the support, and the remaining response values were matched to the nearest support value and set to that value. This discrete data generating model is not actually a gldrm model, but we are only using it to evaluate computation time. Figures 5 and 6 show the average CPU seconds for Simulations 1 and 2, respectively, with support size fixed at 25.

In summary, the computation time scales roughly linearly with n when the support is fixed at 25. When the support grows with n, the computation time grows faster than linearly. Computation time increases with p, but there is not a consistent pattern, especially with fixed support.

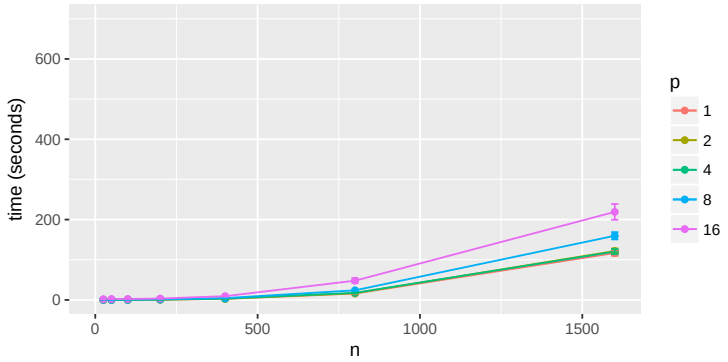


Figure 3: Mean computation time for Simulation 1. Error bars represent ±2 standard errors over 100 replicates.

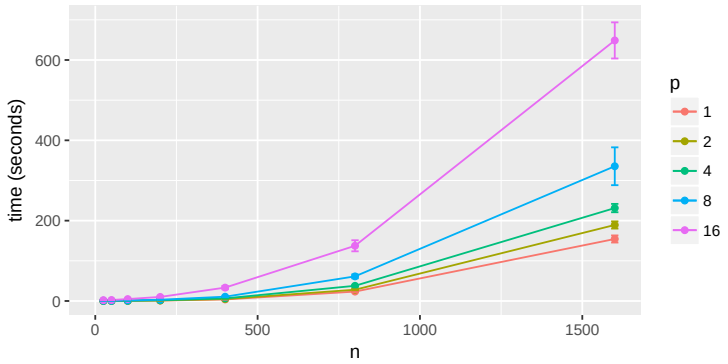


Figure 4: Mean computation time for Simulation 2. Error bars represent ±2 standard errors over 100 replicates.

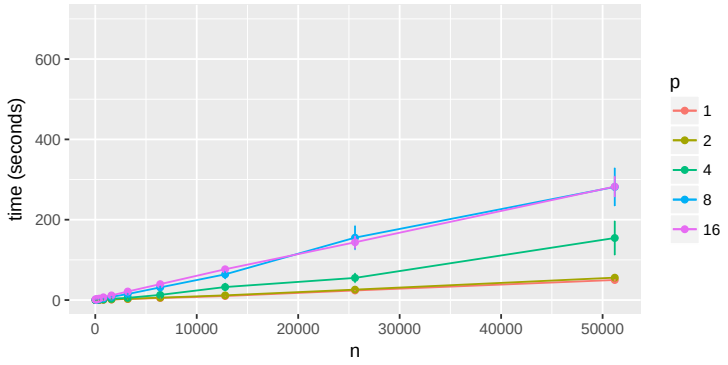


Figure 5: Mean computation time for Simulation 1 with support size fixed at 25. Error bars represent ±2 standard errors over 100 replicates.

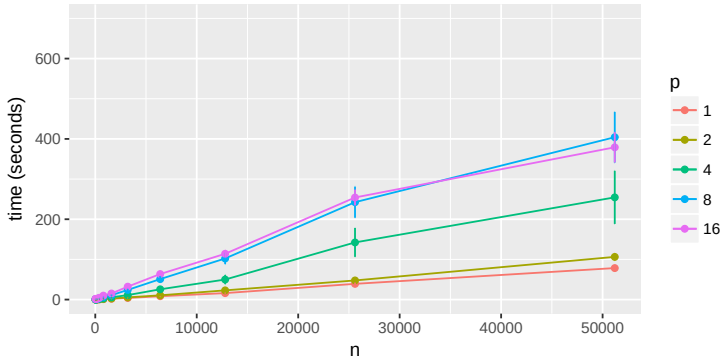


Figure 6: Mean computation time for Simulation 2 with support size fixed at 25. Error bars represent ± 2 standard errors over 100 replicates.

R package: gldrm

The **gldrm** package (Wurm and Rathouz, 2018) was written entirely in the R programming language. Its primary functions are the following.

- `gldrm` is the main function for fitting gldrm models.
- `gldrmLRT` performs a likelihood ratio test between nested gldrm models. The test statistic is calculated as $2 \times (llik - llik_0)/r$, where r is the difference is the number of parameters between the full and null models. Under the null hypothesis, the test statistic follows an asymptotic F distribution with degrees of freedom r and $n - p$, where n is the number of observations and p is the number of parameters in the full model.
- `gldrmCI` calculates Wald or likelihood ratio confidence intervals for a single coefficient.
- `predict.gldrmFit` is a predict method, which is modeled after the `print.glm` method in the **stats** package. Predictions can be obtained on the scale of the response or linear predictor, or the tilted nonparametric density for each observation can be returned.
- `gldrmPIT` Returns a set of randomized probability inverse transform values and plots the histogram and uniform QQ plot.

R example: iris data

We demonstrate the **gldrm** package using the Iris data set from the **datasets** package. This is a data set of $n = 150$ observations. We choose sepal length to be the response variable. This variable has 35 unique values, so the support is $K = 35$. We first demonstrate how to fit a gldrm model with the optimization function, which is simply called `gldrm`. We demonstrate how to perform inference on the regression coefficients. We check the goodness of fit using the randomized probability inverse transform. Finally we show how to obtain predictions for a set of observations, including the predicted mean and nonparametric estimate of the distribution function.

Fit gldrm model

The `gldrm` optimization function takes a formula and data argument, similar to R's `glm` function. Instead of passing both an error distribution and link function through a family argument, the `gldrm` only requires a link function. The `link` argument will accept the name of a link function (any function supported by `make.link` in the `stats` package is accepted). Alternatively, a custom link function can be passed as a list containing three items:

1. `linkfun` A vectorized link function.
2. `linkinv` The corresponding inverse link function, also vectorized.
3. `mu.eta` The derivative of the inverse link function, also vectorized.

This list structure is the same as that of `link-glm` class objects, which are constructed by the `make.link` function. The custom link option allows great flexibility, but it is imperative that the user correctly specifies the inverse link and its derivative.

```
R> ### Load gldrm package and Iris data from datasets package
R> library(gldrm)
R> data(iris, package = "datasets")

R> ### Fit gldrm with all variables
R> fit <- gldrm(Sepal.Length ~ Sepal.Width + Petal.Length + Petal.Width + Species,
R+              data = iris, link = "log")
R> fit

Summary of gldrm fit

Coefficients:
Estimate Std. Error t value Pr(>|t|)
(Intercept)         1.1832     0.0369   32.10  < 2e-16 ***
Sepal.Width         0.0788     0.0128    6.17  6.4e-09 ***
Petal.Length        0.1128     0.0102   11.04  < 2e-16 ***
Petal.Width        -0.0350     0.0248   -1.41    0.162
Speciesversicolor  -0.0561     0.0395   -1.42    0.157
Speciesvirginica   -0.0994     0.0557   -1.79    0.076 .
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Likelihood ratio test against null model:
F-statistic:  57.4
Numerator degrees of freedom:  5
Denominator degrees of freedom:  144
P-value:  < 2e-16
```

Inference

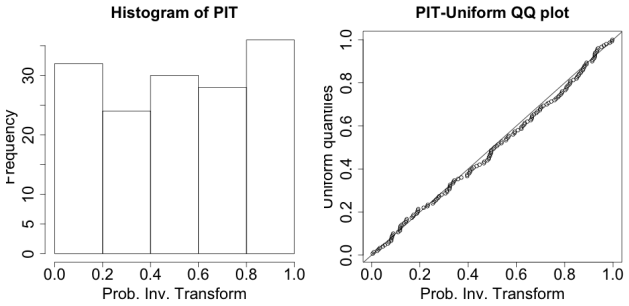
The gldrmLRT function performs a semiparametric likelihood ratio test between two nested models. We demonstrate this on the Iris data by fitting a sub-model that excludes "Species", which is a categorical variable with three levels and two degrees of freedom. We also obtain Wald and likelihood ratio confidence intervals for the petal width coefficient.

```
R> ### Fit gldrm without the categorical variable "Species"
R> fit0 <- gldrm(Sepal.Length ~ Sepal.Width + Petal.Length + Petal.Width,
R+               data = iris, link = "log")

R> ### Likelihood ratio test for the nested models
R> gldrmLRT(fit, fit0)

Likelihood ratio test:

F-statistic:  2.03
Numerator degrees of freedom:  2
Denomicator degrees of freedom:  144
P-value:  0.135

R> ### Wald 95% confidence interval for Petal.Width
R> gldrmCI(fit, term = "Petal.Width", test = "Wald", type = "2-sided", level = .95)

95% Wald confidence interval for Petal.Width:
(-0.084, 0.014)

R> ### Likelihood ratio 95% confidence interval for Petal.Width
R> gldrmCI(fit, term = "Petal.Width", test = "LRT", type = "2-sided", level = .95)

95% likelihood ratio confidence interval for Petal.Width:
(-0.094, 0.025)
```

Goodness of fit

The `gldrmPIT` function produces goodness of fit plots using the probability inverse transform. In addition to plotting, this function also returns the inverse transform values as a vector.

```
R> pit <- gldrmPIT(fit)
```



Prediction

We obtain predictions for three selected observations in the data set: one from each Species. These observations were contained in training data, but we could obtain predictions for out-of-sample observations in the same way. Using the `predict` method, we obtain the fitted mean response value for each observation, which is the estimate of $\mathrm{E}(Y|X = x)$.

We also use the `predict` method to obtain the nonparametric estimate of the conditional density $f(y|X = x)$. This is obtained by setting the argument `type = "fTilt"`, which returns an $n \times K$ matrix with $(i,k)^{\text{th}}$ entry $\tilde{f}_k \exp\{\theta_i s_k - b(\theta_i)\}$. Each row contains the nonparametric conditional density estimate for a given observation and sums to one. We use this matrix to calculate the conditional probabilities for the form $\mathrm{P}(y_1 < Y \leq y_2|X = x)$ for each observation. Note that all observed support values (sepal length values) fall between four and eight, so all probability mass falls within this interval.

```
R> ### Select three observations; one from each Species
R> newdata <- iris[c(1, 51, 101), ]

R> ### Fitted mean Sepal.Length
R> fitted_mean <- predict(fit, newdata = newdata, type = "response")
R> fitted_mean <- round(fitted_mean, 2)

R> ### Estimated Sepal.Length distribution of each observation
R> ### Note: all Sepal.Length values are between 4 and 8
R> fTilt <- predict(fit, newdata = newdata, type = "fTilt")
R> spt <- fit$spt
R> F4 <- rowSums(fTilt[ , spt <= 4])
R> F5 <- rowSums(fTilt[ , spt <= 5])
R> F6 <- rowSums(fTilt[ , spt <= 6])
R> F7 <- rowSums(fTilt[ , spt <= 7])
R> F8 <- rowSums(fTilt[ , spt <= 8])
R> Ftilt <- cbind(F5-F4, F6-F5, F7-F6, F8-F7)
R> Ftilt <- round(Ftilt, 3)
R> colnames(Ftilt) <- c("P(4=Y<=5)", "P(5<Y<=6)", "P(6<Y<=7)", "P(7<Y<=8)")
R> cbind(newdata, fitted_mean, Ftilt)

    Sepal.Length Sepal.Width Petal.Length Petal.Width    Species fitted_mean
1            5.1         3.5          1.4         0.2     setosa        5.00
51           7.0         3.2          4.7         1.4 versicolor        6.43
101          6.3         3.3          6.0         2.5  virginica        6.91
    P(4<Y<=5) P(5<Y<=6) P(6<Y<=7) P(7<Y<=8)
1       0.625     0.375     0.000     0.000
51      0.000     0.136     0.832     0.032
101     0.000     0.006     0.649     0.344
```

Discussion

We introduced a new optimization algorithm for gldrm, which computationally scales better with the number of unique observed response values. This is especially useful for estimation of continuous response data where the number of parameters in the $\tilde{f}_0$ parameter equals the sample size. In particular, the BFGS technique dramatically speeds up the $\tilde{f}_0$ optimization step, and makes gldrm much more computationally feasible for either discrete or continuous response data. The new algorithm is implemented in the **gldrm** package. Simulation results show that the algorithm and software obtain accurate estimates and standard errors. Computational time was shown to be feasible for support sizes well over one thousand.

Future research directions could include the incorporation of random effects into the gldrm framework. Optimization techniques for generalized linear mixed models, such as adaptive Gaussian quadrature, may be useful for model estimation (Pinheiro and Chao, 2006; Pinheiro and Bates, 1995).

Acknowledgments

Research reported in this publication was supported by the National Heart, Lung, and Blood Institute of the National Institutes of Health under award number T32 HL083806 (for MJW) and R01 HL094786-05A1 (for PJR). The content is solely the responsibility of the authors and does not necessarily represent the official views of the National Institutes of Health.

We thank Dr. Alan Huang, University of Queensland, for suggesting the use of probability inverse transform diagnostic plots, as well as providing the corresponding code, which we have incorporated into the **gldrm** package.

Bibliography

C. G. Broyden. The convergence of a class of double-rank minimization algorithms. *IMA Journal of Applied Mathematics*, 6(3):222–231, 1970. URL https://doi.org/10.1093/imamat/6.3.222. [p288]

R. Fletcher. A new approach to variable metric algorithms. *The Computer Journal*, 13(3):317–322, 1970. URL https://doi.org/10.1093/comjnl/13.3.317. [p289]

D. Goldfarb. A family of variable-metric methods derived by variational means. *Mathematics of Computation*, 24(109):23–26, 1970. URL https://doi.org/10.1090/S0025-5718-1970-0258249-6. [p289]

A. Huang. Joint estimation of the mean and error distribution in generalized linear models. *Journal of the American Statistical Association*, 109(505):186–196, 2014. URL https://doi.org/10.1080/01621459.2013.824892. [p288, 289, 292, 295, 298]

A. Huang and P. J. Rathouz. Proportional likelihood ratio models for mean regression. *Biometrika*, 99 (1):223–229, 2012. URL https://doi.org/10.1093/biomet/asr075. [p288, 290]

M. Lovric. *International Encyclopedia of Statistical Science.* Springer, 2011. ISBN 3642048986. [p288]

A. B. Owen, H. Crc, B. Raton, L. New, Y. Washington, A. A. B, and A. Owen. *Empirical Likelihood*. CRC Press, 2001. ISBN 1584880716. [p288]

J. C. Pinheiro and D. M. Bates. Approximations to the log-likelihood function in the nonlinear mixed-effects model. *Journal of Computational and Graphical Statistics*, 4(1):12–35, 1995. URL https://doi.org/10.1080/10618600.1995.10474663. [p303]

J. C. Pinheiro and E. C. Chao. Efficient laplacian and adaptive gaussian quadrature algorithms for multilevel generalized linear mixed models. *Journal of Computational and Graphical Statistics*, 15(1): 58–81, 2006. URL https://doi.org/10.1198/106186006X96962. [p303]

P. J. Rathouz and L. Gao. Generalized linear models with unspecified reference distribution. *Biostatistics*, 10(2):205–218, 2009. URL https://doi.org/10.1093/biostatistics/kxn030. [p288, 289, 290, 295, 306]

D. F. Shanno. Conditioning of quasi-newton methods for function minimization. *Mathematics of Computation*, 24(111):647–656, 1970. URL https://doi.org/10.1090/S0025-5718-1970-0274029-X. [p289]

J. Smith. Diagnostic checks of non-standard time series models. *Journal of Forecasting*, 4(3):283–291, 1985. URL https://doi.org/10.1002/for.3980040305. [p293]

M. Wurm and P. Rathouz. *gldrm: Generalized Linear Density Ratio Models*, 2018. URL https://CRAN.R-project.org/package=gldrm. R package version 1.5. [p289, 292, 300]

Michael J. Wurm
Department of Statistics
University of Wisconsin–Madison
1300 University Avenue
Madison, WI 53706
USA
wurm@uwalumni.com

Paul J. Rathouz
Department of Biostatistics & Medical Informatics
University of Wisconsin School of Medicine and Public Health
K6/446 CSC, Box 4675
600 Highland Avenue
Madison, WI 53792-4675
USA
rathouz@biostat.wisc.edu

Appendix

Notation

In this appendix, we use ∂ to denote the partial derivative and d to denote the total derivative. We require this notation for chain rule derivatives, specifically to derive the score function with respect to β and $\tilde{f}_0$. Each θ_i is implicitly a function of $(\beta, \tilde{f}_0)$ and hence should not be held fixed when differentiating the log-likelihood with respect to either β or $\tilde{f}_0$. We also let

$$\ell_i = \theta_i y_i - \log \sum_{k=1}^{K} f_k \exp(\theta_i s_k) + \sum_{k=1}^{K} I(y_i = s_k) \log f_k \tag{18}$$

denote the contribution of the i^{th} observation to the log-likelihood.

A. Score function for $\tilde{f}_0$

We derive Equation 12, which gives the score function with respect to $\tilde{f}_0$, holding β fixed. Using the definition of ℓ_i in Equation 18 and applying the chain rule, the k^{th} element of the score function is

$$\{S(\tilde{f}_0; \beta)\}_k = \frac{d\ell(\beta, \tilde{f}_0)}{df_k} = \sum_{i=1}^{n} \frac{d\ell_i}{df_k} = \sum_{i=1}^{n} \left(\frac{\partial \ell_i}{\partial f_k} + \frac{d\theta_i}{df_k} \frac{\partial \ell_i}{\partial \theta_i} \right) . \tag{19}$$

The first term on the RHS of Equation 19 is

$$\frac{\partial \ell_i}{\partial f_k} = \frac{I(y_i = s_k)}{f_k} - \frac{\exp\{\theta_i s_k\}}{\sum_{m=1}^{K} f_m \exp\{\theta_i s_m\}} . \tag{20}$$

The second term on the RHS of Equation 19 is

$$\frac{\partial \ell_i}{\partial \theta_i} = y_i - \frac{\sum_{k=1}^{K} s_k f_k \exp\{\theta s_k\}}{\sum_{k=1}^{K} f_k \exp\{\theta s_k\}} = y_i - \mu_i . \tag{21}$$

Recall that $\mu_i = g^{-1}(x_i^T \beta)$, which does not vary with $\tilde{f}_0$. Therefore,

$$0 = \frac{d}{df_k} \mu_i = \frac{\partial \mu_i}{\partial f_k} + \frac{d\theta_i}{df_k} \frac{\partial \mu_i}{\partial \theta_i} , \tag{22}$$

and the third term on the RHS of Equation 19 is

$$\frac{d\theta_i}{df_k} = -\frac{\partial \mu_i}{\partial f_k} \Big/ \frac{\partial \mu_i}{\partial \theta_i} = -\frac{\exp\{\theta_i s_k\}(s_k - \mu_i)}{\exp\{b(\theta_i)\}} \Big/ b''(\theta_i) . \tag{23}$$

Plugging the results of Equations 20, 21, and 23 into Equation 19, we obtain

$$\{S(\tilde{f}_0; \beta)\}_k = \sum_{i=1}^{n} \left\{ \frac{I(y_i = s_k)}{f_k} - \frac{\exp(\theta_i s_k)}{\exp\{b(\theta_i)\}} - \frac{\exp(\theta_i s_k)}{\exp\{b(\theta_i)\}} \frac{s_k - \mu_i}{b''(\theta_i)} (y_i - \mu_i) \right\} . \tag{24}$$

We obtain the result of Equation 12 by applying the Jacobian transformation $\{S(\tilde{g}_0; \beta)\}_k = f_k \{S(\tilde{f}_0; \beta)\}_k$.

B. Information matrix for $\tilde{f}_0$

The $\tilde{f}_0$ information matrix is not used during the BFGS optimization routine. However, we derive it here to correct a missing term in the Rathouz and Gao (2009) derivation. Each matrix element is

$$\begin{aligned}\left\{\mathcal{I}(\tilde{f}_0;\beta)\right\}_{(k,m)} &= \mathrm{E}\left(\left\{S(\tilde{f}_0;\beta)\right\}_k \left\{S(\tilde{f}_0;\beta)\right\}_m \middle| X_k = x_k, X_m = x_m\right) \\ &= \sum_{i=1}^{n} \mathrm{E}\left\{A_k A_m - 2A_k B_m + B_k B_m \middle| X_k = x_k, X_m = x_m\right\},\end{aligned} \tag{25}$$

where

$$A_k = \frac{I(y_i = s_k)}{f_k} - \frac{\exp(\theta_i s_k)}{\exp\{b(\theta_i\}}$$

and

$$B_k = \frac{\exp(\theta_i s_k)}{\exp\{b(\theta_i)\}} - \frac{\exp(\theta_i s_k)}{\exp\{b(\theta_i)\}} \frac{s_k - \mu_i}{b''(\theta_i)}(y_i - \mu_i)\,.$$

From here on, all expectations are conditional on $X_k = x_k, X_m = x_m$, but we suppress the conditional expression. The first term on the RHS of Equation 25 is

$$\mathrm{E}(A_k A_m) = \frac{\exp(\theta_i s_k) I(k = m)}{f_k \exp\{b(\theta_i)\}} - \frac{\exp(\theta_i s_k + \theta_i s_m)}{\exp\{2b(\theta_i)\}}\,. \tag{26}$$

The second term on the RHS of Equation 25 is

$$E(B_k B_m) = \frac{\exp(\theta_i s_k + \theta_i s_m)}{\exp\{2b(\theta_i)\}} \frac{(s_k - \mu_i)(s_m - \mu_i)}{b''(\theta_i)}\,, \tag{27}$$

where we use the fact that $\mathrm{E}\{(y_i - \mu_i)^2\} = \mathrm{Var}(y_i) = b''(\theta_i)$. The third term on the RHS of Equation 25 is

$$\mathrm{E}(A_k B_m) = \frac{\exp(\theta_i s_k + \theta_i s_m)}{\exp\{2b(\theta_i)\}} \frac{(s_k - \mu_i)(s_m - \mu_i)}{b''(\theta_i)}\,, \tag{28}$$

where we use the fact that $\mathrm{E}\{I(y_i = s_k)(y_i - \mu_i)\} = (y_k - \mu_i) f_k \exp(\theta_i s_k) / \exp\{b(\theta_i)\}$. Plugging the results of Equations 26-28 into Equation 25, we obtain

$$\left\{\mathcal{I}(\tilde{f}_0;\beta)\right\}_{(k,m)} = \sum_{i=1}^{n} \left\{ \frac{\exp(\theta_i s_k) I(k = m)}{f_k \exp\{b(\theta_i)\}} - \frac{\exp(\theta_i s_k + \theta_i s_m)}{\exp\{2b(\theta_i)\}} - \frac{\exp(\theta_i s_k + \theta_i s_m)}{\exp\{2b(\theta_i)\}} \frac{(s_k - \mu_i)(s_m - \mu_i)}{b''(\theta_i)} \right\}. \tag{29}$$

C. Score function for β

We derive Equation 15, which gives the score function with respect to β, holding $\tilde{f}_0$ fixed. By the chain rule, each element is

$$S(\beta;\tilde{f}_0) = \frac{d\ell(\beta,\tilde{f}_0)}{d\beta} = \sum_{i=1}^{n} \frac{d\ell_i}{d\beta} = \sum_{i=1}^{n} \frac{d\mu_i}{d\beta} \frac{d\theta_i}{d\mu_i} \frac{\partial \ell_i}{\partial \theta_i} \tag{30}$$

We already derived the first term on the RHS of Equation 30 in Equation 21. Because $\mu_i = b'(\theta_i)$,

$$\frac{d\theta_i}{d\mu_i} = 1 / \left(\frac{d\mu_i}{d\theta_i}\right) = \frac{1}{b''(\theta_i)}\,. \tag{31}$$

Because $g(\mu_i) = x_i^T \beta$,

$$\frac{d\mu_i}{d\beta} = x_i \left(\frac{1}{g'(\mu_i)}\right)\,. \tag{32}$$

Plugging the results of Equations 21, 31, and 32 into Equation 30, we obtain the result of Equation 15.

D. Proof that transformed response with modified link function has equivalent log-likelihood

Let $\nu \equiv \frac{m+M}{2}$ and $\rho \equiv \frac{M-m}{2}$. Let $y_i^* = (y_i - \nu)/\rho$ and $s_k^* = (s_k - \nu)/\rho$. Equation 18 can be rewritten

$$\begin{aligned} \ell_i &= \theta_i(\nu + \rho y_i^*) - \log \sum_{k=1}^{K} f_k \exp\{\theta_i(\nu + \rho s_k^*)\} + \sum_{k=1}^{K} I(y_i^* = s_k^*) \log f_k \\ &= \theta_i^* y_i^* - \log \sum_{k=1}^{K} f_k \exp\{\theta_i^* s_k^*\} + \sum_{k=1}^{K} I(y_i^* = s_k^*) \log f_k \,, \end{aligned} \tag{33}$$

where $\theta_i^* = \theta_i \rho$. Equation 6 can be rewritten as

$$\begin{aligned} g^{-1}(x_i^T \beta) &= \frac{\sum_{k=1}^{K} (\nu + \rho s_k^*) f_k \exp\{\theta_i(\nu + \rho s_k^*)\}}{\sum_{k=1}^{K} f_k \exp\{\theta_i(\nu + \rho s_k^*)\}} \\ &= \nu + \rho \left(\frac{\sum_{k=1}^{K} s_k^* f_k \exp\{\theta_i^* s_k^*\}}{\sum_{k=1}^{K} f_k \exp\{\theta_i^* s_k^*\}} \right) . \end{aligned} \tag{34}$$

Hence, if we define a modified link function $g_*(\mu) = g(\nu + \rho\mu)$ with inverse $g_*^{-1}(\eta) = (g^{-1}(\eta) - \nu)/\rho$, we can write

$$g_*^{-1}(x_i^T \beta) = \left(\frac{\sum_{k=1}^{K} s_k^* f_k \exp\{\theta_i^* s_k^*\}}{\sum_{k=1}^{K} f_k \exp\{\theta_i^* s_k^*\}} \right) . \tag{35}$$

In other words θ_i^* has the same implicit definition as θ_i when the modified link function and transformed response variable are used in place of the oginal. This shows that, as a function of $(\beta, \tilde{f}_0)$, the log-likelihood with the transformed response and modified link function is equivalent to the original log-likelihood. Note that θ_i^* must be multiplied by $1/\rho$ if one would like calculate θ_i on the original scale.

LP Algorithms for Portfolio Optimization: The PortfolioOptim Package

by Andrzej Palczewski

Abstract The paper describes two algorithms for financial portfolio optimization with the following risk measures: CVaR, MAD, LSAD and dispersion CVaR. These algorithms can be applied to discrete distributions of asset returns since then the optimization problems can be reduced to linear programs. The first algorithm solves a simple recourse problem as described by Haneveld using Benders decomposition method. The second algorithm finds an optimal portfolio with the smallest distance to a given benchmark portfolio and is an adaptation of the least norm solution (called also normal solution) of linear programs due to Zhao and Li. The algorithms are implemented in R in the package **PortfolioOptim**.

Introduction

The construction of an optimal portfolio of financial instruments is one of primal goals in asset management. Recent advances in risk measurement advocate using risk measures that take into account the tail behavior of asset return distributions, such as conditional value-at-risk or lower semi-absolute deviation. For these risk measures, finding optimal portfolios in risk-return setting leads to convex programming problems. For a finite discrete distribution of returns and many practically used risk measures the optimization problem can be solved by LP methods. The portfolio optimization by LP methods leads to two problems: for distributions with a large number of values (for example, a large sample from a continuous distribution) one obtains a unique solution but the LP problem is resource demanding; for distributions with a small number of values one often obtains a non-unique solution, which is of limited use for asset management. In the latter case, a unique optimal portfolio is obtained by projecting a benchmark portfolio on the whole set of solutions. In the paper we present an algorithm that solves efficiently the problem with large distributions and an algorithm that finds an orthogonal projection of the benchmark portfolio on the space of solutions.

The R language and environment for statistical computing offer a large variety of tools for portfolio optimization. General purpose optimization tools are reviewed by Theussl and Borchers (2016) (R packages for solving optimization problems) and Koenker and Mizera (2014) (R packages for convex optimization). The book by Pfaff (2016) provides an overview of specific functions for portfolio optimization which are embedded in financial packages. Here we mention a selection of R-packages dedicated primarily to portfolio optimization. The package **fPortfolio** by Würtz et al. (2009) offers a large set of functions for financial data analysis and enables portfolio optimization in mean-variance, mean-MAD and mean-CVaR settings. For these portfolio problems the package employs existing optimization tools: LP, QP and NLP solvers. The package **PortfolioAnalytics** by Peterson and Carl (2015) uses standard linear and quadratic optimization tools (**Rglpk** and **quadprog**) and a number of new packages: stochastic optimization (**DEoptim**, **GenSA**) and particle swarm optimization (**psoptim**). The package **parma** by Ghalanos (2016) offers scenario and moment based optimization of portfolios for a large class of risk and deviation measures using **Rglpk**, **quadprog** and, for non-linear optimization **nloptr**. The above mentioned packages are very effective in solving medium-size portfolio problems, however, due to their use of standard optimization tools they cannot deal with large problems either in the number of assets or the size of the distribution. Moreover, none of these packages is able to select an optimal portfolio which is closest to a given benchmark portfolio.

The new package **PortfolioOptim** overcomes the aforementioned limitations solving portfolio problems in mean-risk setting with linear portfolio constraints and risk measures that make the problem reducible to a linear program. Attempts to apply LP solvers to more general portfolio problems (cf. Mansini et al. (2014) and references therein) are not included in the package. Our first contribution is an efficient search algorithm for optimal portfolios in stochastic programs with a very large number of scenarios. A large number of scenarios in portfolio optimization appears often when a continuous distribution of returns is approximated by a discrete one. The goal is to obtain an optimal portfolio which approximates the optimal portfolio for the continuous distribution. This can be achieved by performing optimization on a large discrete sample generated from the continuous distribution under consideration. A large sample leads to a high dimensional LP problem which can be difficult to solve by general-purpose LP solvers. This problem has been addressed by Künzi-Bay and Mayer (2006) who solved the portfolio optimization problem in mean-CVaR setting using Benders decomposition. They have computed accurately CVaR of the optimal portfolio using samples of order 10^4. Our extension of this result is twofold. First, we design an algorithm that implements Benders decomposition for a general class of simple recourse problems. Portfolio optimization problems for

the risk measures mentioned above are special cases of these simple recourse problems. Second, using an internal point LP solver from GLPK library and appropriately modifying the stopping criterion we substantially increase the size of discrete samples which can be used in computations. Our algorithm can perform computations for samples of order 10^6 on a standard computer in a few seconds.

Our second contribution relates to portfolio optimization problems when only a small number of random scenarios is available. Small discrete samples in portfolio optimization appear when the distribution of returns is an empirical distribution of historical returns. Usually one takes 5 or 10 years of weekly or monthly returns giving between 120 and 500 samples. Solutions to linear programs of such dimensions are in many cases non-unique and can occupy a whole face of a multidimensional simplex. In such cases, a standard software finds only one solution, usually an vertex of the solution simplex. This is often not the optimal portfolio which asset managers consider as the most appropriate. Asset managers have usually certain beliefs about the composition of optimal portfolios and those can be expressed as similarity or vicinity of the desired optimal portfolio to some benchmark portfolio. Therefore, one wants to find an optimal portfolio which has the smallest distance to a given benchmark, or, equivalently, the orthogonal projection of the benchmark on the space of optimal portfolios.

In the LP literature a special case of the above projection problem has been discussed for a long time. This is the problem of the least norm LP solution, also called normal solution (cf. Zhao and Li (2002) and references cited therein). Unfortunately, the algorithms which find normal solutions cannot be easily adapted to our problem. The reason is that portfolio weights make up only a fraction of all coordinates of the LP solution (see the examples in Sections LP computable portfolio problems and Benders decomposition). Contrary to the normal solution which is the projection of the origin onto the simplex of optimal solutions, we are looking for a projection onto a subspace of portfolio weights. In addition, we are not looking for a solution with the least norm but for a solution with the smallest distance to an arbitrary vector (benchmark portfolio). It appears however, that the regularized central path algorithm due to Zhao and Li (2002) which solves simultaneously primal and dual LP problems finding the least norm solutions to both problems, can be modified to our purposes. Our contribution is the extension of this algorithm to the following problem. Given the set S^* of optimal solutions to an LP problem, find $x^* \in S^*$ such that $\|B(x^* - \hat{x})\| \leq \|B(x - \hat{x})\|$ for all $x \in S^*$, where $\hat{x}$ is a given vector and B is the operator of projection on a subspace. This general algorithm is then adapted to obtain an optimal portfolio with the smallest distance to a benchmark (see Section Projection algorithm for portfolio optimization for details).

The rest of the paper is organized as follows. In Section LP computable portfolio problems, we describe the portfolio optimization problems that can be reduced to LP problems and are considered as working examples in the rest of the paper. These portfolio optimization problems are also implemented in the package **PortfolioOptim**. Section Benders decomposition analyzes Benders decomposition algorithm applied to simple recourse problems. As an illustration, we present applications to the portfolio optimization in mean-CVaR and mean-LSAD settings. Section Projection algorithm for portfolio optimization describes the adaptation of the path-following algorithm of Zhao and Li (2002) to the construction of an optimal portfolio with the smallest distance to a benchmark portfolio. We present also the proof of convergence for the modified algorithm. Computational examples and the analysis of performance of both algorithms are presented in Section Numerical examples. The paper ends with short Summary.

LP computable portfolio problems

We consider the portfolio optimization problem in mean-risk setting. The risk is measured by a risk (or deviation) measure $\mathcal{R}$ (for the definitions of risk and deviation measures the reader is advised to consult the paper by Rockafellar et al. (2006)). Let R be a distribution of returns of J risky assets. We denote by $\hat{R}$ centered returns, i.e. $\hat{R} = R - \mathbb{E}[R]$. Consider the problem of finding an optimal portfolio u which fulfills the conditions

$$\begin{cases} \mathcal{R}\left(u^T R\right) \to \min, \\ u^T \mathbb{E}\left[R\right] \geq r_0, \\ u \in \mathbb{X}, \end{cases} \tag{1}$$

where $\mathbb{X}$ is a polyhedral set (i.e. given by linear constraints) and r_0 is the required return of the portfolio. The set $\mathbb{X}$ is usually the whole space (when no limitations on trading positions are imposed) or the positive orthant (when short-selling of assets is prohibited).

We will assume that at optimum the constraint $u^T \mathbb{E}[R] \geq r_0$ is binding. This is the case when r_0 is not smaller than the return for the minimal risk portfolio, i.e. the portfolio which solves the problem $\min_{u \in \mathbb{X}} \mathcal{R}(u^T R)$.

It appears that for a number of risk measures the optimization problem (1) formulated for a discrete distribution R can be reduced to a linear program. Let the asset returns R be given by a discrete

distribution placing weights p_n at points r_n, $n = 1, \ldots, N$. When the distribution is centered points of $\hat{R}$ will be denoted by $\hat{r}_n$.

Consider the risk measured by conditional value at risk (CVaR). For a random outcome X we define $\mathrm{VaR}_\alpha(X)$ as

$$\mathrm{VaR}_\alpha(X) = -\inf\{z : \mathbb{P}(X \le z) > \alpha\}.$$

Then the conditional value at risk is defined as

$$\mathrm{CVaR}_\alpha(X) = \frac{1}{\alpha}\int_0^\alpha \mathrm{VaR}_p(X)\, dp.$$

Rockafellar and Uryasev (2000) observed that for $X = u^T R$ the minimization of CVaR can be formulated as the following nonlinear problem

$$\min_{u,\xi} \xi + \frac{1}{1-\alpha}\mathbb{E}\left[\left(-u^T R - \xi\right)^+\right], \tag{2}$$

where the latent variable ξ corresponds to $\mathrm{VaR}_\alpha(u^T R)$. For discrete distributions and the risk measured by CVaR, the portfolio optimization problem can therefore be reformulated as the linear program

$$\begin{cases} \xi + \frac{1}{1-\alpha}\sum_{n=1}^N p_n y_n \to \min, \\ y_n \ge 0, \quad n = 1, \ldots, N, \\ y_n + \xi + r_n^T u \ge 0, \quad n = 1, \ldots, N, \\ \mu^T u \ge r_0, \\ u \in \mathbb{X}. \end{cases} \tag{3}$$

Here $\mu = \mathbb{E}[R]$, while y_n plays a role of the shortfall of return $u^T r_n$ below ξ: $y_n = (-\xi - u^T r_n)^+ = (\xi + u^T r_n)^-$. For deviation CVaR we replace r_n by $\hat{r}_n$ in (3).

Consider now mean-MAD and mean-LSAD optimization problems. MAD (mean absolute deviation) is defined as (cf. Konno and Yamazaki (1991))

$$MAD\left(u^T R\right) = \mathbb{E}\left[\left|u^T R - \mathbb{E}[u^T R]\right|\right] = \mathbb{E}\left[\left|u^T \hat{R}\right|\right]. \tag{4}$$

For discrete distributions mean-MAD optimization problem reads

$$\begin{cases} \sum_{n=1}^N p_n \left|\hat{r}_n^T u\right| \to \min, \\ \mu^T u \ge r_0, \\ u \in \mathbb{X}. \end{cases} \tag{5}$$

This problem can be formulated as the linear program

$$\begin{cases} \sum_{n=1}^N p_n y_n \to \min, \\ y_n \ge 0, \quad n = 1, \ldots, N \\ y_n + \hat{r}_n^T u \ge 0, \quad n = 1, \ldots, N, \\ y_n - \hat{r}_n^T u \ge 0, \quad n = 1, \ldots, N, \\ \mu^T u \ge r_0, \\ u \in \mathbb{X}. \end{cases} \tag{6}$$

For LSAD (lower semi absolute deviation) as defined by Konno (1990) or Konno et al. (2002)

$$LSAD\left(u^T R\right) = \mathbb{E}\left[\left|u^T R - \mathbb{E}[u^T R]\right|_-\right] = \mathbb{E}\left[\left|u^T \hat{R}\right|_-\right], \tag{7}$$

mean-LSAD optimization has the form

$$\begin{cases} \sum_{n=1}^N p_n \left|\hat{r}_n^T u\right|_- \to \min, \\ \mu^T u \ge r_0, \\ u \in \mathbb{X}. \end{cases} \tag{8}$$

This leads to the following linear program

$$\begin{cases} \sum_{n=1}^{N} p_n y_n \to \min, \\ y_n \geq 0, \quad n = 1, \ldots, N, \\ y_n + \hat{r}_n^T u \geq 0, \quad n = 1, \ldots, N, \\ \mu^T u \geq r_0, \\ u \in \mathbb{X}. \end{cases} \tag{9}$$

Since $LSAD(u^T R) = \frac{1}{2} MAD(u^T R)$, in what follows we shall consider only one of the above optimization problems.

Benders decomposition

Algorithm

In this section, we present a solution to a simple linear recourse problem with random technology matrix when the distribution of stochastic variable is represented by a large number of scenarios. The simple recourse problem is a special case of two-stage recourse problem

$$\begin{cases} c^T x + Q(x) \to \min, \\ x \in \mathbb{X}, \end{cases} \tag{10}$$

with the following form of the recourse subproblem

$$\begin{cases} Q(x) = \mathbb{E}[v(x)], \\ v(x) = \min\limits_{y} \left\{ \left(q^+\right)^T y^+ + \left(q^-\right)^T y^- : y^+ - y^- = b - Ax,\ y^+, y^- \in \mathbb{R}_+^m \right\}. \end{cases} \tag{11}$$

In this framework, q^+ and q^- are known penalty costs, matrix $A = A(\omega)$ and vector $b = b(\omega)$ are random and $\mathbb{X}$ is a bounded, convex subset in $\mathbb{R}^l$.

We see that each pair of recourse variables (y_i^+, y_i^-), $i = 1, \ldots, m$, depends only on the i-th row in the condition $b - Ax$, so that their optimal values can be determined independently. Thus, the second-stage value function $v(x)$ is separated into the sum of m functions

$$v(x) = \sum_{i=1}^{m} \min_{y_i} \left\{ q_i^+ y_i^+ + q_i^- y_i^- : y_i^+ - y_i^- = b_i - A_i x,\ y_i^+, y_i^- \geq 0 \right\},$$

where A_i is the ith row of matrix A and b_i is the ith element of vector b.

For each optimization problem

$$\min_{y_i} \left\{ q_i^+ y_i^+ + q_i^- y_i^- : y_i^+ - y_i^- = b_i - A_i x,\ y_i^+, y_i^- \geq 0 \right\}, \tag{12}$$

the dual problem has the form

$$\sup_{\lambda_i} \left\{ \lambda_i (b_i - A_i x) : \lambda_i \leq q_i^+, -\lambda_i \leq q_i^- \right\}.$$

The dual problem is feasible only if $q_i^+ + q_i^- \geq 0$. When this condition holds, the solution to the dual problem is given by

$$\lambda_i = \begin{cases} q_i^+, & \text{if } b_i - A_i x > 0, \\ -q_i^-, & \text{if } b_i - A_i x \leq 0, \end{cases}$$

and the optimal solution to problem (12) has the form

$$\begin{aligned} y_i^+ &= \max(0, (b_i - A_i x)), \\ y_i^- &= \max(0, -(b_i - A_i x)). \end{aligned}$$

Then the value function $Q(x)$ can be written as

$$\begin{aligned} Q(x) &= \sum_{i=1}^{m} \left(q_i^+ \mathbb{E}\left[(b_i - A_i x)^+ \right] + q_i^- \mathbb{E}\left[(b_i - A_i x)^- \right] \right) \\ &= \sum_{i=1}^{m} \left(q_i^+ \mathbb{E}\left[(A_i x - b_i)^- \right] + q_i^- \mathbb{E}\left[(A_i x - b_i)^+ \right] \right). \end{aligned} \tag{13}$$

Since $s^+ - s^- = s$, we have

$$\begin{aligned} Q(x) &= \sum_{i=1}^{m} \left(q_i^- \left(\bar{A}_i x - \bar{b}_i \right) + \left(q_i^+ + q_i^- \right) \mathbb{E}\left[\left(A_i x - b_i \right)^- \right] \right) \\ &= \left(q^- \right)^T \bar{A} x - \left(q^- \right)^T \bar{b} + \mathbb{E}\left[\left(q^+ + q^- \right)^T \left(b - Ax \right)^+ \right], \end{aligned} \tag{14}$$

where $\bar{A}_i = \mathbb{E}[A_i]$ and $\bar{b}_i = \mathbb{E}[b_i]$.

For a discrete probability space with $\mathbb{P}(\omega = \omega_n) = p_n$, $n = 1, \ldots, N$, the simple recourse problem can be reformulated in the following way

$$\begin{cases} c^T x + \left(q^- \right)^T \left(\bar{A} x - \bar{b} \right) + \sum_{n=1}^{N} p_n h_n \to \min, \\ h_n \geq \left(q^+ + q^- \right)^T \left(b\left(\omega_n \right) - A\left(\omega_n \right) x \right), \\ h_n \geq 0, \\ x \in \mathbb{X}. \end{cases} \tag{15}$$

This linear program with a large size N of the probability space is difficult to solve due to the number of constraints. Klein Haneveld and Van der Vlerk (2006) solved it using a special version of the L-shaped method. Their approach was further extended by Künzi-Bay and Mayer (2006) who applied directly Benders decomposition (cf., Benders (1962)). In terms of Benders cuts (15) is equivalent to (we skip $(q^-)^T \bar{b}$ which is constant)

$$\begin{cases} \min_{x,w} c^T x + \left(q^- \right)^T \bar{A} x + w, \\ w \geq \sum_{n \in K} p_n \left(q^+ + q^- \right)^T \left(b\left(\omega_n \right) - A\left(\omega_n \right) x \right), \quad \text{for all } K \subset \mathcal{M}, \\ w \geq 0, \\ x \in \mathbb{X}, \end{cases} \tag{16}$$

where $\mathcal{M} = \{1, \ldots, N\}$.

This problem is even harder than the original linear program (15) as there are 2^N constraints, but Benders (1962) showed that the above problem can be solved through a sequence of expanding relaxed problems

$$\begin{cases} \min_{x,w} c^T x + \left(q^- \right)^T \bar{A} x + w, \\ w \geq \sum_{n \in K_k} p_n \left(q^+ + q^- \right)^T \left(b\left(\omega_n \right) - A\left(\omega_n \right) x \right), \quad k = 1, \ldots, \nu, \\ w \geq 0, \\ x \in \mathbb{X}, \end{cases} \tag{17}$$

ν is the number of steps, $K_k \subset \mathcal{M}$ are constraints added in step k with $K_k \neq K_l$ when $k \neq l$. Hence, each successive relaxed problem adds more constraints to those already present in the previous steps.

Let

$$\eta_k = \sum_{n \in K_k} p_n \left(q^+ + q^- \right)^T A\left(\omega_n \right), \quad \zeta_k = \sum_{n \in K_k} p_n \left(q^+ + q^- \right)^T b\left(\omega_n \right).$$

Then (17) is written as the linear program

$$\begin{cases} \min_{x,w} c^T x + \left(q^- \right)^T \bar{A} x + w, \\ w \geq \zeta_k - \eta_k x, \quad k = 1, \ldots, \nu, \\ w \geq 0, \\ x \in \mathbb{X}. \end{cases} \tag{18}$$

The complete algorithm is as follows:

Step 1. Initialization: set $K_1 = \mathcal{M}$, $\eta_1 = \sum_{n=1}^{N} p_n (q^+ + q^-)^T A(\omega_n)$, $\zeta_1 = \sum_{n=1}^{N} p_n (q^+ + q^-)^T b(\omega_n)$ and $\nu = 1$. Choose computation accuracy ε.

Step 2. Solve problem (18) and denote the solution by (x^*, w^*). Put

$$\begin{aligned} K^* &= \left\{ n \in \mathcal{M} \colon \left(q^+ + q^- \right)^T \left(b\left(\omega_n \right) - A\left(\omega_n \right) x^* \right) \geq 0 \right\}, \\ w_+^* &= \sum_{n \in K^*} p_n \left(q^+ + q^- \right)^T \left(b\left(\omega_n \right) - A\left(\omega_n \right) x^* \right), \quad \hat{w}^* = \zeta_\nu - \eta_\nu x^*. \end{aligned}$$

Compute

$$\underline{F} = c^T x^* + \left(q^- \right)^T \bar{A} x^* + w_+^*, \quad \overline{F} = c^T x^* + \left(q^- \right)^T \bar{A} x^* + \hat{w}^*.$$

Step 3. If $(\overline{F} - \underline{F}) \leq \varepsilon$ then **stop**. x^* is an optimal solution.

Step 4. Set $\nu = \nu + 1$, $K_\nu = K^*$. Compute $\eta_\nu = \sum_{n \in K_\nu} p_n (q^+ + q^-)^T A(\omega_n)$ and $\zeta_\nu = \sum_{n \in K_\nu} p_n (q^+ + q^-)^T b(\omega_n)$. Add this new constraint to the set of constraints and go to Step 2.

The algorithm written as an R script can be seen on Fig. 1.

Set $\nu = 0$ and $K_1 = \mathcal{M}$.
Put $cmat = (c + (q^-)^T \bar{A}, 1)$, $Amat = NULL$, $bmat = NULL$

repeat
  Put $\nu = \nu + 1$
  Compute $\eta_\nu = \sum_{n \in K_\nu} p_n (q^+ + q^-)^T A(\omega_n)$ and $\zeta_\nu = \sum_{n \in K_\nu} p_n (q^+ + q^-)^T b(\omega_n)$
  Put $Amat = rbind(Amat, (-\eta_\nu, \zeta_\nu, -1))$, $bmat = c(bmat, 0)$
  Solve the linear problem $\min_{(x,w)} cmat^T * (x, w)$ with the constraints $Amat * (x, w)^T \leq bmat^T$ and denote its solution by (x^*, w^*).
  Put

$$K^* = \left\{ n \in \mathcal{M} \colon \left(q^+ + q^-\right)^T \left(b\left(\omega_n\right) - A\left(\omega_n\right) x^*\right) \geq 0 \right\},$$

$$w_+^* = \sum_{n \in K^*} p_n \left(q^+ + q^-\right)^T \left(b\left(\omega_n\right) - A\left(\omega_n\right) x^*\right), \quad \hat{w}^* = \zeta_\nu - \eta_\nu x^*.$$

  Set $K_{\nu+1} = K^*$
  Compute

$$\underline{F} = c^T x^* + \left(q^-\right)^T \bar{A} x^* + w_+^*, \quad \overline{F} = c^T x^* + \left(q^-\right)^T \bar{A} x^* + \hat{w}^*.$$

while $(\overline{F} - \underline{F}) > \varepsilon$

Output: x^*

Figure 1: Benders decomposition algorithm for a simple recourse problem

Examples

Conditional value at risk (CVaR)

We begin with an example of mean-CVaR optimization for a discrete distribution of returns. Our goal is to find portfolio u which solves

$$\begin{cases} \mathrm{CVaR}_\alpha \left(u^T R\right) \to \min, \\ u^T \mathbb{E}\left[R\right] \geq r_0, \\ u \in \mathbb{X}. \end{cases} \tag{19}$$

As is shown in Section LP computable portfolio problems, this problem is reduced to the linear program

$$\begin{cases} \xi + \frac{1}{1-\alpha} \sum_{n=1}^N p_n y_n \to \min, \\ y_n \geq 0, \quad n = 1, \ldots, N, \\ y_n + \xi + r_n^T u \geq 0, \quad n = 1, \ldots, N, \\ u^T \mu \geq r_0, \\ u \in \mathbb{X}, \end{cases} \tag{20}$$

where $\mu = \mathbb{E}[R]$. Benders decomposition for the above problem leads to the sequence of expanding relaxed problems

$$\begin{cases} \min_{u,\xi,w} \xi + \frac{1}{1-\alpha} w, \\ \sum_{n \in K_k} p_n \left(\xi + r_n^T u\right) + w \geq 0, \quad k = 1, \ldots, \nu, \\ u^T \mu \geq r_0, \\ u \in \mathbb{X}. \end{cases} \tag{21}$$

Using notation from Section Benders decomposition we identify $x = (u, \xi)$, $q^- = 0$, $(q^+)^T A(\omega_n) = (r_n, 1)$, $b(\omega_n) = 0$ and $c = (0_J, 1)$, where 0_J is the zero vector of length J. The algorithm can be readily applied.

Mean absolute deviation(MAD) and lower semi-absolute deviation (LSAD)

Due to the relation $LSAD(u^T R) = \frac{1}{2} MAD(u^T R)$ it is sufficient to consider only the mean-LSAD portfolio optimization:

$$\begin{cases} \text{LSAD}\left(u^T R\right) \to \min, \\ u^T \mu \geq r_0, \\ u \in \mathbb{X}. \end{cases} \tag{22}$$

As before, for a discrete set of returns that problem can be formulated as a linear program (see equation (9)) and Benders decomposition can be applied leading to a sequence of expanding relaxed problems. Introducing the new variable

$$\eta_k = \sum_{n \in K_k} p_n \hat{r}_n,$$

we reduce the problem to the linear program

$$\begin{cases} \min_{u,w} w, \\ \eta_k^T u + w \geq 0, \quad k = 1, \ldots, \nu, \\ \mu^T u \geq r_0, \\ u \in \mathbb{X}. \end{cases} \tag{23}$$

Similarly as for CVaR, we recognize in that formulation the Benders problem from Section Benders decomposition.

Projection algorithm for portfolio optimization

Algorithm

As we have discussed earlier, some mean-risk optimization problems can be transformed into linear programs. Those linear programs can be written down in the standard form

$$\begin{aligned} & c^T x \to \min, \\ & Ax \geq b, \\ & x \geq 0, \\ & x \in \mathbb{R}^n. \end{aligned} \tag{24}$$

In practical computations, it appears that solutions to the above linear program are sometimes non-unique and form a (multidimensional) simplex. A typical linear solver find only one of the solutions – usually one of the vertices of the solution simplex. In this section our aim is to find a solution to the above linear program which is closest to a given vector. It often lies in the interior of a face of the solution simplex.

Let S^* denote the set of optimal solutions to (24), B be a given invertible matrix in $\mathbb{R}^n$ and $\hat{x}$ a vector from $\mathbb{R}^n$. We want to find $x^* \in S^*$ such that

$$\|B\left(x^* - \hat{x}\right)\| \leq \|B\left(x - \hat{x}\right)\| \text{ for all } x \in S^*. \tag{25}$$

The least norm solution to linear program mentioned in the Introduction is a special case of problem (24–25). It corresponds to $\hat{x} = 0$ and B being an identity matrix. Finding an optimal portfolio closest to a given benchmark in the framework of LP problems can also be reduced, after some modification, to the solution of problem (24–25). To describe this modification observe that for LP computable portfolio problems the independent variable x in (24) contains more coordinates that only

portfolio weights. Let $x = (x', u)$ where u corresponds to portfolio weights and x' contains all other coordinates (for CVaR optimization (3) $x = (\xi, y, u)$, hence $x' = (\xi, y)$, for MAD optimization (6) and LSAD optimization (9) $x = (y, u)$). Given the benchmark portfolio $\hat{u}$, the objective is to find an optimal portfolio which minimizes the distance $\|u - \hat{u}\|$. To achieve that goal by solving problem (24–25) we have to extend $\hat{u}$ to a vector $\hat{x} = (\hat{x}', \hat{u})$ on the whole space $\mathbb{R}^n$ taking arbitrary $\hat{x}'$ and projecting the difference $(x - \hat{x})$ on the subspace spanned by the portfolio coordinates. Let B^* be this projection operator, then the aim is to minimize $\|B^*(x - \hat{x})\|$. Matrix B^* is diagonal with entries 1 on the diagonal positions corresponding to portfolio weights and 0 otherwise. Since B^* is not invertible the considered portfolio problem cannot be reduced to the solution of problem (24–25). Hence we introduce a matrix B replacing in B^* zero diagonal entries with some small positive number ε. Then B is invertible and an optimal portfolio close to a benchmark can be found by solving problem (24–25). One can further improve the accuracy of the computation by redoing the optimization with $\hat{x} = (x'^*, \hat{u})$, where x'^* is the solution obtained in the first optimization.

To solve problem (24–25) we follow the approach by Zhao and Li (2002) based on the path-following algorithm. We begin with the reformulation of (24) as the logarithmic barrier problem

$$\begin{aligned} & c^T x - \rho \left(\sum_{i=1}^{n} \log x_i + \sum_{i=1}^{m} \log z_i \right) \to \min, \\ & Ax - z = b, \\ & x, z > 0, \\ & x \in \mathbb{R}^n, z \in \mathbb{R}^m, \end{aligned} \tag{26}$$

where z is introduced to replace the inequality $Ax \geq b$ by equality and $\rho > 0$ is a regularizing parameter.

The Lagrangian for this problem reads

$$L(x, y, z) = c^T x + y^T (z - Ax + b) - \rho \left(\sum_{i=1}^{n} \log x_i + \sum_{i=1}^{m} \log z_i \right)$$

and the Kuhn-Tucker solvability conditions are

$$\begin{aligned} & \operatorname{diag}(x)\, s = \rho e, \\ & \operatorname{diag}(z)\, y = \rho e, \\ & s + A^T y - c = 0, \\ & z - Ax + b = 0, \end{aligned} \tag{27}$$

where $\operatorname{diag}(u)$ denotes the diagonal matrix with vector u on diagonal; the new variable $s = \rho \operatorname{diag}(x)^{-1} e$ is introduced to obtain the canonical representation of the central path approach; and $e = (1, \ldots, 1)$ (of an appropriate dimension).

To find the solution to (26) which fulfills the condition $\|B(x - \hat{x})\|^2 \to \min$ for a given vector $\hat{x}$ and matrix B, we modify the Lagrangian

$$L_P(x, y, z) = c^T x + y^T (z - Ax + b) - \rho \left(\sum_{i=1}^{n} \log x_i + \sum_{i=1}^{m} \log z_i \right) + \frac{1}{2} \theta \left(\|B(x - \hat{x})\|^2 - \|y\|^2 \right).$$

The stationary point of this Lagrangian gives the primal solution for which $\|B(x - \hat{x})\|^2$ achieves minimal value and the dual solution which has minimal norm. The Kuhn-Tucker conditions give

$$\begin{aligned} & \operatorname{diag}(x)\, s = \rho e, \\ & \operatorname{diag}(z)\, y = \rho e, \\ & s + A^T y - c = \theta B^T B (x - \hat{x}), \\ & z - Ax + b = \theta y. \end{aligned} \tag{28}$$

To simplify the problem we assume that $\theta = \kappa \rho^p$ for given constants $\kappa \in (0, 1]$ and $p \in (0, 1)$. We define the nonlinear mapping

$$F_\rho(x, y, s, z) = \begin{pmatrix} \operatorname{diag}(x)\, s - \rho e \\ \operatorname{diag}(z)\, y - \rho e \\ s + A^T y - c - \kappa \rho^p B^T B (x - \hat{x}) \\ z - Ax + b - \kappa \rho^p y \end{pmatrix}. \tag{29}$$

The problem is now to find $(x^*, y^*, s^*, z^*) \geq 0$ which solve the equation

$$F_0\left(x^*, y^*, s^*, z^*\right) = 0. \tag{30}$$

The solution to this equation is searched by the Newton iterative method

$$F_{\rho_k}\left(x_k, y_k, s_k, z_k\right) + \nabla F_{\rho_k}\left(x_k, y_k, s_k, z_k\right)\left(\Delta x, \Delta y, \Delta s, \Delta z\right) = 0, \tag{31}$$

where $\nabla F_\rho(x, y, s, z)$ denotes the gradient of function F_ρ given by the expression

$$\nabla F_\rho\left(x, y, s, z\right) = \begin{pmatrix} \operatorname{diag}(s) & 0 & \operatorname{diag}(x) & 0 \\ 0 & \operatorname{diag}(z) & 0 & \operatorname{diag}(y) \\ -\kappa\rho^p B^T B & A^T & \mathbf{1} & 0 \\ -A & -\kappa\rho^p \mathbf{1} & 0 & \mathbf{1} \end{pmatrix}. \tag{32}$$

These iterative solutions form the path-following algorithm in a neighborhood of the regularized central path

$$\mathcal{N}_\beta(\rho) = \left\{(x, y, s, z) : \left\|F_\rho\left(x, y, s, z\right)\right\|_\infty \leq \beta\rho\right\},$$

with $\beta \in (0, 1)$.

The complete algorithm is as follows:

Central path projection algorithm

Step 1. Initialization. Set $\beta \in (0, 1)$ and assign scalars b_1, b_2 and σ in $(0, 1)$. Select $(x_0, y_0, s_0, z_0) > 0$, $\kappa \in (0, 1)$, $p \in (0, 1)$ and $\rho_0 \in (1, \infty)$ such that $(x_0, y_0, s_0, z_0) \in \mathcal{N}_\beta(\rho_0)$.

Step 2. Newton's iterates. If $F_{\rho_k}(x_k, y_k, s_k, z_k) = 0$ put

$$\left(x_{k+1}, y_{k+1}, s_{k+1}, z_{k+1}\right) = \left(x_k, y_k, s_k, z_k\right)$$

and go to Step 3.
Otherwise, find $(\Delta x, \Delta y)$ which solve the following linear system

$$\begin{aligned} &\begin{pmatrix} \operatorname{diag}(s_k) + \kappa\left(\rho_k\right)^p B^T B \operatorname{diag}(x_k) & -\operatorname{diag}(x_k) A^T \\ \operatorname{diag}(y_k) A & \operatorname{diag}(z_k) + \kappa\left(\rho_k\right)^p \operatorname{diag}(y_k) \end{pmatrix} \begin{pmatrix} \Delta x \\ \Delta y \end{pmatrix} \\ &= \begin{pmatrix} \rho_k e - \operatorname{diag}(x_k) s_k \\ \rho_k e - \operatorname{diag}(z_k) y_k \end{pmatrix} - \begin{pmatrix} c \operatorname{diag}(x_k)\left(\kappa\left(\rho_k\right)^p B^T B\left(x_k - \hat{x}\right) - A^T y_k - s_k + c\right) \\ \operatorname{diag}(y_k)\left(A x_k + \kappa\left(\rho_k\right)^p y_k - z_k - b\right) \end{pmatrix}. \end{aligned} \tag{33}$$

Then set

$$\begin{aligned} \begin{pmatrix} \Delta s \\ \Delta z \end{pmatrix} &= \begin{pmatrix} \kappa\left(\rho_k\right)^p B^T B & -A^T \\ A & \kappa\left(\rho_k\right)^p \mathbf{1} \end{pmatrix} \begin{pmatrix} \Delta x \\ \Delta y \end{pmatrix} \\ &+ \begin{pmatrix} \kappa\left(\rho_k\right)^p B^T B\left(x_k - \hat{x}\right) - A^T y_k - s_k + c \\ A x_k + \kappa\left(\rho_k\right)^p y_k - z_k - b \end{pmatrix}. \end{aligned} \tag{34}$$

Find α such that $(x_k + \lambda\Delta x, y_k + \lambda\Delta y, s_k + \lambda\Delta s, z_k + \lambda\Delta z) > 0$ for all $\lambda \in (0, \alpha)$. Then find the maximal improvement step for λ by the Armijo rule: find the smallest j such that $\lambda_j = \alpha b_1^j$ and

$$\begin{aligned} &\left\|F_{\rho_k}\left(x_k + \lambda_j\Delta x, y_k + \lambda_j\Delta y, s_k + \lambda_j\Delta s, z_k + \lambda_j\Delta z\right)\right\|_\infty \\ &\leq \left(1 - \sigma\lambda_j\right)\left\|F_{\rho_k}\left(x_k, y_k, s_k, z_k\right)\right\|_\infty \end{aligned} \tag{35}$$

Set

$$\left(x_{k+1}, y_{k+1}, s_{k+1}, z_{k+1}\right) = \left(x_k, y_k, s_k, z_k\right) + \lambda_j\left(\Delta x, \Delta y, \Delta s, \Delta z\right)$$

and go to Step 3.

Step 3. Reduction of ρ. Find the maximal improvement step for ρ_k by the Armijo rule: find the smallest j such that $\gamma_j = b_2^j$ and

$$\left\|F_{(1-\gamma_j)\rho_k}\left(x_{k+1}, y_{k+1}, s_{k+1}, z_{k+1}\right)\right\|_\infty \leq \beta\left(1 - \gamma_j\right)\rho_k.$$

Set $\rho_{k+1} = (1 - \gamma_j)\rho_k$ and go to Step 2.

We prove now that starting from the initial iterate defined in Step 1, we obtain a convergent sequence of iterates. The estimate of the norm F creates a difficulty because of the term ρ_k^p which for

$\rho_k < 1$ cannot be estimated by the first power of ρ_k. That requires such a choice of the starting point which cancels the terms with ρ^p. This goal is achieved by a proper choice of constant κ. The following lemma is an adaptation of Lemma 4.1 from Zhao and Li (2002).

Lemma B.4.1. *The central path projection algorithm is well-defined. The sequence ρ_k is monotonically decreasing and $(x_k, y_k, s_k, z_k) \in \mathcal{N}_\beta(\rho_k)$ for all $k \geq 0$.*

Proof. The proof goes along the lines of the original proof of Zhao and Li (2002). We have only to remark that the nonsingularity of matrix ∇F_ρ follows from the fact that the matrix

$$\begin{pmatrix} \kappa\rho^p B^T B & -A^T \\ A & \kappa\rho^p \mathbf{1} \end{pmatrix}$$

is positive semidefinite for $\rho > 0$.

For the correctness of Step 3 of the algorithm, we have to show that point $(x_{k+1}, y_{k+1}, s_{k+1}, z_{k+1})$ belongs to $\mathcal{N}_\beta(\rho_{k+1})$. To this end, we have to prove the estimate

$$\left\| F_{\rho_k}(x, y, s, z) - F_{\rho_l}(x, y, s, z) \right\|_\infty \leq (\rho_l - \rho_k) + \left((\rho_l)^p - (\rho_k)^p \right) \| (x - \hat{x}, y) \|_\infty ,$$

for $\rho_l \geq \rho_k$ and $(x, y, s, z) > 0$. Taking into account that $\kappa \leq 1$ that estimate follows from the definition of F

$$\begin{aligned} &\left\| F_{\rho_k}(x, y, s, z) - F_{\rho_l}(x, y, s, z) \right\|_\infty \\ &\leq (\rho_l - \rho_k) + \kappa \left((\rho_l)^p - (\rho_k)^p \right) \|y\|_\infty + \kappa \left((\rho_l)^p - (\rho_k)^p \right) \left\| B^T B (x - \hat{x}) \right\|_\infty \\ &\leq (\rho_l - \rho_k) + \left((\rho_l)^p - (\rho_k)^p \right) \| (x - \hat{x}, y) \|_\infty . \end{aligned}$$

The rest of the proof is similar as in Zhao and Li (2002). The boundedness of (x_k, y_k) follows from Lemma B.4.2 below. □

The proof of the convergence of the iterative sequence requires some modification of the proof by Zhao and Li (2002). The modification is formulated in the following lemma.

Lemma B.4.2. *When the solution set to equation (30) is nonempty, then the sequence (x_k, y_k, s_k, z_k) obtained by the central path projection algorithm is bounded.*

Proof. We follow the line of the proof of Theorem 4.1 in Zhao and Li (2002). Let (u_k, v_k, w_k, q_k) be defined as

$$(u_k, v_k, w_k, q_k) = \frac{1}{\rho_k} F_{\rho_k}(x_k, y_k, s_k, z_k) .$$

Then $\| (u_k, v_k, w_k, q_k) \|_\infty \leq \beta$ by the definition of $\mathcal{N}_\beta(\rho_k)$ and the following system of equations holds

$$\begin{aligned} &\operatorname{diag}(x_k)\, s_k = \rho_k (e + u_k) , \\ &\operatorname{diag}(y_k)\, z_k = \rho_k (e + v_k) , \\ &s_k = -A^T y_k + c + \kappa (\rho_k)^p B^T B (x_k - \hat{x}) + \rho_k w_k, \\ &z_k = A x_k - b + \kappa (\rho_k)^p y_k + \rho_k q_k. \end{aligned} \tag{36}$$

Let (x^*, y^*, s^*, z^*) be an optimal solution, i.e. a solution to equation (30). Then $(x^*, y^*, s^*, z^*) \geq 0$ and (s^*, z^*) is given by the expression

$$\begin{pmatrix} s^* \\ z^* \end{pmatrix} = \begin{pmatrix} 0 & -A^T \\ A & 0 \end{pmatrix} \begin{pmatrix} x^* \\ y^* \end{pmatrix} + \begin{pmatrix} c \\ -b \end{pmatrix} . \tag{37}$$

Due to (28) $(x^*)^T s^* = 0$ and $(y^*)^T z^* = 0$.

In what follows, we use the positive semidefiniteness

$$\begin{pmatrix} x \\ y \end{pmatrix}^T \begin{pmatrix} \kappa\rho^p B^T B & -A^T \\ A & \kappa\rho^p \mathbf{1} \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix} = \kappa\rho^p \|Bx\|_2^2 + \kappa\rho^p \|y\|_2^2 . \tag{38}$$

Then we have

$$
\begin{aligned}
0 \leq & \begin{pmatrix} x^* \\ y^* \end{pmatrix}^T \begin{pmatrix} s_k \\ z_k \end{pmatrix} + \begin{pmatrix} s^* \\ z^* \end{pmatrix}^T \begin{pmatrix} x_k \\ y_k \end{pmatrix} \\
= & \begin{pmatrix} x^* - x_k \\ y^* - y_k \end{pmatrix}^T \left(\begin{pmatrix} \kappa (\rho_k)^p B^T B & -A^T \\ A & \kappa (\rho_k)^p \mathbf{1} \end{pmatrix} \begin{pmatrix} x_k \\ y_k \end{pmatrix} + \begin{pmatrix} c \\ -b \end{pmatrix} \right. \\
& \left. + \rho_k \begin{pmatrix} w_k \\ q_k \end{pmatrix} - \begin{pmatrix} \kappa (\rho_k)^p B^T B \hat{x} \\ 0 \end{pmatrix} \right) + \begin{pmatrix} s^* \\ z^* \end{pmatrix}^T \begin{pmatrix} x_k \\ y_k \end{pmatrix} + \begin{pmatrix} x_k \\ y_k \end{pmatrix}^T \begin{pmatrix} s_k \\ z_k \end{pmatrix} \\
= & - \begin{pmatrix} x_k - x^* \\ y_k - y^* \end{pmatrix}^T \begin{pmatrix} \kappa (\rho_k)^p B^T B & -A^T \\ A & \kappa (\rho_k)^p \mathbf{1} \end{pmatrix} \begin{pmatrix} x_k - x^* \\ y_k - y^* \end{pmatrix} \\
& - \begin{pmatrix} x_k - x^* \\ y_k - y^* \end{pmatrix}^T \left(\begin{pmatrix} \kappa (\rho_k)^p B^T B & -A^T \\ A & \kappa (\rho_k)^p \mathbf{1} \end{pmatrix} \begin{pmatrix} x^* \\ y^* \end{pmatrix} + \begin{pmatrix} c \\ -b \end{pmatrix} \right. \\
& \left. + \rho_k \begin{pmatrix} w_k \\ q_k \end{pmatrix} - \begin{pmatrix} \kappa (\rho_k)^p B^T B \hat{x} \\ 0 \end{pmatrix} \right) + \begin{pmatrix} x_k \\ y_k \end{pmatrix}^T \begin{pmatrix} s^* \\ z^* \end{pmatrix} + \begin{pmatrix} x_k \\ y_k \end{pmatrix}^T \begin{pmatrix} s_k \\ z_k \end{pmatrix} \\
= & - \kappa (\rho_k)^p \left\| \begin{matrix} B (x_k - x^*) \\ y_k - y^* \end{matrix} \right\|_2^2 - \kappa (\rho_k)^p \begin{pmatrix} x_k - x^* \\ y_k - y^* \end{pmatrix}^T \begin{pmatrix} B^T B x^* + (\rho_k)^{1-p} \kappa^{-1} w_k - B^T B \hat{x} \\ y^* + (\rho_k)^{1-p} \kappa^{-1} q_k \end{pmatrix} \\
& + \begin{pmatrix} x_k \\ y_k \end{pmatrix}^T \begin{pmatrix} s_k \\ z_k \end{pmatrix}.
\end{aligned}
\tag{39}
$$

From equation (36) we obtain

$$
(x_k)^T s_k = \rho_k e^T (e + u_k), \quad (y_k)^T z_k = \rho_k e^T (e + v_k)
$$

and the estimate

$$
\left| \begin{pmatrix} x_k \\ y_k \end{pmatrix}^T \begin{pmatrix} s_k \\ z_k \end{pmatrix} \right| \leq \rho_k c
$$

for some positive constant c.

Finally we obtain

$$
\left\| \begin{matrix} B (x_k - x^*) \\ y_k - y^* \end{matrix} \right\|_2^2 \leq \left\| \begin{matrix} x_k - x^* \\ y_k - y^* \end{matrix} \right\|_2 \left\| \begin{matrix} B^T B (x^* - \hat{x}) + (\rho_k)^{1-p} \kappa^{-1} w_k \\ y^* + (\rho_k)^{1-p} \kappa^{-1} q_k \end{matrix} \right\|_2 + (\rho_k)^{1-p} \kappa^{-1} c. \tag{40}
$$

Since matrix B is invertible, we have

$$
\|x\|_2 \leq \left\| B^{-1} \right\| \|Bx\|_2 .
$$

Taking into account the above estimates and the estimate $\rho_k \leq \rho_0$, we obtain

$$
\left\| \begin{matrix} x_k - x^* \\ y_k - y^* \end{matrix} \right\|_2 \leq c. \tag{41}
$$

The boundedness of (s_k, z_k) follows from the above estimate and equation (36).

□

The complete algorithm is presented on Fig. 2. This algorithm requires an appropriate initial step: ρ_0 and a starting point $(x_0, y_0, s_0, z_0) \in \mathcal{N}_\beta(\rho_0)$. Hence we choose $(x_0, y_0, s_0, z_0) > 0$ such that $\|F_{\rho_0}(x_0, y_0, s_0, z_0)\|_\infty \leq \beta\rho_0$, which guarantees that $(x_0, y_0, s_0, z_0) \in \mathcal{N}_\beta(\rho_0)$.

The choice of (x_0, y_0, s_0, z_0) starts with $y_0 = e$ and $z_0 = \kappa\rho_0^p e$. We choose x_0 and s_0 so that the term $\kappa\rho_0^p B^T B(x_0 - \hat{x})$ is canceled. Taking $x_0 = \max(e, e + \hat{x})$ makes $x_0 > 0$. On the other hand, $\kappa\rho_0^p B^T B(x_0 - \hat{x})$ is a vector with nonzero components bounded from above by $\kappa\rho_0^p(1 + \|\hat{x}^-\|)$. Hence we define $s_{0,init} = B^T B(x_0 - \hat{x})$ and compute $\kappa = 1/\| \operatorname{diag}(x_0) s_{0,init} \|_\infty$. Then we take $s_0 = \kappa\rho_0^p s_{0,init}$. That procedure eliminates from $\|F_{\rho_0}\|_\infty$ the terms with ρ_0^p and replaces them by constants. We can now describe the construction of the initial point step-by-step:

1. Set $y_0 = e$.
2. Set $x_0 = \max(e, e + \hat{x})$.
3. Take $\rho_0 = \max\left(1, \left\| \begin{matrix} cA^T y_0 - c \\ -Ax_0 + b \end{matrix} \right\|_\infty\right) + \delta$

Input: Select positive constants: $p, \beta, b_1, b_2, \sigma \in (0,1)$. Select initial values ρ_0, $\kappa \in (0,1]$ and $(x_0, y_0, s_0, z_0) > 0$ such that $(x_0, y_0, s_0, z_0) \in \mathcal{N}_\beta(\rho_0)$

Step 1

If $F_{\rho_k}(x_k, y_k, s_k, z_k) = 0$

set $(x_{k+1}, y_{k+1}, s_{k+1}, z_{k+1}) = (x_k, y_k, s_k, z_k)$ and go to Step 3

Otherwise solve the linear system

$$\begin{pmatrix} \operatorname{diag}(s_k) + \kappa\rho_k^p \operatorname{diag}(x_k) B^T B & -\operatorname{diag}(x_k) A^T \\ \operatorname{diag}(y_k) A & \operatorname{diag}(z_k) + \kappa\rho_k^p \operatorname{diag}(y_k) \end{pmatrix} \begin{pmatrix} \Delta x \\ \Delta y \end{pmatrix}$$
$$= \begin{pmatrix} \rho_k e - \operatorname{diag}(x_k) \left(\kappa\rho_k^p \operatorname{diag}(x_k) B^T B (x_k - \hat{x}) - A^T y_k + c \right) \\ \rho_k e - \operatorname{diag}(y_k) \left(A x_k + \kappa\rho_k^p y_k - b \right) \end{pmatrix}$$

and set

$$\begin{pmatrix} \Delta s \\ \Delta z \end{pmatrix} = \begin{pmatrix} \kappa\rho_k^p B^T B & -A^T \\ A & \kappa\rho_k^p \mathbf{1} \end{pmatrix} \begin{pmatrix} \Delta x \\ \Delta y \end{pmatrix} + \begin{pmatrix} \kappa\rho_k^p B^T B (x_k - \hat{x}) - A^T y_k - s_k + c \\ A x_k + \kappa\rho_k^p y_k - z_k - b \end{pmatrix}.$$

Step 2

Find the step size λ_k such that

$(x_{k+1}, y_{k+1}, s_{k+1}, z_{k+1}) = (x_k + \lambda_k \Delta x, y_k + \lambda_k \Delta y, s_k + \lambda_k \Delta s, z_k + \lambda_k \Delta z) \in \mathcal{N}_\beta(\rho_k)$

This is achieved in 2 substeps

substep 1

Find α such that $\forall \lambda \in (0, \alpha)$ $(x_k + \lambda \Delta x, y_k + \lambda \Delta y, s_k + \lambda \Delta s, z_k + \lambda \Delta z) > 0$

substep 2

Let $\lambda_k = \alpha b_1^j$, where j is the smallest integer such that

$$\left\| F_{\rho_k}(x_k + \lambda_k \Delta x, y_k + \lambda_k \Delta y, s_k + \lambda_k \Delta s, z_k + \lambda_k \Delta z) \right\|_\infty \leq (1 - \sigma\lambda_k) \left\| F_{\rho_k}(x_k, y_k, s_k, z_k) \right\|_\infty.$$

Step 3

Find the smallest $\rho_{k+1} < \rho_k$ such that $(x_{k+1}, y_{k+1}, s_{k+1}, z_{k+1}) \in \mathcal{N}_\beta(\rho_{k+1})$.

Let $\rho_{k+1} = (1 - b_2^j)\rho_k$ where j is the smallest integer for which we have

$$\left\| F_{\rho_{k+1}}(x_{k+1}, y_{k+1}, s_{k+1}, z_{k+1}) \right\|_\infty \leq \beta\rho_{k+1}.$$

Stopping criterion: $\rho_{k+1} \leq tol$ or $\|F_{\rho_{k+1}}(x_{k+1}, y_{k+1}, s_{k+1}, z_{k+1})\|_\infty \leq tol$.

Figure 2: Regularized central path algorithm with projection

4. Set $s_{0,init} = B^T B(x_0 - \hat{x})$.
5. Compute $\kappa = 1/\| \operatorname{diag}(x_0) s_{0,init} \|_\infty$.
6. Set $z_0 = \kappa \rho_0^p e$.
7. Set $s_0 = \kappa \rho_0^p s_{0,init}$.
8. Compute
$$\xi = \frac{\left\| F_{\rho_0}(x_0, y_0, s_0, z_0) \right\|_\infty}{\rho_0}.$$
9. Set $\beta \in (\xi, 1)$.

Remark B.4.3. Due to the above choice of the initial point we have the estimates

$$\begin{aligned}
&\|\operatorname{diag}(x_0)\, s_0 - \rho_0 e\|_\infty \le \left|\rho_0 - \rho_0^p\right| \le \rho_0, \\
&\|\operatorname{diag}(z_0)\, y_0 - \rho_0 e\|_\infty \le \kappa \left|\rho_0 - \rho_0^p\right| + (1-\kappa)\rho_0 \le \rho_0, \\
&\left\| s_0 + A^T y_0 - c - \kappa \rho_0^p B^T B(x_0 - \hat{x}) \right\|_\infty \le \left\| A^T y_0 - c \right\|_\infty, \\
&\left\| z_0 - A x_0 + b - \kappa \rho_0^p y_0 \right\|_\infty \le \| -A x_0 + b \|_\infty .
\end{aligned}$$

Hence

$$\left\| F_{\rho_0}(x_0, y_0, s_0, z_0) \right\|_\infty \le \rho_0$$

and for a properly chosen $\delta > 0$ the above inequality is sharp and $\xi < 1$. Then taking $\beta \in (\xi, 1)$ we obtain the desired estimate $\|F_{\rho_0}(x_0, y_0, s_0, z_0)\|_\infty \le \beta \rho_0$.

The computational complexity of our algorithm arises from the solution of the $(n+m)$-dimensional system (33). We can reduce the dimension of this system observing that it can be written as

$$\begin{aligned}
&\left(\operatorname{diag}(s_k) + \kappa (\rho_k)^p B^T B \operatorname{diag}(x_k)\right) \Delta x - \operatorname{diag}(x_k) A^T \Delta y \\
&\qquad = \rho_k e - \operatorname{diag}(x_k) \left(\kappa (\rho_k)^p B^T B (x_k - \hat{x}) - A^T y_k + c\right), \\
&\operatorname{diag}(y_k) A \Delta x + \left(\operatorname{diag}(z_k) + \kappa (\rho_k)^p \operatorname{diag}(y_k)\right) \Delta y \\
&\qquad = \rho_k e - \operatorname{diag}(y_k) \left(A x_k + \kappa (\rho_k)^p y_k - b\right).
\end{aligned} \tag{42}$$

When $n \ge m$ eliminating Δx gives the equation

$$\begin{aligned}
H_k \Delta y = &\rho_k e - \operatorname{diag}(y_k) \left(A x_k + \kappa (\rho_k)^p y_k - b\right) - \operatorname{diag}(y_k) A \\
&\times \left(\operatorname{diag}(s_k) + \kappa (\rho_k)^p \operatorname{diag}(x_k) B^T B\right)^{-1} \\
&\times \left(\rho_k e - \operatorname{diag}(x_k) \left(\kappa (\rho_k)^p B^T B (x_k - \hat{x}) - A^T y_k + c\right)\right),
\end{aligned} \tag{43}$$

where

$$\begin{aligned}
H_k = &\operatorname{diag}(z_k) + \kappa (\rho_k)^p \operatorname{diag}(y_k) \\
&+ \operatorname{diag}(y_k) A \left(\operatorname{diag}(s_k) + \kappa (\rho_k)^p \operatorname{diag}(x_k) B^T B\right)^{-1} \operatorname{diag}(x_k) A^T.
\end{aligned} \tag{44}$$

For Δx we obtain then

$$\begin{aligned}
\Delta x = &\left(\operatorname{diag}(s_k) + \kappa (\rho_k)^p \operatorname{diag}(x_k) B^T B\right)^{-1} \\
&\times \left(\operatorname{diag}(x_k) A^T \Delta y + \rho_k e - \operatorname{diag}(x_k) \left(\kappa (\rho_k)^p B^T B (x_k - \hat{x}) - A^T y_k + c\right)\right).
\end{aligned} \tag{45}$$

For $m > n$ we can eliminate Δy to obtain

$$\begin{aligned}
M_k \Delta x = &\rho_k e - \operatorname{diag}(x_k) \left(\kappa (\rho_k)^p B^T B (x_k - \hat{x}) - A^T y_k + c\right) + \operatorname{diag}(x_k) A^T \\
&\times \left(\operatorname{diag}(z_k) + \kappa (\rho_k)^p \operatorname{diag}(y_k)\right)^{-1} \left(\rho_k e - \operatorname{diag}(y_k) \left(A x_k + \kappa (\rho_k)^p y_k - b\right)\right),
\end{aligned} \tag{46}$$

where

$$\begin{aligned}
M_k = &\operatorname{diag}(s_k) + \kappa (\rho_k)^p B^T B \operatorname{diag}(x_k) \\
&+ \operatorname{diag}(x_k) A^T \left(\operatorname{diag}(z_k) + \kappa (\rho_k)^p \operatorname{diag}(y_k)\right)^{-1} \operatorname{diag}(y_k) A.
\end{aligned} \tag{47}$$

For Δy we obtain then

$$\begin{aligned}\Delta y = &\left(\operatorname{diag}(z_k) + \kappa(\rho_k)^p \operatorname{diag}(y_k)\right)^{-1} \\ &\times \left(-\operatorname{diag}(y_k) A\Delta x + \rho_k e - \operatorname{diag}(y_k)\left(Ax_k + \kappa(\rho_k)^p y_k - b\right)\right).\end{aligned} \tag{48}$$

Examples

We show now how the general projection algorithm described above can be used to find an optimal portfolio with the smallest distance to a given benchmark portfolio. We begin with the mean-CVaR optimization of Section LP computable portfolio problems. For simplicity, we take $\mathbb{X} = \{u : u \geq 0\}$. Then the linear program corresponding to this problem reads

$$\begin{cases} \xi + \frac{1}{1-\alpha}\sum_{n=1}^{N} p_n y_n \to \min, \\ y_n \geq 0, \quad n = 1, \ldots, N, \\ y_n + \xi + r_n^T u \geq 0, \quad n = 1, \ldots, N, \\ u^T \mu \geq r_0, \\ u \geq 0. \end{cases} \tag{49}$$

Given the benchmark portfolio w_b, we are looking for a solution of the above linear program with the additional constraint

$$\|u - w_b\|_2 \to \min. \tag{50}$$

A solution to this problem can be obtained by the central path projection algorithm. To this end, we define $x = (u, \xi, y)$, $c = (0_J, 1, p)$, where 0_J is the zero vector of length J and p is the N dimensional vector with entries p_n. Matrix A and vector b are given by the expressions

$$A = \begin{pmatrix} r & 0_N^T & \operatorname{diag}(e_N) \\ \mu & 0 & 0_N \end{pmatrix}, \quad b = \begin{pmatrix} c0_N^T \\ r_0 \end{pmatrix}, \tag{51}$$

where r is the $N \times J$ matrix of discrete returns, 0_N denotes the row vector of length N with zero entries and e_N – the row vector of length N with all entries equal 1. We take $\hat{x} = (w_b, 0, 0_N)$ and

$$B = \begin{pmatrix} \operatorname{diag}(e_J) & 0 \\ 0 & \operatorname{diag}(\varepsilon_{N+1}) \end{pmatrix}, \tag{52}$$

where ε_k is the row vector of length k with all entries equal ε.

With the above definitions, the solution obtained by the central path projection algorithm is an optimal solution to problem (49) with constraint (50).

For the mean-LSAD problem with $\mathbb{X} = \{u : u \geq 0\}$, the linear program is

$$\begin{cases} \sum_{n=1}^{N} p_n y_n \to \min, \\ y_n \geq 0, \quad n = 1, \ldots, N, \\ y_n + \hat{r}_n^T u \geq 0, \quad n = 1, \ldots, N, \\ u^T \mu \geq r_0, \\ u \geq 0. \end{cases} \tag{53}$$

To reduce this problem to the standard form appropriate for the central path projection algorithm, we define $x = (u, y)$, $c = (0_J, p)$, $\hat{x} = (w_b, 0_N)$ and matrices

$$A = \begin{pmatrix} \hat{r} & \operatorname{diag}(e_N) \\ \mu & 0_N \end{pmatrix}, \quad b = \begin{pmatrix} 0_N^T \\ r_0 \end{pmatrix}, \quad B = \begin{pmatrix} \operatorname{diag}(e_J) & 0 \\ 0 & \operatorname{diag}(\varepsilon_N) \end{pmatrix}, \tag{54}$$

where $\hat{r}$ is the $N \times J$ matrix of discrete centered returns. Running the central path projection algorithm with the above defined vectors and matrices, we obtain an optimal solution to problem (53) with constraint (50).

Numerical examples

The algorithms described in Sections Benders decomposition and Projection algorithm for portfolio optimization are implemented in R in the package **PortfolioOptim**. The package offers two public functions:

- `BDportfolio_optim` – which performs portfolio optimization using Benders decomposition;

- `PortfolioOptimProjection` – which implements the projection algorithm described in Section Projection algorithm for portfolio optimization.

For solving LP subproblems in `BDportfolio_optim`, we use function `Rglpk_solve_LP` from the R-package **Rglpk**. All computations are carried out on a computer with Intel Core i5-5200U processor with 8 GB RAM running Linux operating system.

Benders decomposition algorithm

It has been already observed by Künzi-Bay and Mayer (2006) that a good discrete approximation of a continuous distribution of returns requires large samples. The size of the sample depends on the goal of optimization procedure. It has been shown in Künzi-Bay and Mayer (2006) that the value of the objective function is accurately estimated with samples of size 10 000–20 000. Our experience shows that estimation of the optimal portfolio weights requires much larger samples.

Consider first the model used in Künzi-Bay and Mayer (2006): they provide the vector of mean μ and the covariance matrix Σ for monthly returns of 5 assets (MSCI.CH, MSCI.E, MSCI.W, Pictet.Bond and JPM.Global), and assume that the asset returns have joint normal distribution. We represent this distribution by a random sample of size N and solve the optimization problem (21) in which we impose additional constrains taking $\mathbb{X} = \{u\colon \sum_{k=1}^{K} u_k = 1, u \geq 0\}$. In all computations we take the target portfolio return $r_0 = 0.005$ and the CVaR confidence level $\alpha = 0.95$.

First we initialize computations with Künzi-Bay and Mayer (2006) data.

```
library(mvtnorm)
library(PortfolioOptim)

generate_data_normal <- function (means, covmat, num)
{
    k <- ncol(covmat)
    sim_data <- rmvnorm  (n=num, mean = means, sigma=covmat)
    sim_data <- matrix(num, k, data = sim_data)
    colnames(sim_data) <- colnames(covmat)
    prob <- matrix(1/num,num,1)
    mod_returns <- cbind(sim_data, prob)
    return (mod_returns)
}

prepare_data_KM <- function ()
{
sample_cov <- matrix(5,5, data = c( 0.003059 ,  0.002556 , 0.002327 , 0.000095 , 0.000533,
 0.002556 ,  0.003384 , 0.002929 ,  0.000032 ,  0.000762,
 0.002327 ,  0.002929 , 0.003509 ,  0.000036 ,  0.000908,
 0.000095 ,  0.000032 , 0.000036 ,  0.000069  , 0.000048 ,
 0.000533 ,  0.000762 , 0.000908  , 0.000048  , 0.000564))
sample_mean <- c( 0.007417,  0.005822,   0.004236,  0.004231,   0.005534)
colnames(sample_cov) <- c("MSCI.CH", "MSCI.E", "MSCI.W", "Pictet.Bond", "JPM.Global")
return(list( sample_mean = sample_mean, sample_cov = sample_cov))
}
```

We perform two tests (both tests are run simultaneously). In the first test, we compare computational time for different sample sizes. In the second test, we compare the accuracy of the obtained optimal portfolios. We perform computations for different sample sizes; for each sample size we generate 10 independent samples and assess the mean and the variance of the running time and portfolio weights. The R code for these tests is as follows.

```
data_nA <- prepare_data_KM()
sample_cov <- data_nA$sample_cov
sample_mean <- data_nA$sample_mean
k <- ncol(sample_cov)
a0 <- rep(1,k)
Aconstr <- rbind(a0,-a0)
bconstr <- c(1+1e-8, -1+1e-8)
lbound <- rep(0,k)
ubound <- rep(1,k)
R0 = 0.005
```

```
ET <- NULL
weights <- NULL
repetition = 10
sample_size = 10 000  # also run for 100 000 and 1 000 0000

ptm <- proc.time()[3]
  for (i in 1:repetition ){
    mod_returns <- generate_data_normal (sample_mean, sample_cov, sample_size)
    res  <-  BDportfolio_optim (mod_returns, R0, risk = "CVAR",  alpha = 0.95,
            Aconstr, bconstr,lbound, ubound, maxiter = 200, tol = 1e-10 )
    ET <- c(ET, proc.time()[3] - ptm)
    ptm <- proc.time()[3]
    weights <- rbind(weights, t(res$theta))
  }

cat(''running time and its standard deviation \n'')
print(mean(ET))
print(sqrt(var(ET)))

cat(''optimal portfolio and confidence intervals of its weights \n'')
print((colMeans(weights))*100)
print(sqrt(apply(weights, 2, var))*100*4/sqrt(repetition))
```

Since standard LP solvers (used for comparison by Künzi-Bay and Mayer (2006)) are too memory demanding to run on our hardware, we report computational time only for our BDportfolio_optim function. Table 1 contains the average running times for 10 different realizations together with the standard deviations.

sample size	mean	st. dev.
10 000	0.0663	0.0085
100 000	0.4953	0.0363
1 000 000	4.518	0.1086

Table 1: Averaged running time (in sec.) and its standard deviation for computations with different sample sizes.

The average running time for samples of size 10 000 is slightly smaller than the value reported by Künzi-Bay and Mayer (2006) (they give the value 0.088 sec.) but it can be attributed to a slightly faster CPU. It is also visible from Table 1 that even for samples of size 1 000 000 the computational time is small enough for such sample sizes to be used in practical computations.

The estimates of optimal portfolio weights for different sample sizes are collected in Table 2. The 95% confidence intervals of portfolio weights are reported in brackets. If the difference between portfolio weights is larger that two standard deviations we conclude that the difference is statistically significant (this corresponds to 95% two-sided Student-t test whose critical value for the sample of size 10 is 2.228). The results of Table 2 show that samples of size 10 000 are much to small to produce reliable portfolio weights. The values for that sample size are statistically significantly different from the values for 1 000 000 samples. From the values of confidence intervals of portfolio weights we can conclude that the computation of optimal portfolio weights from samples of size 10 000 can give values with an error up to 50% (for all non-zero weights and the sample of size 10 000 the confidence intervals are of order of one half of the corresponding weights). This is a clear indication that to obtain reliable values of portfolio weights we have to use samples of size 1 000 000 or take the average of a large number of repetitions, which is computationally equivalent.

We analyze now the effect of the number of assets in portfolio on the accuracy of computations. To this end, we use the data-set `etfdata` from the R-package **parma**. For comparison we take two subsets: `etfdata[1:500,1:5]`(with 5 assets) and `etfdata[1:500,1:10]` (with 10 assets). We add to our code a new function computing the vector of means and covariance matrix for a data set with k assets and perform computations with $k = 5$ and $k = 10$.

```
library(parma)
library(xts)
```

	sample size		
assets	10 000	100 000	1 000 000
MSCI.CH	9.6 (4.33)	11.2 (1.45)	10.9 (0.39)
MSCI.E	0.0 (0.00)	0.0 (0.00)	0.0 (0.00)
MSCI.W	0.0 (0.00)	0.0 (0.00)	0.0 (0.00)
Pictet.Bond	53.6 (13.24)	56.2 (2.33)	56.8 (0.83)
JPM.Global	36.8 (11.41)	32.6 (2.37)	32.3 (0.74)

Table 2: Optimal portfolios for different sample sizes. 95% confidence intervals of portfolio weights given in parentheses (all values are in percentage points).

```
library(quantmod)

data(etfdata)
prepare_data <- function (k)
{
    quotData <- as.xts(etfdata[1:500, 1:k])
    retData = NULL
    for (i in 1:k)
      retData <- cbind(retData,weeklyReturn(quotData[,i], type='arithmetic'))

    colnames(retData) <- colnames(quotData)
    sample_cov <- cov(retData)
    sample_mean <- colMeans(retData)
    return(list(sample_mean = sample_mean, sample_cov = sample_cov))
}

  data_nA <- prepare_data(5)  # also run with prepare_data(10)
  sample_cov <- data_nA$sample_cov
  sample_mean <- data_nA$sample_mean
  k <- ncol(sample_cov)
  a0 <- rep(1,k)
  Aconstr <- rbind(a0,-a0)
  bconstr <- c(1+1e-8, -1+1e-8)
  lbound <- rep(0,k)
  ubound <- rep(1,k)

  R0 = 0.004
  sample_size = 10 000  # also run for 100 000 and 1 000 0000
  repetition = 100
  weights <- NULL
  for (i in 1:repetition ){
      returns <- generate_data_normal(sample_mean, sample_cov, sample_size)
      res  <-  BDportfolio_optim (returns, R0, risk = "CVAR",  alpha = 0.95,
              Aconstr, bconstr, lbound, ubound, maxiter = 200, tol = 1e-10 )
      weights <- rbind(weights, t(res$theta))
  }
  print(sum(sqrt(apply(weights, 2, var))*100*4/sqrt(repetition))/k)
```

For each data set we generate samples of size 10 000, 100 000 and 1 000 000. For each sample size and data set we compute optimal portfolios. These computations are repeated 100 times and the empirical variances of portfolio weights are calculated. Using these results we compute widths of 95% confidence intervals of portfolio weights. To facilitate comparison of results between 5 and 10 assets we report in Table 3 the average width of 95% confidence interval per asset, i.e. the sum of widths for each asset divided by the number of assets. Notice that there is no significant difference in

accuracy between portfolios of 5 and 10 assets. It can be interpreted as a kind of robustness of Benders decomposition algorithm. The results fit almost perfectly to the theoretical picture of the square root dependence of confidence intervals on sample size. The values in Table 3 for 1 000 000 samples confirm that with that sample size we can get almost perfect estimate of portfolio weights with the average confidence interval of 0.1–0.2% which is more than sufficient for practitioners.

	sample size		
assets number	10 000	100 000	1 000 000
5	0.97	0.29	0.09
10	1.18	0.42	0.15

Table 3: The average width of 95% confidence interval of portfolio weights (values are in percentage points).

Projection algorithm

In testing the projection algorithm described in Section Projection algorithm for portfolio optimization the following values of parameters are used $\delta = 0.5$, $p = 0.8$, $\sigma = 10^{-3}$, $b1 = 0.9$ and $b2 = 0.9$. The starting point in all computations is calculated by the procedure described in that Section. The algorithm is tested on simulated data generated from a normal distribution using the vectors of means and covariance matrices estimated from the previously used data sets `etfdata[1:500,1:5]` and `etfdata[1:500,1:10]` from the package **parma**. Samples of 125, 250 and 500 returns are generated assuming they are weekly returns (the vector of means and covariance matrix appropriately scaled) which corresponds to 2.5, 5 and 10 years of weekly data. For each sample we compute the vector of portfolio weights using the function PortfolioOptimProjection with tol = 1e-6 and taking as the benchmark the portfolio $1/J$, i.e. the portfolio with all weights equal to $1/J$, where J is the number of assets in the portfolio. The computation for each sample size and asset set is repeated 10 times.

The code is as follows

```
data_nA <- prepare_data(5) # also run with prepare_data(10)
sample_cov <- data_nA$sample_cov
sample_mean <- data_nA$sample_mean
k <- ncol(sample_cov)
a0 <- rep(1,k)
Aconstr <- rbind(a0,-a0)
bconstr <- c(1+1e-8, -1+1e-8)
lbound <- rep(0,k)
ubound <- rep(1,k)

w_m = rep(1/k,k)
R0 = 0.005
returns_size =125 # also run for 250 and 500
ET <- NULL
ptm <- proc.time()[3]
for (i in 1:10 ){
mod_returns <- generate_data_normal(sample_mean, sample_cov, returns_size )
res  <-  PortfolioOptimProjection (mod_returns, R0 , risk = "CVAR",  alpha = 0.95,
        w_m, Aconstr, bconstr,lbound, ubound, 800, tol = 1e-6 )
ET <- c(ET, proc.time()[3] - ptm)
ptm <- proc.time()[3]
}
cat(''Mean running time and its standard deviation \n'')
print(c( mean(ET), sqrt(var(ET))))
```

The average running time per one computation and its standard deviation is reported in Table 4.

Taking into account that the sample of size n corresponds to the matrix A of approximate dimension $n \times n$ the results of Table 4 are compatible with the results reported in Table 6.1 of Zhao and Li (2002). The observation which is missed in Zhao and Li (2002) is the standard deviation of the running time. This standard deviation is of the same order of magnitude as the average running time. We can conclude that the computational time depends not only on the size of matrix A but also on the entries,

number of assets	sample size	mean	sd. deviation
5	125	6.8618	5.7377
	250	35.1910	30.2499
	500	134.9312	95.2407
10	125	3.1451	1.6976
	250	30.0261	22.5226
	500	125.1935	104.7991

Table 4: Average running time and its standard deviation for the computation of optimal portfolios by the central path projection algorithm for different number of assets and different sample sizes (all values in sec. of CPU time).

which is particularly surprising since samples are drawn from the same distribution. Of course, the sample of size 125 can be considered as small even for returns of 5 assets. But the sample of size 500 is already quite large. In addition, the averaged running time is increasing with the sample size, however that increase is very irregular. At the same time, the standard deviation behaves also irregularly (similarly as for the averaged running time). All these observations show that the computation time is very sensitive not only to the dimensionality of the problem but also to a particular realization of the data. However we can still conclude that on average the algorithm is efficient for moderate-size portfolio optimization problems. Indeed, Zhao and Li (2002) already observed that the convergence rate of their algorithm was slow. The same is valid for our extension of their algorithm, which makes the algorithm not practically applicable for portfolio optimization problems when the size of the discrete distribution of returns is larger than 10^3. There is no clear indication which part of the algorithm has the main effect on slowing down the convergence.

Summary

We presented two optimization algorithms for financial portfolios. These algorithms find optimal portfolios in problems when the nonlinear optimization in mean-risk setting can be reformulated as a linear programming problem. The algorithm implementing Benders cuts allows for large samples. Our experience with very large data samples obtained by simulation from a given distribution shows that the algorithm is robust and estimated optimal portfolios are very stable (the standard deviation of portfolio weights is below 0.5 %). The second algorithm based on the central path projection can be useful for small data samples where solutions are not unique and form a multidimensional simplex. Extracting from this simplex a point with the smallest distance to a given benchmark portfolio can be used to improve the decision process in asset management. However, the second algorithm which implements the central path projection requires further analysis. In particular, we would like to make the algorithm's performance and running time less dependent on the realization of the data.

Acknowledgments

The author gratefully acknowledges financial support from National Science Centre, Poland, project 2014/13/B/HS4/00176.

Bibliography

J. F. Benders. Partitioning procedures for solving mixed-variables programming problems. *Numerische Mathematik*, 4:238–252, 1962. URL https://doi.org/10.1007/BF01386316. [p312]

A. Ghalanos. *parma: Portfolio Allocation and Risk Management Applications*, 2016. URL https://cran.r-project.org/web/packages/parma/parma.pdf. R package version 1.5-3. [p308]

W. K. Klein Haneveld and M. H. Van der Vlerk. Integrated chance constraints: Reduced forms and an algorithm. *Computational Management Science*, 3:245–269, 2006. URL https://doi.org/10.1007/s10287-005-0007-3. [p312]

R. Koenker and I. Mizera. Convex optimization in R. *Journal of Statistical Software*, 60(5):1–23, 2014. URL https://doi.org/10.18637/jss.v060.i05. [p308]

H. Konno. Piecewise linear risk function and portfolio optimization. *Journal of the Operations Research Society of Japan*, 33:139–156, 1990. URL https://doi.org/10.15807/jorsj.33.139. [p310]

H. Konno and H. Yamazaki. Mean-absolute deviation portfolio optimization model and its application to Tokyo stock market. *Management Science*, 37:519–531, 1991. URL https://doi.org/10.1287/mnsc.37.5.519. [p310]

H. Konno, H. Waki, and A. Yuuki. Portfolio optimization under lower partial risk measures. *Asia-Pacific Financial Markets*, 9:127–140, 2002. URL https://doi.org/10.1023/A:1022238119491. [p310]

A. Künzi-Bay and J. Mayer. Computational aspects of minimizing conditional value at risk. *Computational Management Science*, 3:3–27, 2006. URL https://doi.org/10.1007/s10287-005-0042-0. [p308, 312, 322, 323]

R. Mansini, W. Ogryczak, and M. G. Speranza. Twenty years of linear programming based portfolio optimization. *European Journal of Operational Research*, 234:518–535, 2014. URL https://doi.org/10.1016/j.ejor.2013.08.035. [p308]

B. G. Peterson and P. Carl. *Portfolio Analysis, Including Numerical Methods for Optimization of Portfolios*, 2015. URL https://cran.r-project.org/web/packages/PortfolioAnalytics/PortfolioAnalytics.pdf. R package version 1.0.3636. [p308]

B. Pfaff. *Financial Risk Modelling and Portfolio Optimization with R, 2nd Ed.* John Wiley & Sons, 2016. ISBN 978-1-119-11966-1. [p308]

R. T. Rockafellar and S. Uryasev. Optimization of conditional value-at-risk. *Journal of Risk*, 2:21–42, 2000. URL https://doi.org/10.21314/JOR.2000.038. [p310]

R. T. Rockafellar, S. Uryasev, and M. Zabarankin. Master funds in portfolio analysis with general deviation measures. *Journal of Banking and Finance*, 30:743–778, 2006. URL https://doi.org/10.1016/j.jbankfin.2005.04.004. [p309]

S. Theussl and H. W. Borchers. *CRAN Task View: Optimization and Mathematical Programming*. CRAN, 2016. URL http://cran.r-project.org/web/views/Optimization.html. [p308]

D. Würtz, Y. Chalabi, W. Chen, and A. E. Pfaff. *Portfolio Optimization with R/Rmetrics*. Rmetrics Association & Finance Online, 2009. [p308]

Y.-B. Zhao and D. Li. Locating the least 2-norm solution of linear programs via a path-following method. *SIAM Journal on Optimization*, 12(4):893–912, 2002. URL https://doi.org/10.1137/S1052623401386368. [p309, 315, 317, 325, 326]

Andrzej Palczewski
Faculty of Mathematics, Informatics and Mechanics, University of Warsaw
Banacha 2, 02-097 Warsaw
Poland
A.Palczewski@mimuw.edu.pl

Welfare, Inequality and Poverty Analysis with rtip: An Approach Based on Stochastic Dominance

by Angel Berihuete, Carmen D. Ramos and Miguel A. Sordo

Abstract Disparities in economic welfare, inequality and poverty across and within countries are of great interest to sociologists, economists, researchers, social organizations and political scientists. Information about these topics is commonly based on surveys. We present a package called **rtip** that implements techniques based on stochastic dominance to make unambiguous comparisons, in terms of welfare, poverty and inequality, among income distributions. Besides providing point estimates and confidence intervals for the most commonly used indicators of these characteristics, the package **rtip** estimates the usual Lorenz curve, the generalized Lorenz curve, the TIP (Three I's of Poverty) curve and allows to test statistically whether one curve is dominated by another.

Introduction

Surveys of income provide policy makers, researchers, social organizations and general public with a rich source of data to address and understand the topics of welfare, income inequality and poverty. In one approach, income differences among states, populations or groups are summarised by univariate indices. The most popular aggregate inequality and poverty indices (such as the Gini index and the at-risk-of-poverty rate) summarize inequality or poverty by a univariate index. Such an index is easy to compare over time, across regions or countries but may be insufficient for more detailed research and refined decision making. Increasing availability of data sets from a variety of statistical sources (including national and international statistical offices) allows scientists and policy-makers to carry out detailed comparisons by developing and applying more sophisticated tools and graphical representations from real data sets.

This paper presents an R package, called **rtip**, that implements methods and techniques based on stochastic dominance to make unambiguous comparisons, in terms of welfare, poverty and inequality, of income distributions. Stochastic dominance requires unanimity in rankings for large classes of indices (rather than simpler rankings based on a single index). We regard one income distribution as dominating another only if the same ranking is obtained for them with an entire family of indices. Stochastic dominance rules for income distributions can be easily implemented by seeking a dominance relation between three graphical representations: the Lorenz curve, the generalized Lorenz curve and the TIP (Three I's of Poverty) curve (also called the cumulative poverty gap (CPG) curve or the poverty profile curve, see Barrett et al. (2016) for other names). Two income distributions can be unambiguously ranked in terms of inequality if their Lorenz curves do not intersect (see Atkinson, 1970). Similarly, two income distributions can be unambiguously ranked with respect to a wide class of reasonable social welfare functions (see Shorrocks, 1983, for details) if their respective generalized Lorenz curves do not intersect. The literature also contains results (see Jenkins and Lambert, 1998a,b; Sordo and Ramos, 2011) connecting unambiguous poverty rankings by general classes of poverty measures with non-intersections of TIP curves. The package **rtip** estimates these curves directly from data sets and implements some tests to assess any statistical significance.

We can find some functions, in different software languages, addressing the analysis of income distributions via generalized Lorenz curves and their associated indices of inequality. For example, the Stata commands[1] **glcurve** (Jenkins and Van Kerm, 2004; Van Kerm and Jenkins, 2001), **svylorenz** (Jenkins, 2006), **clorenz** (Araar, 2005), **alorenz** (Azevedo and Franco, 2006) and **lorenz** (Jann, 2016), estimate, draw and provide variance estimates for generalized Lorenz curves. The command **povdeco** (Jenkins, 1999) estimates the poverty indices from the Foster et al. (1984) family. With **rtip**, we want to make these methods available to R users [2]. We can find some packages in the R environment to analyse inequality and poverty of income distributions. For example, various indices of poverty and inequality are included in the packages **IC2** (Plat, 2012) and **ineq** (Zeileis, 2014), which also provide some graphical tools, including the Lorenz curve. A comprehensive collection of indicator methodology is included in the package **laeken** (Alfons and Templ, 2013), which implements functions and generalized Lorenz curves to estimate a wide set of social inclusion and poverty indicators in

[1] user-written functions in Stata are called commands.

[2] This code was developed in the context of a project undertaken by the authors (Sordo et al., 2014) to compare, by using stochastic dominance techniques, inequality and poverty in Andalusia with other European regions. **rtip** has been also applied to the study of inequality by using tax return data (Sordo et al., 2017).

complex surveys. An interesting book combining EU-SILC surveys (the survey on income and living conditions conducted by Eurostat, the statistical office of the EU) and R is Longford (2015).

Some strengths of using the package **rtip** presented in this paper compared with others are the following:

(a) **rtip** provides functions to load microdata having EU-SILC format (see Datasets section).

(b) **rtip** evaluates income distributions on several equivalence scales and with adjustment for differences in household composition. The choice of the equivalence scale is important and can considerably affect the results (see, for example, Buhmann et al. (1988); Jenkins and Cowell (1994); De Vos and Zaidi (1997), who showed that the choice of the equivalence scale is not innocuous). Although the package **rtip** employs by default the OECD-modified scale used by Eurostat, it also allows to use the parametric scale in Buhmann et al. (1988), which covers almost all equivalence scales used in practice.

(c) Besides providing point estimates and confidence intervals for the most commonly used indicators for inequality, welfare and poverty, the package **rtip** implements estimation of the usual Lorenz curve, the generalized Lorenz curve and the TIP curve and provides a way, based on the distribution-free test for generalized Lorenz and deprivation dominance suggested by Xu (1997) and Xu and Osberg (1998), respectively, to test statistically whether a curve is dominated by another.

The remainder of this paper is organized as follows. First, we explain the processes of loading data and setting up the surveys. Then, we briefly describe the indices and curves included in the package, explaining how to estimate them with **rtip**. Description of the statistical procedures implemented to test dominance of these curves is also presented and illustrated with examples. Finally we provide conclusions as well as a section to explain to users how to load their own datasets.

Datasets

The **rtip** package offers the possibility to load data extracted from EU-SILC using the `loadEUSILC` function. Users can modify `loadEUSILC` in order to load their own datasets. For instance, `loadLCS` function is a modification of `loadEUSILC` to load data from the Spanish Living Conditions Survey (see the Appendix). Note that **rtip** is not restricted to datasets with EU-SILC format and users can load their own datasets.

The **rtip** package contains two datasets: `eusilc2` and `LCS2014`.

```
library(rtip)
data(eusilc2)
data(LCS2014)
```

The `eusilc2` dataset is a modification of the synthetic `eusilc` dataset in the package **laeken** (see Alfons and Templ, 2013). The aim of this modification[3] is to set up the variables according to expectations of the **rtip** package (see Table 1). The `eusilc` dataset is related to Austrian EU-SILC data from 2006, and was generated to comply with EU-SILC confidentiality rules. The `LCS2014` dataset[4] is a data frame containing a selection of variables from the living conditions survey released in 2014 by the Spanish National Statistics Institute (INE in Spanish).

Both datasets, `eusilc2` and `LCS2014`, have seven variables which are briefly described in Table 1 (for further information see Eurostat, 2007). The records in these datasets are households, not individuals. Thus the 14,827 individual-level records in the `eusilc` dataset are condensed to 6,000 household-level records in the `eusilc2` dataset.

Setting up the surveys

The income variable most commonly studied for the inequality and poverty assessment is the disposable household income. Most official statistics (and **rtip**) use the equivalised disposable household income, which is the total disposable income of the household adjusted by taking into account its

[3]The code used to modify the `eusilc` dataset can be found at https://github.com/AngelBerihuete/rtip/blob/master/data-raw/eusilc_eusilc2.R.

[4]The dataset can be obtained at http://www.ine.es/dyngs/INEbase/en/operacion.htm?c=Estadistica_C&cid=1254736176807&menu=ultiDatos&idp=1254735976608.

Name of the variable	Meaning
DB010	Year of the survey
DB020	Country
DB040	Region
DB090	Household cross-sectional weight
HX040	Household size
HX050	Equivalised household size
HX090	Equivalised disposable income

Table 1: Variables needed before setting up the dataset. Variables HX050 and HX090, provided by Eurostat, are calculated using the OECD-modified scale.

composition (number of adults and children). The adjustment is made by dividing the disposable household income by the equivalised household size, which is defined as the number of household members converted into equivalent adults by using a specific equivalence scale. The equivalisation factor employed by Eurostat is the OECD-modified scale, which gives a weight of 1.0 to the first person aged 14 or more, a weight of 0.5 to other persons aged 14 or more and a weight of 0.3 to persons aged 0-13. **rtip** uses this equivalisation factor by default, but can also use the parametric scale of Buhmann et al. (1988). In this case, the equivalised household size is given by n^s, where n is the number of members of the household and s is a parameter known as elasticity of equivalence, with $0 \leq s \leq 1$. The value $s = 0.5$ is frequently used to make comparisons between countries (when $s = 0$ the composition is irrelevant).

The information is held in two files: basic household register (H-file) and household data (D-file). These files are loaded by functions `loadEUSILC` or `loadLCS` from EU-SILC or INE surveys, respectively. Next, the data is set up by using the function `setupDataset` which has one mandatory argument (`dataset`) and five optional arguments: the country to be analysed (`country`), the region of the country (`region`), the equivalence scale (`s`), a deflator (`deflator`) and the purchasing power parity rate (`pppr`). All optional arguments have `NULL` default value, except for `country = 'ES'`.

The `country` and the `region` are expressed using the nomenclature of territorial units for statistics[5]. All regions of the country will be selected by default (`region = NULL`) but it is possible to select one or more regions using a character string. Income is expressed in current monetary units. If `deflator` is not `NULL` the value assigned will be used as a deflator. Finally, by setting up the ratio of the purchasing power parity conversion factor to market exchange rate (`pppr`) we can compare the income across countries.

```
dataset <- setupDataset(eusilc2, country = "AT", region = NULL,
        s = NULL, deflator = NULL, pppr = NULL)
```

Next, we form a data frame with a new variable `ipuc`. The `ipuc` variable is the income per unit of consumption[6] (or equivalised disposable income) which also takes into account the deflator and the purchasing power parity rate. If `deflator = NULL`, `pppr = NULL` and `s = NULL`, `ipuc` is set to `HX090` (see Table 1).

Indicators and curves

In this section we briefly describe some indicators and curves that are widely used in the study of poverty, inequality and welfare. The `eusilc2` dataset contains all the data necessary for estimating them.

Let X be a random variable[7] with cumulative distribution function F, and let $\zeta_p = F^{-1}(p)$ be the quantile function with $0 \leq p \leq 1$. Let n be the number of observations in the sample, let $x := \{x_i\}_{i=1}^n$ be individual incomes sorted into ascending order so that $x_1 \leq x_2 \leq \cdots \leq x_n$, and let $\omega := \{\omega_i\}_{i=1}^n$ denote the corresponding sample weights[8]. Weighted quantiles for the estimation of the population values are given by $\hat{\zeta}_p(x, \omega) = x_{(r)}$, where $x_{(r)}$ is the r-th order statistic such that $r = [p \sum_{i=1}^n \omega_i]$ is the closest integer not greater than $p \sum_{i=1}^n \omega_i$. $\hat{\zeta}_p(x, \omega)$ is the sample quantile level such that $100p$ percent

[5]Nomenclature of territorial units can be found at http://ec.europa.eu/eurostat/web/nuts/overview.

[6]The number of units of consumption is the number of household members converted into equivalent adults by using a specific equivalence scale.

[7]Unless otherwise stated it refers to equivalised disposable household income.

[8]Following indications in Alfons and Templ (2013), we take into account sample weights in the estimation of indices and curves.

of the weighted sample of observations is less than or equal to $\hat{\zeta}_p(x, \omega)$ and $100(1-p)$ percent of the weighted observations is greater (Beach and Kaliski, 1986).

Indicators of poverty, inequality and welfare

Poverty curves and indicators are based on the poverty threshold that distinguishes between poor and non-poor households. The median income, $\zeta_{0.5}$, is frequently used for this purpose. On the recommendation of Eurostat this threshold, called the *at-risk-of-poverty threshold*, is set at 60% of the national median equivalised disposable income ($arpt = 0.6 \cdot \zeta_{0.5}$) and is estimated by

$$\widehat{arpt} = 0.6 \cdot \hat{\zeta}_{0.5}(x, \omega), \tag{1}$$

using the function `arpt()`. The default setting of 60% can be easily changed via the argument `pz`. Optional variable names can be set up for the income per unit of consumption (`ipuc`), the household cross-sectional weight (`hhcsw`) and the household size (`hhsize`). Default values correspond to EU-SILC names. For the sake of clarity we use the default variable names in the following examples. A sample call to `arpt` would look like:

```
arpt(dataset, ipuc = "ipuc", hhcsw = "DB090", hhsize = "HX040", pz = 0.6)
```

The indicator called *at-risk-of-poverty rate*, defined as the proportion of persons with an equivalised disposable income below the *at-risk-of-poverty threshold*, is estimated using the function `arpr()` (see Table 2). A sample call to `arpr` would look like:

```
arpr(dataset, arpt.value = arpt(dataset, pz = 0.6))
```

Using the function `arpr()` and changing the argument `pz` to 40%, 50% and 70% in `arpt()`, we obtain the *dispersion around the at-risk-of-poverty threshold* (percentage of persons with an equivalised disposable income below 40%, 50% and 70% of the national median equivalised disposable income, respectively). For example:

```
arpr(dataset, arpt.value = arpt(dataset, pz = 0.4))
```

Table 2 contains a brief description of some other well-known poverty indicators such as the Foster, Greer and Thorbecke (FGT_1) poverty index (Foster et al., 1984). This index is calculated by the function `s1()`. Its normalized version, also called the poverty gap ratio, is obtained by setting `norm = TRUE`. In this case, the index provides the average of the ratio of the poverty gaps[9] to the at-risk-of-poverty threshold.

```
s1(dataset, arpt.value = arpt(dataset), norm = TRUE)
```

The **rtip** package also provides the function `s2()` to calculate the Sen-Shorrocks-Thon (SST) poverty index (Shorrocks, 1995; Zheng, 1997). It is estimated as twice the area below the TIP curve (see TIP curve in next section).

For income inequality, we calculate the most commonly used indices of inequality, the *Gini index*, and the *quintile share ratio* (see Table 2). The *mean income per person*, *per household* and *per unit of consumption* are estimated by the respective functions `mip()`, `mih()` and `miuc()`.

Confidence intervals can be obtained for all the indicators in **rtip** package by a bootstrap method. We use the **boot** package (Canty and Ripley, 2016) to generate bootstrap replicates. The user can change the number of replicates with the parameter `rep` which, by default, is set to 500. If `verbose = TRUE` we obtain a plot showing the histogram of the indicator estimations (replicates) and the Standard Normal quantile-quantile plot of bootstrap estimates. For instance, in case of the `arpt` function and 98% confidence interval, a typical call would look like:

```
arpt(dataset, pz = 0.6, ci = 0.98, rep = 500, verbose = TRUE)
```

Curves for inequality, welfare and poverty

Lorenz and generalized Lorenz curve

The Lorenz curve (Gastwirth, 1971) is defined as

[9] A poverty gap is the difference between the *at-risk-of-poverty* threshold and the equivalised disposable income, with the non-poor being given a difference of zero.

Index name, function	Description	Formula
At-risk-of-poverty rate, `arpr()`	Proportion of persons with an equivalised disposable income below the *at-risk-of-poverty threshold*	$\frac{\sum_{i=1}^{n} \omega_i \mathbf{I}\left(x_i \leq \widehat{arpt}\right)}{\sum_{i=1}^{n} \omega_i} \cdot 100$ (2)
Relative median at-risk of-poverty gap, `rmpg()`	Difference between the *at-risk-of-poverty threshold* and the median equivalised disposable income of people below the *at-risk-of-poverty threshold*, expressed as a percentage of the threshold	$\frac{\widehat{arpt} - \hat{\zeta}^{p}_{0.5}(x,\omega)}{\widehat{arpt}} \cdot 100$ (3)
Foster, Greer and Thorbecke, `s1()`	Average of the absolute poverty gaps (difference between the *at-risk-of-poverty* threshold and the equivalised disposable income, with the non-poor being given a difference of zero)	$\frac{\sum_{i=1}^{n} \omega_i (\widehat{arpt} - x_i)^{+}}{\sum_{i=1}^{n} \omega_i}$ (4)
Gini, `gini()`	Relationship of cumulative proportions of the population arranged according to the level of equivalised disposable income, to the cumulative proportions of the equivalised total disposable income they receive	$\left[\frac{2\sum_{i=1}^{n}\left(\omega_i x_i \sum_{j=1}^{i} \omega_j\right) - \sum_{i=1}^{n} \omega_i^2 x_i}{\sum_{i=1}^{n} \omega_i \sum_{i=1}^{n} \omega_i x_i} - 1\right] \cdot 100$
Quintile share ratio, `qsr()`	The ratio of total income received by the 20 percent of the population with the highest income to that received by the 20 percent of the population with the lowest income	$\frac{\sum_{i=1}^{n} \omega_i x_i \mathbf{I}\left(x_i > \hat{\zeta}_{0.8}(x,\omega)\right)}{\sum_{i=1}^{n} \omega_i x_i \mathbf{I}\left(x_i \leq \hat{\zeta}_{0.2}(x,\omega)\right)} \cdot 100$ (5)

Table 2: Common indicators of poverty and inequality coded in **rtip** package. In formula 4 we define $(x)^{+} = x$ if $x \geq 0$ and $(x)^{+} = 0$ if $x < 0$. In formulae 2 and 5, $\mathbf{I}(\cdot)$ is the usual indicator function that equals 1 if the bracketed expression is true, and 0 otherwise. In formula 3, $\hat{\zeta}^{p}_{0.5}(x,\omega)$ is the median income of poor persons.

ISSN 2073-4859

$$L(p) = \frac{1}{\mu}\int_0^p \zeta_q dq,$$

and the generalized Lorenz (GL) curve is given by $GL(p) = \mu L(p)$ (Shorrocks, 1983) where μ denotes the mean income. The values of the GL curve are estimated on a regular grid of K points selected such that $p_i = i/K$, and their population quantiles denoted by $\zeta_{p_i} = F^{-1}(p_i)$ with $i = 1, 2, ..., K$. The conditional mean of income less than or equal to ζ_{p_i} is denoted as $\gamma_i = \mathbb{E}\left[X | X \leq \zeta_{p_i}\right]$, for $i = 1, 2, ..., K$ ($\gamma_K = \mu$). The $K \times 1$ vector of GL ordinates at $p_1, p_2, \ldots, p_K$ is given by $\theta = [p_1\gamma_1, p_2\gamma_2, \ldots, p_K\gamma_K]'$ and can be estimated consistently by

$$\hat{\theta} = [p_1\hat{\gamma}_1, p_2\hat{\gamma}_2, \ldots, p_K\hat{\gamma}_K]', \tag{6}$$

where the sample counterpart of γ_i is

$$\hat{\gamma}_i = \frac{\sum_{j=1}^{r_i} \omega_j x_j}{\sum_{j=1}^{r_i} \omega_j},$$

and $r_i = [p_i \sum_{j=1}^n \omega_j]$ (the closest integer not greater than $p_i \sum_{j=1}^n \omega_j$) is an integer such that $\hat{\zeta}_{p_i}(x, \omega) = x_{(r_i)}$ is the r_i-th sample quantile level such that $100p_i$ percent of the weighted sample of observations is less than or equal to $\hat{\zeta}_{p_i}(x, \omega)$, $i = 1, 2, ..., K$ (Beach and Davidson, 1983; Beach and Kaliski, 1986). In the package **rtip**, the function `lc()` is implemented to estimate both Lorenz and GL curve ordinates. This function calculates the Lorenz curve for the number of abscissae p_i given by the argument `samplesize`. Following the examples in Beach and Davidson (1983), Beach and Kaliski (1986) and Xu (1997), we set `samplesize = 10` by default. If `samplesize = complete`, ordinates are computed in each value along the whole distribution. By setting `generalized = TRUE`, the GL curve is calculated. The left-hand panel of Figure 1 shows the Lorenz curve for income distribution of the Burgenland region. It is produced with:

```
Burgenland <- setupDataset(eusilc2, country = "AT", region = "Burgenland")
lorenz_curve <- lc(Burgenland, samplesize = 10, generalized = FALSE, plot = FALSE)

p1 <- ggplot(lorenz_curve, aes(x.lg, y.lg)) + geom_line() +
        geom_segment(aes(x = 0, y = 0, xend = 1, yend = 1),
             linetype = "dotted", color = "grey") +
        scale_x_continuous(expression(p)) +
        scale_y_continuous(expression(L(p))) + theme_bw()
print(p1)
```

The variable `lorenz_curve` contains the abscissae p_i and Lorenz curve ordinates. A less elaborate plot is obtained by setting `plot=TRUE` in `lc` function.

TIP curve

For an individual (or household-level) measure of deprivation Y, with distribution function F_Y, the deprivation profile for F_Y is (Shorrocks, 1995, 1998)

$$D(F_Y, p) = \int_{F_Y^{-1}(1-p)}^{\infty} y dF_Y(y) = \int_{1-p}^{1} F_Y^{-1}(q) dq, \;\; p \in [0, 1].$$

Let $z > 0$ be a poverty threshold and let X be an income random variable. If we consider the poverty gap $Y = (z - X)^+$ as the measure of deprivation, where $x^+ = max\{x, 0\}$, the TIP (Three I's of Poverty) curve is obtained and denoted $TIP(p, z)$. Alternatively, we can write (Jenkins and Lambert, 1998a,b)

$$TIP(p, z) = \int_0^{r_z^X} \left(z - F^{-1}(t)\right) dt, \;\; p \in [0, 1],$$

where $r_z^X = \sup\{F(x) : x < z\}$ is the proportion of people with income below z. Scaling by z the poverty gap, that is, using $Y = (z - X)^+/z$ as the measure of deprivation, we obtain the normalized TIP curve, which is simply $TIP(p, z)/z$.

For its estimation, let the observations of poverty gaps $y := \{y_i\}_{i=1}^n$ be ordered in increasing order so that $y_1 \leq y_2 \leq \cdots \leq y_n$ with $\hat{z} = \widehat{arpt}$ given by equation (1) and let $\omega := \{\omega_i\}_{i=1}^n$ denote the

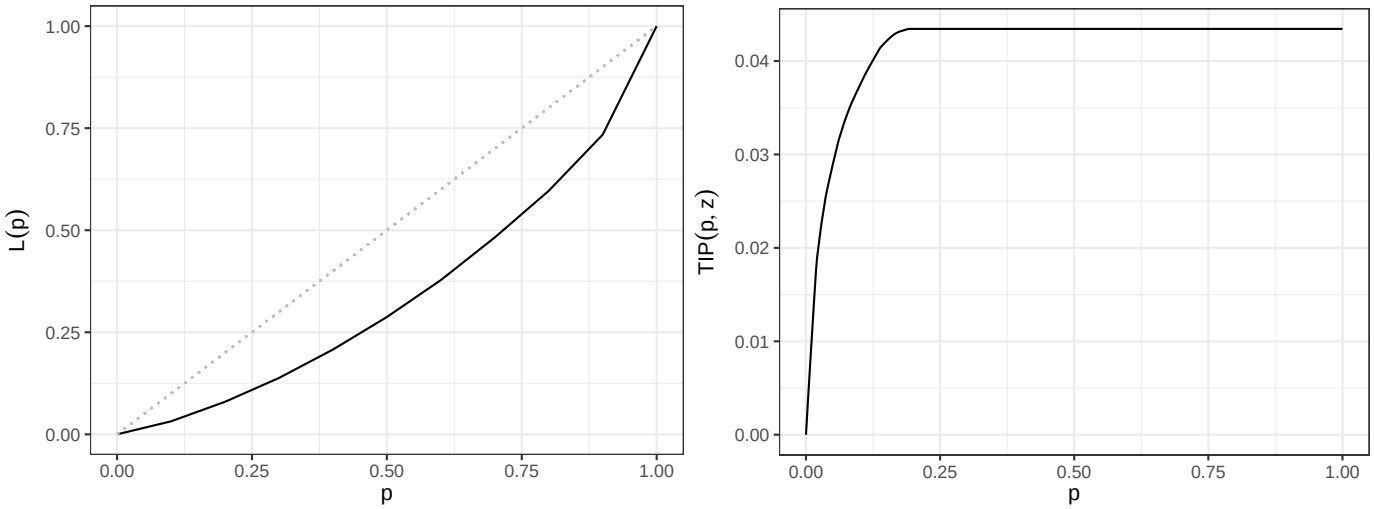


Figure 1: Lorenz curve for Burgenland region (left). Dotted diagonal represents the benchmark for a perfectly equal income distribution. Normalized TIP curve for Burgenland region (right).

corresponding sampling weights. Using the relation between deprivation profile and the GL curve, $D(F_Y, p) = \mu(F_Y) - GL(F_Y, 1-p)$ for $p \in [0,1]$, the $K \times 1$ vector of TIP curve ordinates corresponding to $[p_1, p_2, \ldots, p_K]'$ is estimated consistently by

$$\hat{\phi} = [(\hat{\gamma}_K - p_{K-1}\hat{\gamma}_{K-1}), (\hat{\gamma}_K - p_{K-2}\hat{\gamma}_{K-2}), \ldots, (\hat{\gamma}_K - p_1\hat{\gamma}_1), \hat{\gamma}_K]', \tag{7}$$

where the sample counterpart of γ_i is

$$\hat{\gamma}_i = \frac{\sum_{j=1}^{r_i} \omega_j y_j}{\sum_{j=1}^{r_i} \omega_j},$$

and $r_i = [p_i \sum_{j=1}^{n} \omega_j]$ (the closest integer not greater than $p_i \sum_{j=1}^{n} \omega_j$) is an integer for which $\hat{\zeta}_{p_i}(y, \omega) = y_{(r_i)}$ is the r_i-th sample quantile level such that $100p_i$ percent of the weighted sample of poverty gaps is less than or equal to $\hat{\zeta}_{p_i}(y, \omega)$, $i = 1, 2, ..., K$ (Beach and Kaliski, 1986; Xu and Osberg, 1998). The function `tip()` estimates both the unnormalized and normalized TIP curve ordinates. Normalization is established by setting `norm = TRUE` and the estimated poverty threshold, $\hat{z}$, is computed by the function `arpt`. The number of ordinates computed is given by the the argument `samplesize`, and following the example in Xu and Osberg (1998), we set `samplesize = 50` by default. If `samplesize = complete`, tip ordinates are computed in each value along the whole distribution. The right-hand panel of Figure 1 shows the normalized TIP curve for income distribution of Burgenland region. It is produced with:

```
tip_curve <- tip(Burgenland, arpt(Burgenland), samplesize = 50, norm = TRUE)
p2 <- ggplot(tip_curve, aes(x.tip, y.tip)) + geom_line() +
        scale_x_continuous(expression(p)) +
        scale_y_continuous(expression(TIP(p, z))) +
        theme_bw()
print(p2)
```

As in the previous example, a less elaborate plot is obtained by setting `plot=TRUE` in the call to the `tip` function.

Dominance tests

Given two income distributions X_1 and X_2, X_1 is said to Lorenz dominate X_2 if the Lorenz curve of X_1 lies everywhere above that of X_2, which is interpreted as less inequality in X_1 than in X_2. The normative aspects of Lorenz dominance have been studied by Atkinson (1970) and its relationship to other dominance criteria may be found in Arnold (1987). Similarly, X_1 is said to dominate X_2 in the generalized Lorenz sense if the generalized Lorenz curve of X_1 lies everywhere above that of X_2, which is interpreted in terms of welfare (see Shorrocks, 1983). For the relationship of generalized Lorenz dominance to other dominance criteria, see Ramos et al. (2000). Given a poverty threshold $z > 0$, X_1

is said to TIP dominate X_2 if $TIP_{X_1}(p,z) \geq TIP_{X_2}(p,z)$ for all $p \in (0,1)$, which means that there is less poverty in X_2 than in X_1 according to various wide classes of poverty indices, see Jenkins and Lambert (1998a,b) and Sordo et al. (2007). If we use different poverty thresholds for different income distributions, that is, z_1 in X_1 and z_2 in X_2, non-intersection of normalized TIP curves is equivalent to unanimous poverty orderings by different classes of poverty indices based on normalized poverty gaps.

Since the initial papers by Beach and Davidson (1983) and Bishop et al. (1989) focussing on statistical tests for Lorenz dominance and generalized Lorenz dominance, respectively, many studies have been conducted to implement these ranking criteria empirically (see Chapter 17 in Duclos and Araar (2006) for a review). Some tests in the literature are based on a two-stage testing strategy including multiple pairwise sub-tests. Xu (1997) offers an alternative to this approach by providing a joint and simpler procedure to test for generalized Lorenz dominance directly, which is adapted to testing for deprivation dominance in Xu and Osberg (1998). We have implemented Xu (1997) and Xu and Osberg (1998) procedures in **rtip**.

Generalized Lorenz dominance

To make statistical inference about GL dominance from sample GL curve estimates we have implemented the asymptotically distribution-free statistical inference procedure in Xu (1997). This article provides one test based on Theorem 1 in Beach and Davidson (1983) which derives the asymptotic joint variance-covariance matrix of GL curve ordinates. EU-SILC surveys involve sampling weights. We implemented an extension of the methodology in Beach and Davidson (1983) to samples which involve weighted observations (see Beach and Kaliski, 1986).

Given two income distributions, X_1 and X_2, let θ_1 and θ_2 be the $K \times 1$ vectors of GL curve ordinates for X_1 and X_2, respectively. The dominance relation tested by the null hypothesis is $H_0 : \theta_1 - \theta_2 \geq 0$ against the alternative hypothesis $H_1 : \theta_1 - \theta_2 \ngeq 0$. The test statistic, T, for the GL dominance is (see Xu, 1997)

$$T = \tilde{\Delta}' \left[\frac{\hat{\Sigma}_1}{n_1} + \frac{\hat{\Sigma}_2}{n_2} \right]^{-1} \tilde{\Delta}, \tag{8}$$

where $\tilde{\Delta} = [(\hat{\theta}_1 - \hat{\theta}_2) - (\tilde{\theta}_1 - \tilde{\theta}_2)]$; n_i is the size of a random sample from X_i; $\hat{\Sigma}_i$ is the estimated $K \times K$ covariance matrix for the unrestricted vector of GL ordinates $\hat{\theta}_i$ given by (6) while $\tilde{\theta}_i$ is the restricted estimate minimizing ($i = 1, 2$)

$$\Delta' \left[\frac{\hat{\Sigma}_1}{n_1} + \frac{\hat{\Sigma}_2}{n_2} \right]^{-1} \Delta \tag{9}$$

$$\text{s.t.} \; (\theta_1 - \theta_2) \geq 0$$

with $\Delta = [(\hat{\theta}_1 - \hat{\theta}_2) - (\theta_1 - \theta_2)]$. The auxiliary function `OmegaGL()` computes the empirical unrestricted vector of GL curve ordinates, $\hat{\theta}_i$, and its corresponding covariance matrix, $\hat{\Sigma}_i$, $i = 1, 2$. The function for testing generalized Lorenz dominance between two income distributions is `testGL()`. For both functions the number of ordinates, K, estimated by **rtip** and employed for testing dominance is controlled by the argument `samplesize`. Following the example in Xu (1997) the default value is `samplesize=10`. The upper-and lower-bounds of critical values (at the α significance level) for testing inequality restrictions are provided by Kodde and Palm (1986). If the value of the T statistic (called `Tvalue`) falls into an inconclusive region (between the lower- and upper-bounds) the simulated p-value is estimated following Wolak (1989), otherwise the p-value is set to `NA`. For instance, to test the null hypothesis that the income distribution of Burgenland dominates the income distribution of Carinthia in the generalized Lorenz sense we use the following procedure:

```
Burgenland <- setupDataset(eusilc2, country = "AT", region = "Burgenland")
Carinthia <- setupDataset(eusilc2, country = "AT", region = "Carinthia")
testGL(Burgenland, Carinthia, generalized = TRUE, samplesize = 10, alpha = 0.05)
```

The output produced is:

```
$Tvalue
         [,1]
[1,] 7723.701

$p.value
[1] NA
```

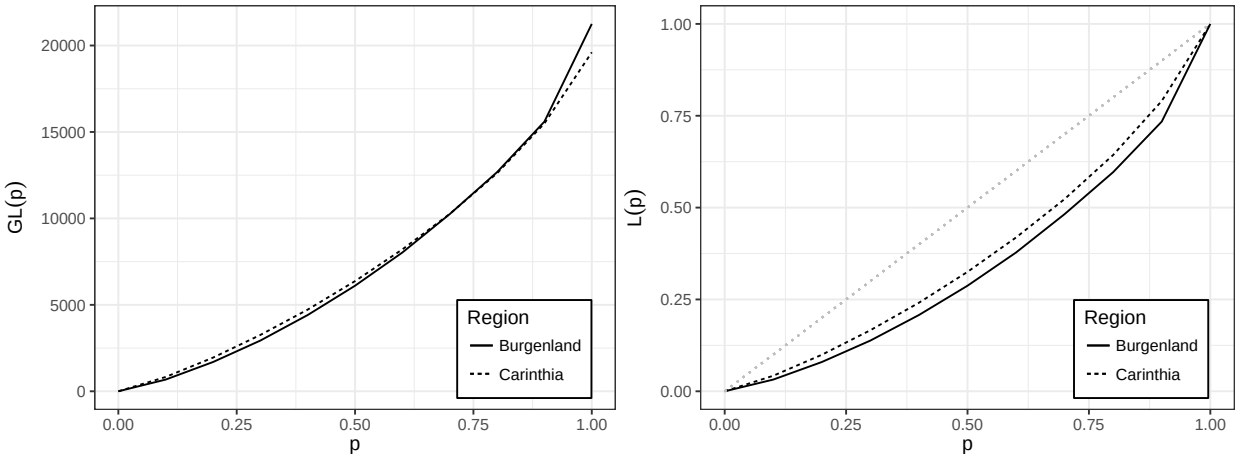


Figure 2: Generalized Lorenz curves for Burgenland and Carinthia (left). Lorenz curves for Burgenland and Carinthia Dotted diagonal represents the benchmark for a perfect equality (right).

```
$decision
[1] "Reject null hypothesis"
```

In this example, the null hypothesis of dominance is rejected with a significance level of 5%. The left-hand panel of Figure 2 displays the two estimated GL curves.

To test the null hypothesis that the income distribution of Carinthia dominates the income distribution of Burgenland in the Lorenz sense we use the function `testGL()` as follows:

```
testGL(Carinthia, Burgenland, generalized = FALSE)
```

This results in:

```
$Tvalue
             [,1]
[1,]  4.049037e-12

$p.value
[1] NA

$decision
[1] "Do not reject null hypothesis"
```

In this case we do not have evidence to reject the null hypothesis with a significance level of 5%. The right-hand panel of Figure 2, which displays the two estimated Lorenz curves, suggests (without statistical significance) that Carinthia may dominate Burgenland in the Lorenz sense.

TIP dominance

To make statistical inference about TIP dominance we have implemented the asymptotically distribution-free statistical procedure in Xu and Osberg (1998). As in the case of GL dominance, we have followed the methodology suggested by Beach and Davidson (1983) and Beach and Kaliski (1986).

Let's consider two poverty thresholds $z_1, z_2 > 0$; two income distributions, X_1 and X_2, and the corresponding poverty gaps $Y_i = (z_i - X_i)^+$, $i = 1, 2$. Let ϕ_1 and ϕ_2 be the $K \times 1$ vectors of TIP curve ordinates for X_1 and X_2, respectively. The dominance relation tested on the null hypothesis is $H_0 : \phi_1 - \phi_2 \geq 0$ against the alternative hypothesis $H_1 : \phi_1 - \phi_2 \ngeq 0$. Following the methodology in Xu and Osberg (1998) to test for TIP dominance between two TIP curves, the test-statistic implemented is such that

$$T = \tilde{\Delta}' \left[\frac{\hat{\Omega}_1}{n_1} + \frac{\hat{\Omega}_2}{n_2} \right]^{-1} \tilde{\Delta}, \tag{10}$$

where $\tilde{\Delta} = [(\hat{\phi}_1 - \hat{\phi}_2) - (\tilde{\phi}_1 - \tilde{\phi}_2)]$; n_i is, for i=1 ,2, the size of a random sample from X_i; $\hat{\Omega}_i = R\hat{\Sigma}_i R'$, $\hat{\Sigma}_i$ is the estimated $K \times K$ covariance matrix for the unrestricted vector of GL ordinates and R is the

$K \times K$ differencing matrix[10] according to

$$\begin{bmatrix} 0 & \cdots & 0 & -1 & 1 \\ 0 & \cdots & -1 & 0 & 1 \\ 0 & \cdots & 0 & 0 & 1 \\ \vdots & \ddots & \vdots & \vdots & \vdots \\ -1 & \cdots & 0 & 0 & 1 \\ 0 & \cdots & 0 & 0 & 1 \end{bmatrix}$$

The matrix $\hat{\Omega}_i$ is, for $i = 1, 2$, the estimated $K \times K$ covariance matrix for the unrestricted vector of TIP ordinates $\hat{\phi}_i$ given by (7) while $\tilde{\phi}_i$ is the restricted estimate minimizing

$$\Delta' \left[\frac{\hat{\Omega}_1}{n_1} + \frac{\hat{\Omega}_2}{n_2} \right]^{-1} \Delta \tag{11}$$

$$\text{s.t.} \ (\phi_1 - \phi_2) \geq 0$$

with $\Delta = [(\hat{\phi}_1 - \hat{\phi}_2) - (\phi_1 - \phi_2)]$. Since the TIP curve becomes horizontal at p equals to the at-risk-of-poverty rate (*arpr*), the test has only been implemented over the interval $[0, max\{arpr_1, arpr_2\}]$ where $arpr_i$ is the at-risk-of-poverty rate for X_i, $i = 1, 2$. Therefore, $\hat{\phi}_i$ are truncated with the same dimension $k = max\{arpr_1, arpr_2\} < K$ and the dimension of $\hat{\Omega}_i$ is $k \times k$, for $i = 1, 2$. The auxiliary function `OmegaTIP()` computes the empirical unrestricted vector of TIP curve ordinates, $\hat{\phi}_i$, and its corresponding covariance matrix, $\hat{\Omega}_i$, $i = 1, 2$. The function for testing TIP dominance between two income distributions is `testTIP()`. Following the practical example in Xu and Osberg (1998) the default value is `samplesize=50`. The rules of rejection and non-rejection based on the value of the statistic T (called `Tvalue`) are in Xu and Osberg (1998). By setting the argument `norm` equal to `TRUE`, the function `testTIP()` uses normalized TIP curves. For example, to test the null hypothesis that the normalized TIP curve of Carinthia dominates the normalized TIP curve of Burgenland we use the following procedure:

```
testTIP(Carinthia, Burgenland, norm = TRUE, samplesize = 50, alpha = 0.05)
```

This yields:

```
$Tvalue
         [,1]
[1,] 11939.99

$p.value
[1] NA

$decision
[1] "Reject null hypothesis"
```

The null hypothesis of dominance is rejected with a significance level of 5%. Figure 3 displays the two estimated normalized TIP curves.

Conclusions

The package **rtip** presented in this paper compares income distributions in terms of welfare, poverty and inequality. Besides providing point estimates and confidence intervals for some commonly used indicators, **rtip** implements the methodology and techniques based on the stochastic dominance to compare income distributions. In particular, we can estimate with **rtip** the usual Lorenz curve, the generalized Lorenz curve, the TIP curve of income distributions and test statistically whether one curve is dominated by another. Although potential users of **rtip** may have their own data, the package allows to load microdata from EU-SILC survey and Spanish Living Conditions survey and offers different equivalence scales to adjust measures of income for differences in household composition. A development version of the **rtip** package can be found at GitHub[11].

[10] Recall that the deprivation profile $D(F_Y, \cdot)$ for F_Y is related to the GL curve $GL(F_Y, \cdot)$ of the deprivation measure for F_Y as follows: $D(F_Y, p) = \mu(F_Y) - GL(F_Y, 1 - p)$, for $p \in [0, 1]$.

[11] https://github.com/AngelBerihuete/rtip.

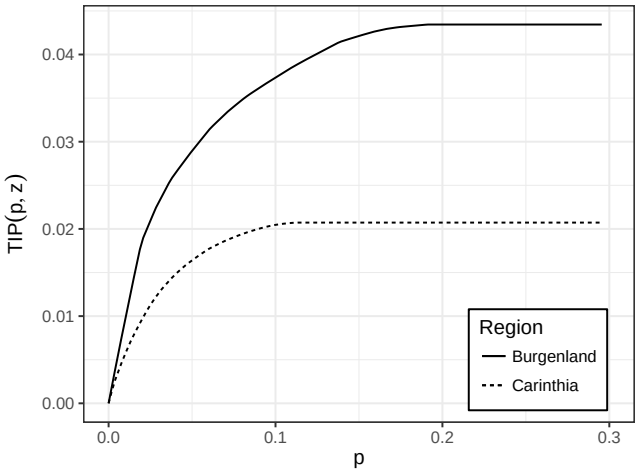


Figure 3: Normalized TIP curves for Burgenland and Carinthia regions (the normalization factors are the respective poverty thresholds).

Acknowledgements

The authors thank two anonymous referees and the Editor of the journal for their detailed comments which have led to significant improvements in both the content and presentation of the paper.
Miguel A. Sordo and Carmen D. Ramos acknowledge the support received from Ministerio de Economía y Competitividad (Spain) under grant MTM2014-57559-P.

Bibliography

A. Alfons and M. Templ. Estimation of Social Exclusion Indicators from Complex Surveys: The R Package **laeken**. *Journal of Statistical Software*, 54(15):1–25, 2013. URL http://doi.org/10.2139/ssrn.2244876. [p328, 329, 330]

A. Araar. *Clorenz: Stata Module to Estimate Lorenz and Concentration Curves*, 2005. URL https://ideas.repec.org/c/boc/bocode/s456515.html. [p328]

B. Arnold. *Majorization and the Lorenz Order: A Brief Introduction*. Springer-Verlag, 1987. [p334]

A. Atkinson. On the Measurement of Inequality. *Journal of Economic Theory*, 2(3):244–263, 1970. URL https://doi.org/10.1016/0022-0531(70)90039-6. [p328, 334]

J. P. Azevedo and S. Franco. **alorenz**: *Stata Module to Produce Pen's Parade, Lorenz and Generalised Lorenz Curve*, 2006. URL https://ideas.repec.org/c/boc/bocode/s456749.html. [p328]

G. F. Barrett, S. G. Donald, and Y.-C. Hsu. Consistent Tests for Poverty Dominance Relations. *Journal of Econometrics*, 191(2):360–373, 2016. [p328]

C. M. Beach and R. Davidson. Distribution-Free Statistical Inference with Lorenz Curves and Income Shares. *The Review of Economic Studies*, 50(4):723–735, 1983. URL https://doi.org/10.2307/2297772. [p333, 335, 336]

C. M. Beach and S. F. Kaliski. Lorenz Curve Inference with Sample Weights: An Application to the Distribution of Unemployment Experience. *Journal of the Royal Statistical Society. Series C (Applied Statistics)*, 35(1):38–45, 1986. [p331, 333, 334, 335, 336]

J. A. Bishop, S. Chakraborty, and P. D. Thistle. Asymptotically Distribution-Free Statistical Inference for Generalized Lorenz Curves. *The Review of Economics and Statistics*, 71(4):725–727, 1989. URL https://www.jstor.org/stable/1928121. [p335]

B. Buhmann, L. Rainwater, G. Schmaus, and T. M. Smeeding. Equivalence Scales, Well-Being, Inequality and Poverty: Sensitivity Estimates across Ten Countries Using the Luxembourg Income Study (LIS) Database. *Review of Income and Wealth*, 34(2):115–142, 1988. URL https://doi.org/10.1111/j.1475-4991.1988.tb00564.x. [p329, 330]

A. Canty and B. D. Ripley. **boot**: *Bootstrap R (S-Plus) Functions*, 2016. R package version 1.3-18. [p331]

K. De Vos and M. A. Zaidi. Equivalence Scale Sensitivity of Poverty Statistics for the Member States of the European Community. *Review of Income and Wealth*, 43(3):319–33, 1997. URL https://doi.org/10.1111/j.1475-4991.1997.tb00222.x. [p329]

J. Y. Duclos and A. Araar. *Poverty and Equity: Measurement, Policy and Estimation with DAD*. Economic Inequality and Poverty Series. Springer-Verlag, 2006. ISBN 9780387258935. URL https://books.google.es/books?id=KOwnYw4qvW4C. [p335]

Eurostat. *Description of SILC User Database Variables: Cross-Sectional and Longitudinal*. Unit F-3: Living conditions and social protection statistics, Directorate F: Social Statistics and Information Society, Eurostat, Luxembourg, 2007. [p329]

J. Foster, J. Greer, and E. Thorbecke. A Class of Decomposable Poverty Measures. *Econometrica*, 52(3): 761–766, 1984. [p328, 331]

J. L. Gastwirth. A General Definition of the Lorenz Curve. *Econometrica*, 39(6):1037–1039, 1971. URL https://EconPapers.repec.org/RePEc:ecm:emetrp:v:39:y:1971:i:6:p:1037-39. [p331]

B. Jann. **lorenz***: Stata Module to Estimate and Display Lorenz Curves and Concentration Curves*, 2016. URL https://ideas.repec.org/c/boc/bocode/s458133.html. [p328]

S. P. Jenkins. **povdeco***: Stata Module to Calculate Poverty Indices with Decomposition by Subgroup*, 1999. URL https://ideas.repec.org/c/boc/bocode/s366004.html. [p328]

S. P. Jenkins. **svylorenz***: Stata Module to Derive Distribution-Free Variance Estimates from Complex Survey Data*, 2006. URL https://ideas.repec.org/c/boc/bocode/s456602.html. [p328]

S. P. Jenkins and F. A. Cowell. Parametric Equivalence Scales and Scale Relativities. *The Economic Journal*, pages 891–900, 1994. URL https://doi.org/10.2307/2234983. [p329]

S. P. Jenkins and P. J. Lambert. Ranking Poverty Gap Distributions: Further TIPs for Poverty Analysis. *Research on Economic Inequality: A Research Annual*, 8:31–38, 1998a. [p328, 333, 335]

S. P. Jenkins and P. J. Lambert. Three 'I's of Poverty Curves and Poverty Dominance: Tips for Poverty Analysis. *Research on Economic Inequality: A Research Annual*, 8:39–56, 1998b. URL https://doi.org/10.1093/oxfordjournals.oep.a028611. [p328, 333, 335]

S. P. Jenkins and P. Van Kerm. **glcurve***: Stata Module to Derive Generalised Lorenz Curve Ordinates*, 2004. URL https://ideas.repec.org/c/boc/bocode/s366302.html. [p328]

D. A. Kodde and F. Palm. Wald Criteria for Jointly Testing Equality and Inequality Restrictions. *Econometrica*, 54(5):1243–48, 1986. [p335]

N. T. Longford. *Statistical Studies of Income, Poverty and Inequality in Europe Computing and Graphics in R Using EU-SILC*. Statistics in the Social and Behavioral Sciences. Chapman & Hall/CRC Press, 2015. ISBN 978-1-4665-6832-7. [p329]

D. Plat. **IC2***: Inequality and Concentration Indices and Curves*, 2012. URL https://CRAN.R-project.org/package=IC2. R package version 1.0-1. [p328]

H. M. Ramos, J. Ollero, and M. A. Sordo. A Sufficient Condition for Generalized Lorenz Order. *Journal of Economic Theory*, 90(2):286–292, 2000. URL https://doi.org/10.1006/jeth.1999.2606. [p334]

A. F. Shorrocks. Ranking Income Distributions. *Economica*, 50:3–17, 1983. URL https://doi.org/10.2307/2554117. [p328, 333, 334]

A. F. Shorrocks. Revisiting the Sen Poverty Index. *Econometrica*, 63(5):1225–1230, 1995. URL https://doi.org/10.2307/2171728. [p331, 333]

A. F. Shorrocks. Deprivation profiles and deprivation indices. In S. P. Jenkins, A. Kapteyn, and B. M. S. van Praag, editors, *The Distribution of Welfare and Household Production: International Perspectives*, chapter 11, pages 250–267. Cambridge University Press, Cambridge, 1998. [p333]

M. A. Sordo and C. D. Ramos. Poverty Comparisons When TIP Curves Intersect. *SORT-Statistics and Operations Research Transactions*, 35:65–80, 2011. [p328]

M. A. Sordo, H. M. Ramos, and C. D. Ramos. Poverty Measures and Poverty Orderings. *SORT-Statistics and Operations Research Transactions*, 31(2):169–180, 2007. [p335]

M. A. Sordo, C. D. Ramos, and A. Berihuete. Bienestar, Desigualdad y Pobreza En Andalucía: Un Estudio Comparativo Con El Resto De España a Partir De Las Encuestas De Condiciones De Vida 2006 y 2012. *Statistics: Colección Actualidad*, 71, 2014. ISSN 1699-8294. [p328]

M. A. Sordo, A. Berihuete, C. D. Ramos, and H. Ramos. On a Property of Lorenz Curves with Monotone Elasticity and Its Application to the Study of Inequality by Using Tax Data. *SORT-Statistics and Operations Research Transactions*, 41, 2017. URL https://doi.org/10.2436/20.8080.02.50. [p328]

P. Van Kerm and S. P. Jenkins. Generalized Lorenz Curves and Related Graphs: An Update for Stata 7. *Stata Journal*, 1(1):107–112, 2001. URL http://EconPapers.repec.org/RePEc:tsj:stataj:v:1:y:2001:i:1:p:107-112. [p328]

F. A. Wolak. Testing Inequality Constraints in Linear Econometric Models. *Journal of Econometrics*, 41 (2):205–235, 1989. URL https://doi.org/10.1016/0304-4076(89)90094-8. [p335]

K. Xu. Asymptotically Distribution-Free Statistical Test for Generalized Lorenz Curves: An Alternative Approach. *Journal of Income Distribution*, 7(1):45–62, 1997. URL https://doi.org/10.1016/S0926-6437(97)80004-2. [p329, 333, 335]

K. Xu and L. Osberg. A Distribution-Free Test for Deprivation Dominance. *Econometric Reviews*, 17(4): 415–429, 1998. URL http://doi.org/10.1080/07474939808800425. [p329, 334, 335, 336, 337]

A. Zeileis. **ineq**: *Measuring Inequality Concentration and Poverty*, 2014. URL https://CRAN.R-project.org/package=ineq. R package version 0.2-13. [p328]

B. Zheng. Aggregate Poverty Measures. *Journal of Economic Surveys*, 11(2):123–162, 1997. URL https://doi.org/10.1111/1467-6419.00028. [p331]

Angel Berihuete
Departamento de Estadística e Investigación Operativa
Universidad de Cádiz
Spain
ORCiD: 0000-0002-8589-4423
angel.berihuete@uca.es

Carmen Dolores Ramos
Departamento de Estadística e Investigación Operativa
Universidad de Cádiz
Spain
ORCiD: 0000-0002-0134-605X
carmen.ramos@uca.es

Miguel Angel Sordo
Departamento de Estadística e Investigación Operativa
Universidad de Cádiz
Spain
ORCiD: 0000-0003-1383-9051
mangel.sordo@uca.es

Appendix

Users can create their own load function using the `loadLCS` function as a template (see the example below). The load function must produce a data frame with the following variable names: DB010, DB020, DB040, DB090, HX040, HX050, HX090 (see Table 1 to set properly the variables). In the case of the Spanish Living Conditions Survey, for instance, the Spanish Statistical Office delivers four files with many variables. We only require some of them to work with **rtip**:

- 'lcs_d_file.csv':
  - DB010: a numeric vector containing the year of the survey.
  - DB020: a factor with one level which is the country considered.
  - DB030: a numeric vector containing the household ID.
  - DB040: a factor with as many levels as there are regions in the country.

  - DB090: a numeric vector containing information about household cross-sectional weight.

- 'lcs_hfile.csv':
  - HB010: a numeric vector containing the year of the survey.
  - HB030: a numeric vector containing the household ID.
  - HX040: an integer vector containing information about the household size.
  - HX240: a numeric vector containing information about the equivalised household size. The scale employed is the modified OECD scale.
  - vhRentaa: a numeric vector containing the total disposable household income.

A function to load these files and variables is the following:

```
loadLCS <- function(lcs_d_file, lcs_h_file){

  dataset1 <- read.table(lcs_d_file, header=TRUE, sep= ',')
  dataset2 <- read.table(lcs_h_file, header=TRUE, sep= ',')

  check1 <- identical(dataset1$DB010, dataset2$HB010)
  check2 <- identical(dataset1$DB030, dataset2$HB030)

  if (!check1) {
    stop('Different years!')
  } else if (!check2) {
    stop('You do not have the same identification for homes')
  } else {
    subdataset1 <- subset(dataset1, select = c("DB010", "DB020","DB030",
                                               "DB040", "DB090"))
    subdataset2 <- subset(dataset2, select = c("HB010", "HB030", "HX040",
                                               "HX240", "vhRentaa"))

    subdataset2$HX050 <- subdataset2$HX240
    subdataset2$HX090 <- subdataset2$vhRentaa/subdataset2$HX240

    dataset <- cbind(subdataset1, subdataset2)
    dataset <- subset(dataset, select = c("DB010", "DB020","DB040",
                                          "DB090", "HX040", "HX050",
                                          "HX090"))
    return(dataset)
  }
}
```

The code required to reproduce the examples in this paper can be downloaded from: https://gist.github.com/AngelBerihuete/7e88d55845044ce04a9e61edcd5954f2.

dimRed and coRanking—Unifying Dimensionality Reduction in R

by Guido Kraemer, Markus Reichstein, and Miguel D. Mahecha

Abstract "Dimensionality reduction" (DR) is a widely used approach to find low dimensional and interpretable representations of data that are natively embedded in high-dimensional spaces. DR can be realized by a plethora of methods with different properties, objectives, and, hence, (dis)advantages. The resulting low-dimensional data embeddings are often difficult to compare with objective criteria. Here, we introduce the **dimRed** and **coRanking** packages for the R language. These open source software packages enable users to easily access multiple classical and advanced DR methods using a common interface. The packages also provide quality indicators for the embeddings and easy visualization of high dimensional data. The **coRanking** package provides the functionality for assessing DR methods in the co-ranking matrix framework. In tandem, these packages allow for uncovering complex structures high dimensional data. Currently 15 DR methods are available in the package, some of which were not previously available to R users. Here, we outline the **dimRed** and **coRanking** packages and make the implemented methods understandable to the interested reader.

Introduction

Dimensionality Reduction (DR) essentially aims to find low dimensional representations of data while preserving their key properties. Many methods exist in literature, optimizing different criteria: maximizing the variance or the statistical independence of the projected data, minimizing the reconstruction error under different constraints, or optimizing for different error metrics, just to name a few. Choosing an inadequate method may imply that much of the underlying structure remains undiscovered. Often the structures of interest in a data set can be well represented by fewer dimensions than exist in the original data. Data compression of this kind has the additional benefit of making the encoded information better conceivable to our brains for further analysis tasks like classification or regression problems.

For example, the morphology of a plant's leaves, stems, and seeds reflect the environmental conditions the species usually grow in (e.g., plants with large soft leaves will never grow in a desert but might have an advantage in a humid and shadowy environment). Because the morphology of the entire plant depends on the environment, many morphological combinations will never occur in nature and the morphological space of all plant species is tightly constrained. Díaz et al. (2016) found that out of six observed morphological characteristics only two embedding dimensions were enough to represent three quarters of the totally observed variability.

DR is a widely used approach for the detection of structure in multivariate data, and has applications in a variety of fields. In climatology, DR is used to find the modes of some phenomenon, e.g., the first Empirical Orthogonal Function of monthly mean sea surface temperature of a given region over the Pacific is often linked to the El Niño Southern Oscillation or ENSO (e.g., Hsieh, 2004). In ecology the comparison of sites with different species abundances is a classical multivariate problem: each observed species adds an extra dimension, and because species are often bound to certain habitats, there is a lot of redundant information. Using DR is a popular technique to represent the sites in few dimensions, e.g., Aart (1972) matches wolfspider communities to habitat and Morrall (1974) match soil fungi data to soil types. (In ecology the general name for DR is ordination or indirect gradient analysis.) Today, hyperspectral satellite imagery collects so many bands that it is very difficult to analyze and interpret the data directly. Resuming the data into a set of few, yet independent, components is one way to reduce complexity (e.g., see Laparra et al., 2015). DR can also be used to visualize the interiors of deep neural networks (e.g., see Han et al., 2017), where the high dimensionality comes from the large number of weights used in a neural network and convergence can be visualized by means of DR. We could find many more example applications here but this is not the main focus of this publication.

The difficulty in applying DR is that each DR method is designed to maintain certain aspects of the original data and therefore may be appropriate for one task and inappropriate for another. Most methods also have parameters to tune and follow different assumptions. The quality of the outcome may strongly depend on their tuning, which adds additional complexity. DR methods can be modeled after physical models with attracting and repelling forces (Force Directed Methods), projections onto low dimensional planes (PCA, ICA), divergence of statistical distributions (SNE family), or the reconstruction of local spaces or points by their neighbors (LLE).

As an example for how changing internal parameters of a method can have a great impact, the breakthrough for Stochastic Neighborhood Embedding (SNE) methods came when a Student's *t*-

distribution was used instead of a normal distribution to model probabilities in low dimensional space to avoid the "crowding problem", that is, a sphere in high dimensional space has a much larger volume than in low dimensional space and may contain too many points to be represented accurately in few dimensions. The t-distribution, allows medium distances to be accurately represented in few dimensions by larger distances due to its heavier tails. The result is called in t-SNE and is especially good at preserving local structures in very few dimensions, this feature made t-SNE useful for a wide array of data visualization tasks and the method became much more popular than standard SNE (around six times more citations of van der Maaten and Hinton (2008) compared to Hinton and Roweis (2003) in Scopus (Elsevier, 2017)).

There are a number of software packages for other languages providing collections of methods: In Python there is scikit-learn (Pedregosa et al., 2011), which contains a module for DR. In Julia we currently find ManifoldLearning.jl for nonlinear and MultivariateStats.jl for linear DR methods. There are several toolboxes for DR implemented in Matlab (Van Der Maaten et al., 2009; Arenas-Garcia et al., 2013). The Shogun toolbox (Sonnenburg et al., 2017) implements a variety of methods for dimensionality reduction in C++ and offers bindings for a many common high level languages (including R, but the installation is anything but simple, as there is no CRAN package). However, there is no comprehensive package for R and none of the former mentioned software packages provides means to consistently compare the quality of different methods for DR.

For many applications it can be difficult to objectively find the right method or parameterization for the DR task. This paper presents the **dimRed** and **coRanking** packages for the popular programming language R. Together, they provide a standardized interface to various dimensionality reduction methods and quality metrics for embeddings. They are implemented using the S4 class system of R, making the packages both easy to use and to extend.

The design goal for these packages is to enable researchers, who may not necessarily be experts in DR, to apply the methods in their own work and to objectively identify the most suitable methods for their data. This paper provides an overview of the methods collected in the packages and contains examples as to how to use the packages.

The notation in this paper will be as follows: $X = [x_i]_{1\leq i\leq n}^T \in \mathbb{R}^{n\times p}$, and the observations $x_i \in \mathbb{R}^p$. These observations may be transformed prior to the dimensionality reduction step (e.g., centering and/or standardization) resulting in $X' = [x'_i]_{1\leq i\leq n}^T \in \mathbb{R}^{n\times p}$. A DR method then embeds each vector in X' onto a vector in $Y = [y_i]_{1\leq i\leq n}^T \in \mathbb{R}^{n\times q}$ with $y_i \in \mathbb{R}^q$, ideally with $q \ll p$. Some methods provide an explicit mapping $f(x'_i) = y_i$. Some even offer a inverse mapping $f^{-1}(y_i) = \hat{x}'_i$, such that one can reconstruct a (usually approximate) sample from the low-dimensional representation. For some methods, pairwise distances between points are needed, we set $d_{ij} = d(x_i, x_j)$ and $\hat{d}_{ij} = d(y_i, y_j)$, where d is some appropriate distance function.

When referring to `functions` in the **dimRed** package or base R simply the function name is mentioned, functions from other packages are referenced with their namespace, as with `package::function`.

Dimensionality Reduction Methods

In the following section we do not aim for an exhaustive explanation to every method in **dimRed** but rather to provide a general idea on how the methods work. An overview and classification of the most commonly used DR methods can be found in Figure 1.

In all methods, parameters have to be optimized or decisions have to be made, even if it is just about the preprocessing steps of data. The **dimRed** package tries to make the optimization process for parameters as easy as possible, but, if possible, the parameter space should be narrowed down using prior knowledge. Often decisions can be made based on theoretical knowledge. For example, sometimes an analysis requires data to be kept in their original scales and sometimes this is exactly what has to be avoided as when comparing different physical units. Sometimes decisions based on the experience of others can be made, e.g., the Gaussian kernel is probably the most universal kernel and therefore should be tested first if there is a choice.

All methods presented here have the embedding dimensionality, q, as a parameter (or `ndim` as a parameter for `embed`). For methods based on eigenvector decomposition, the result generally does not depend on the number of dimensions, i.e., the first dimension will be the same, no matter if we decide to calculate only two dimensions or more. If more dimensions are added, more information is maintained, the first dimension is the most important and higher dimensions are successively less important. This means, that a method based on eigenvalue decomposition only has to be run once if one wishes to compare the embedding in different dimensions. In optimization based methods this is generally not the case, the number of dimensions has to be chosen a priori, an embedding of 2 and 3 dimensions may vary significantly, and there is no ordered importance of dimensions. This means that comparing dimensions of optimization-based methods is computationally much more expensive.

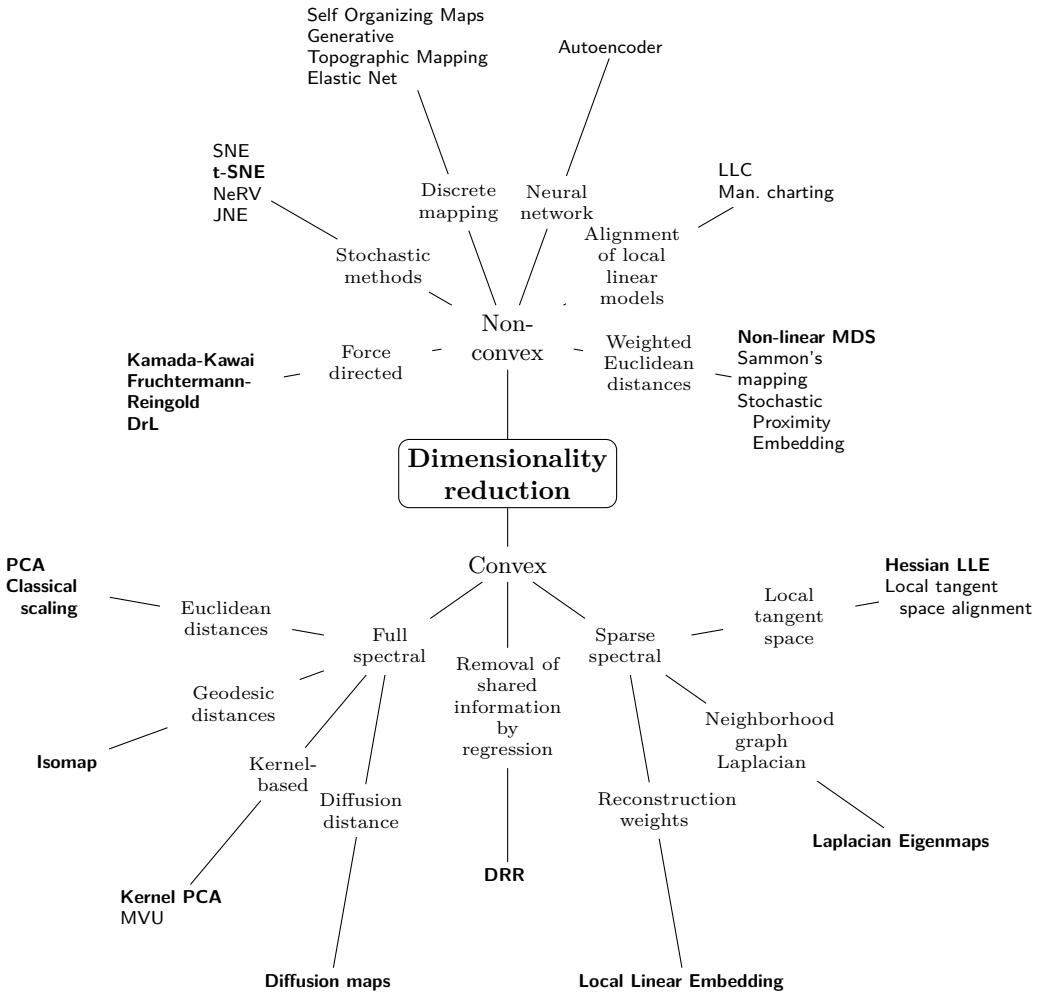


Figure 1: Classification of dimensionality reduction methods. Methods in bold face are implemented in **dimRed**. Modified from Van Der Maaten et al. (2009).

We try to give the computational complexity of the methods. Because of the actual implementation, computation times may differ largely. R is an interpreted language, so all parts of an algorithm that are implemented in R often will tend to be slow compared to methods that call efficient implementations in a compiled language. Methods where most of the computing time is spent for eigenvalue decomposition do have very efficient implementations as R uses optimized linear algebra libraries. Although, eigenvalue decomposition itself does not scale very well in naive implementations ($\mathcal{O}(n^3)$).

PCA

Principal Component Analysis (PCA) is the most basic technique for reducing dimensions. It dates back to Pearson (1901). PCA finds a linear projection (U) of the high dimensional space into a low dimensional space $Y = XU$, maintaining maximum variance of the data. It is based on solving the following eigenvalue problem:

$$(C_{XX} - \lambda_k I)u_k = 0 \tag{1}$$

where $C_{XX} = \frac{1}{n}X^T X$ is the covariance matrix, λ_k and u_k are the k-th eigenvalue and eigenvector, and I is the identity matrix. The equation has several solutions for different values of λ_k (leaving aside the trivial solution $u_k = 0$). PCA can be efficiently applied to large data sets, because it computationally scales as $\mathcal{O}(np^2 + p^3)$, that is, it scales linearly with the number of samples and R uses specialized linear algebra libraries for such kind of computations.

PCA is a rotation around the origin and there exist a forward and inverse mapping. PCA may suffer from a scale problem, i.e., when one variable dominates the variance simply because it is in a higher scale, to remedy this, the data can be scaled to zero mean and unit variance, depending on the use case, if this is necessary or desired.

Base R implements PCA in the functions `prcomp` and `princomp`; but several other implementations exist i.e., **pcaMethods** from Bioconductor which implements versions of PCA that can deal with missing data. The **dimRed** package wraps `prcomp`.

kPCA

Kernel Principal Component Analysis (kPCA) extends PCA to deal with nonlinear dependencies among variables. The idea behind kPCA is to map the data into a high dimensional space using a possibly non-linear function ϕ and then to perform a PCA in this high dimensional space. Some mathematical tricks are used for efficient computation.

If the columns of X are centered around 0, then the principal components can also be computed from the inner product matrix $K = X^T X$. Due to this way of calculating a PCA, we do not need to explicitly map all points into the high dimensional space and do the calculations there, it is enough to obtain the inner product matrix or kernel matrix $K \in \mathbb{R}^{n \times n}$ of the mapped points (Schölkopf et al., 1998).

Here is an example calculating the kernel matrix using a Gaussian kernel:

$$K = \phi(x_i)^T \phi(x_j) = \kappa(x_i, x_j) = \exp\left(-\frac{\|x_i - x_j\|^2}{2\sigma^2}\right), \tag{2}$$

where σ is a length scale parameter accounting for the width of the kernel. The other trick used is known as the "representers theorem." The interested reader is referred to Schölkopf et al. (2001).

The kPCA method is very flexible and there exist many kernels for special purposes. The most common kernel function is the Gaussian kernel (Equation 2). The flexibility comes at the price that the method has to be finely tuned for the data set because some parameter combinations are simply unsuitable for certain data. The method is not suitable for very large data sets, because memory scales with $\mathcal{O}(n^2)$ and computation time with $\mathcal{O}(n^3)$.

Diffusion Maps, Isomap, Locally Linear Embedding, and some other techniques can be seen as special cases of kPCA. In which case, an out-of-sample extension using the Nystöm formula can be applied (Bengio et al., 2004). This can also yield applications for bigger data, where an embedding is trained with a sub-sample of all data and then the data is embedded using the Nyström formula.

Kernel PCA in R is implemented in the **kernlab** package using the function `kernlab::kpca`, and supports a number of kernels and user defined functions. For details see the help page for `kernlab::kpca`.

The **dimRed** package wraps `kernlab::kpca` but additionally provides forward and inverse methods (Bakir et al., 2004) which can be used to fit out-of sample data or to visualize the transformation of

the data space.

Classical Scaling

What today is called Classical Scaling was first introduced by Torgerson (1952). It uses an eigenvalue decomposition of a transformed distance matrix to find an embedding that maintains the distances of the distance matrix. The method works because of the same reason that kPCA works, i.e., classical scaling can be seen as a kPCA with kernel $x^T y$. A matrix of Euclidean distances can be transformed into an inner product matrix by some simple transformations and therefore yields the same result as a PCA. Classical scaling is conceptually more general than PCA in that arbitrary distance matrices can be used, i.e., the method does not even need the original coordinates, just a distance matrix D. Then it tries to find and embedding Y so that $\hat{d}_{ij}$ is as similar to d_{ij} as possible.

The disadvantage is that is computationally much more demanding, i.e., an eigenvalue decomposition of a $n \times n$ matrix has to be computed. This step requires $\mathcal{O}(n^2)$ memory and $\mathcal{O}(n^3)$ computation time, while PCA requires only the eigenvalue decomposition of a $d \times d$ matrix and usually $n \gg d$. R implements classical scaling in the `cmdscale` function.

The **dimRed** package wraps `cmdscale` and allows the specification of arbitrary distance functions for calculating the distance matrix. Additionally a forward method is implemented.

Isomap

As Classical Scaling can deal with arbitrarily defined distances, Tenenbaum et al. (2000) suggested to approximate the structure of the manifold by using geodesic distances. In practice, a graph is created by either keeping only the connections between every point and its k nearest neighbors to produce a k-nearest neighbor graph (k-NNG), or simply by keeping all distances smaller than a value ε producing an ε-neighborhood graph (ε-NNG). Geodesic distances are obtained by recording the distance on the graph and classical scaling is used to find an embedding in fewer dimensions. This leads to an "unfolding" of possibly convoluted structures (see Figure 3).

Isomap's computational cost is dominated by the eigenvalue decomposition and therefore scales with $\mathcal{O}(n^3)$. Other related techniques can use more efficient algorithms because the distance matrix becomes sparse due to a different preprocessing.

In R, Isomap is implemented in the **vegan** package. `vegan::isomap` calculates an Isomap embedding and `vegan::isomapdist` calculates a geodesic distance matrix. The **dimRed** package uses its own implementation. This implementation is faster mainly due to using a KD-tree for the nearest neighbor search (from the **RANN** package) and to a faster implementation for the shortest path search in the k-NNG (from the **igraph** package). The implementation in **dimRed** also includes a forward method that can be used to train the embedding on a subset of data points and then use these points to approximate an embedding for the remaining points. This technique is generally referred to as landmark Isomap (De Silva and Tenenbaum, 2004).

Locally Linear Embedding

Points that lie on a manifold in a high dimensional space can be reconstructed through linear combinations of their neighborhoods if the manifold is well sampled and the neighbohoods lie on a locally linear patch. These reconstruction weights, W, are the same in the high dimensional space as the internal coordinates of the manifold. Locally Linear Embedding (LLE; Roweis and Saul, 2000) is a technique that constructs a weight matrix $W \in \mathbb{R}^{n \times n}$ with elements w_{ij} so that

$$\sum_{i=1}^{n} \left\| x_i - \sum_{j=1}^{n} w_{ij} x_j \right\|^2 \tag{3}$$

is minimized under the constraint that $w_{ij} = 0$ if x_j does not belong to the neighborhood and the constraint that $\sum_{j=1}^{n} w_{ij} = 1$. Finally the embedding is made in such a way that the following cost function is minimized for Y,

$$\sum_{i=1}^{n} \left\| y_i - \sum_{j=1}^{n} w_{ij} y_j \right\|^2. \tag{4}$$

This can be solved using an eigenvalue decomposition.

Conceptually the method is similar to Isomap but it is computationally much nicer because the weight matrix is sparse and there exist efficient solvers. In R, LLE is implemented by the package **lle**, the embedding can be calculated with `lle::lle`. Unfortunately the implementation does not make

use of the sparsity of the weight matrix W. The manifold must be well sampled and the neighborhood size must be chosen appropriately for LLE to give good results.

Laplacian Eigenmaps

Laplacian Eigenmaps were originally developed under the name spectral clustering to separate non-convex clusters. Later it was also used for graph embedding and DR (Belkin and Niyogi, 2003).

A number of variants have been proposed. First, a graph is constructed, usually from a distance matrix, the graph can be made sparse by keeping only the k nearest neighbors, or by specifying an ε neighborhood. Then, a similarity matrix W is calculated by using a Gaussian kernel (see Equation 2), if $c = 2\sigma^2 = \infty$, then all distances are treated equally, the smaller c the more emphasis is given to differences in distance. The degree of vertex i is $d_i = \sum_{j=1}^{n} w_{ij}$ and the degree matrix, D, is the diagonal matrix with entries d_i. Then we can form the graph Laplacian $L = D - W$ and, then, there are several ways how to proceed, an overview can be found in Luxburg (2007).

The **dimRed** package implements the algorithm from Belkin and Niyogi (2003). Analogously to LLE, Laplacian eigenmaps avoid computational complexity by creating a sparse matrix and not having to estimate the distances between all pairs of points. Then the eigenvectors corresponding to the lowest eigenvalues larger than 0 of either the matrix L or the normalized Laplacian $D^{-1/2}LD^{-1/2}$ are computed and form the embedding.

Diffusion Maps

Diffusion Maps (Coifman and Lafon, 2006) take a distance matrix as input and calculates the transition probability matrix P of a diffusion process between the points to approximate the manifold. Then the embedding is done by an eigenvalue decompositon of P to calculate the coordinates of the embedding. The algorithm for calculating Diffusion Maps shares some elements with the way Laplacian Eigenmaps are calculated. Both algorithms depart from the same weight matrix, Diffusion Map calculate the transition probability on the graph after t time steps and do the embedding on this probability matrix.

The idea is to simulate a diffusion process between the nodes of the graph, which is more robust to short-circuiting than the k-NNG from Isomap (see bottom right Figure 3). Diffusion maps in R are accessible via the `diffusionMap::diffuse()` function, which is available in the **diffusionMap** package. Additional points can be approximated into an existing embedding using the Nyström formula (Bengio et al., 2004). The implementation in **dimRed** is based on the `diffusionMap::diffuse` function.

non-Metric Dimensional Scaling

While Classical Scaling and derived methods (see section Classical Scaling) use eigenvector decomposition to embed the data in such a way that the given distances are maintained, non-Metric Dimensional Scaling (nMDS, Kruskal, 1964a,b) uses optimization methods to reach the same goal. Therefore a stress function,

$$S = \sqrt{\frac{\sum_{i<j} \left(d_{ij} - \hat{d}_{ij}\right)^2}{\sum_{i<j} d_{ij}^2}}, \tag{5}$$

is used, and the algorithm tries to embed y_i in such a way that the order of the d_{ij} is the same as the order of the $\hat{d}_{ij}$ Because optimization methods can fit a wide variety of problems, there are very loose limits set to the form of the error or stress function. For instance Mahecha et al. (2007) found that nMDS using geodesic distances can be almost as powerful as Isomap for embedding biodiversity patterns. Because of the flexibility of nMDS, there is a whole package in R devoted to Multidimensional Scaling, `smacof` (de Leeuw and Mair, 2009).

Several packages provide implementations for nMDS in R, for example **MASS** and **vegan** with the functions `MASS::isoMDS` and `vegan::monoMDS`. Related methods include Sammons Mapping which con be found as `MASS::sammon`. The **dimRed** package wraps `vegan::monoMDS`.

Force Directed Methods

The data X can be considered as a graph with weighted edges, where the weights are the distances between points. Force directed algorithms see the edges of the graphs as springs or the result of an electric charge of the nodes that result in an attractive or repulsive force between the nodes, the

algorithms then try to minimize the overall energy of the graph.

$$E = \sum_{i<j} k_{ij}(d_{ij} - \hat{d}_{ij})^2, \tag{6}$$

where k_{ij} is the spring constant for the spring connecting points i and j.

Graph embedding algorithms generally suffer from long running times (though compared to other methods presented here they do not scale as badly) and many local optima. This is why a number of methods have been developed that try to deal with some of the shortcomings, for example, the Kamada-Kawai (Kamada and Kawai, 1989), the Fruchtermann-Reingold (Fruchterman and Reingold, 1991), or the DrL (Martin et al., 2007) algorithms.

There are a number of graph embedding algorithms included in the **igraph** package, they can be accessed using the `igraph::layout_with_*` function family. The **dimRed** package only wraps the three algorithms mentioned above; there are many others which are not interesting for dimensionality reduction.

t-SNE

Stochastic Neighbor Embedding (SNE; Hinton and Roweis, 2003) is a technique that minimizes the Kullback-Leibler divergence of scaled similarities of the points i and j in a high dimensional space, p_{ij}, and a low dimensional space, q_{ij}:

$$KL(P\|Q) = \sum_{i \neq j} p_{ij} \log \frac{p_{ij}}{q_{ij}}. \tag{7}$$

SNE uses a Gaussian kernel (see Equation 2) to compute similarities in a high and a low dimensional space. The t-Distributed Stochastic Neighborhood Embedding (t-SNE; van der Maaten and Hinton, 2008) improves on SNE by using a t-Distribution as a kernel in low dimensional space. Because of the heavy-tailed t-distribution, t-SNE maintains local neighborhoods of the data better and penalizes wrong embeddings of dissimilar points. This property makes it especially suitable to represent clustered data and complex structures in few dimensions.

The t-SNE method has one parameter, perplexity, to tune. This determines the neighborhood size of the kernels used.

The general runtime of t-SNE is $\mathcal{O}(n^2)$, but an efficient implementation using tree search algorithms that scales as $\mathcal{O}(n \log n)$ exists and can be found in the **Rtsne** package in R. The t-SNE implementation in **dimRed** wraps the **Rtsne** package.

There exist a number of derived techniques for dimensionality reduction, e.g., NeRV (Venna et al., 2010) and JNE (Lee et al., 2013), that improve results but for which there do not yet exist packages on CRAN implementing them.

ICA

Independent Component Analysis (ICA) interprets the data X as a mixture of independent signals, e.g., a number of sound sources recorded by several microphones, and tries to "un-mix" them to find the original signals in the recorded signals. ICA is a linear rotation of the data, just as PCA, but instead of recovering the maximum variance, it recovers statistically independent components. A signal matrix S and a mixing matrix A are estimated so that $X = AS$.

There are a number of algorithms for ICA, the most widely used is fastICA (Hyvarinen, 1999) because it provides a fast and robust way to estimate A and S. FastICA maximizes a measure for non-Gaussianity called negentropy J (Comon, 1994). This is equivalent to minimizing mutual information between the resulting components. Negentropy J is defined as follows:

$$H(u) = -\int f(u) \log f(Y)\, \mathrm{d}u, \tag{8}$$

$$J(u) = H(u_{\text{gauss}}) - H(u), \tag{9}$$

where $u = (u_1, \ldots, u_n)^T$ is a random vector with density $f(\cdot)$ and u_{gauss} is a Gaussian random variable with the same covariance structure as u. FastICA uses a very efficient approximation to calculate negentropy. Because ICA can be translated into a simple linear projection, a forward and an inverse method can be supplied.

There are a number of packages in R that implement algorithms for ICA, the **dimRed** package

wraps the `fastICA::fastICA()` function from **fastICA**.

DRR

Dimensionality Reduction via Regression is a very recent technique extending PCA (Laparra et al., 2015). Starting from a rotated (PCA) solution $X' = XU$, it predicts redundant information from the remaining components using non-linear regression.

$$y_{\cdot i} = x'_{\cdot i} - f_i(x'_{\cdot 1}, x'_{\cdot 2}, \ldots, x'_{\cdot i-1}) \tag{10}$$

with $x_{\cdot i}$ and $y_{\cdot i}$ being the loading of observations on the i-th axis. In theory, any kind of regression can be used. the authors of the original paper choose Kernel Ridge Regression (KRR; Saunders et al., 1998) because it is a flexible nonlinear regression technique and computational optimizations for a fast calculation exist. DRR has another advantage over other techniques presented here, because it provides an exact forward and inverse function.

The use of KRR also has the advantage of making the method convex, here we list it under non-convex methods, because other types of regression may make it non-convex.

Mathematicaly, functions are limited to map one input to a single output point. Therefore, DRR reduces to PCA if manifolds are too complex; but it seems very useful for slightly curved manifolds. The initial rotation is important, because the result strongly depends on the order of dimensions in high dimensional space.

DRR is implemented in the package **DRR**. The package provides forward and inverse functions which can be used to train on a subset.

Quality criteria

The advantage of unsupervised learning is that one does not need to specify classes or a target variable for the data under scrutiny. Instead the chosen algorithm arranges the input data. For example, arranged into clusters or into a lower dimensional representation. In contrast to a supervised problem, there is no natural way to directly measure the quality of any output or to compare two methods by an objective measure like for instance modeling efficiency or classification error. The reason is that every method optimizes a different error function, and it would be unfair to compare t-SNE and PCA by means of either recovered variance or KL-Divergence. One fair measure would be the reconstruction error, i.e., reconstructing the original data from a limited number of dimensions, but as discussed above not many methods provide forward and inverse mappings.

However, there are a series of independent estimators on the quality of a low-dimensional embedding. The **dimRed** package provides a number of quality measures which have been proposed in the literature to measure performance of dimensionality reduction techniques.

Co-ranking matrix based measures

The co-ranking matrix (Lee and Verleysen, 2009) is a way to capture the changes in ordinal distance. As before, let $d_{ij} = d(x_i, x_j)$ be the distances between x_i and x_j, i.e., in high dimensional space and $\hat{d}_{ij} = d(y_i, y_j)$ the distances in low dimensional space, then we can define the rank of y_j with respect to y_i

$$\hat{r}_{ij} = |\{k : \hat{d}_{ik} < \hat{d}_{ij} \text{ or } (\hat{d}_{ik} = \hat{d}_{ij} \text{ and } 1 \leq k < j \leq n)\}|, \tag{11}$$

and, analogously, the rank in high-dimensional space as:

$$r_{ij} = |\{k : d_{ik} < d_{ij} \text{ or } (d_{ik} = d_{ij} \text{ and } 1 \leq k < j \leq n)\}|, \tag{12}$$

where the notation $|A|$ denotes the number of elements in a set A. This means that we simply replace the distances in a distance matrix column wise by their ranks. This means, that r_{ij} is an integer which indicates that x_i is the r_{ij}-th closest neighbor of x_j in the set X.

The co-ranking matrix Q then has elements

$$q_{kl} = |\{(i, j) : \hat{r}_{ij} = k \text{ and } r_{ij} = l\}|, \tag{13}$$

which is the 2d-histogram of the ranks. That is, q_{ij} is an integer which counts how many points of distance rank j became rank i. In a perfect DR, this matrix will only have non-zero entries in the diagonal, if most of the non-zero entries are in the lower triangle, then the DR collapsed far away

points onto each other; if most of the non-zero entries are in the upper triangle, then the DR teared close points apart. For a detailed description of the properties of the co-ranking matrix the reader is referred to Lueks et al. (2011).

The co-ranking matrix can be computed using function `coRanking::coranking()` and can be visualized using `coRanking::imageplot()`. A good embedding should scatter the values around the diagonal of the matrix. If the values are predominantly in the lower triangle, then the embedding collapses the original structure causing far away points to be much closer; if the values are predominantly in the upper triangle the points from the original structure are torn apart. Nevertheless this method requires visual inspection of the matrix. For an automated assessment of quality, a scalar value that assigns a quality to an embedding is needed.

A number of metrics can be computed from the co-ranking matrix. For example:

$$Q_{NX}(k) = \frac{1}{kn} \sum_{i=1}^{k} \sum_{j=1}^{k} q_{ij}, \tag{14}$$

which is the number of points that belong to the k-th nearest neighbors in both high- and low-dimensional space, normalized to give a maximum of 1 (Lee and Verleysen, 2009). This quantity can be adjusted for random embeddings, giving the Local Continuity Meta Criterion (Chen and Buja, 2009):

$$\mathrm{LCMC}(k) = Q_{NX}(k) - \frac{k}{n-1} \tag{15}$$

The above measures still depend on k, but LCMC has a well defined maximum at $k_{\max}$. Two measures without parameters are then defined:

$$Q_{\text{local}} = \frac{1}{k_{\max}} \sum_{k=1}^{k_{\max}} Q_{NX}(k) \text{ and} \tag{16}$$

$$Q_{\text{global}} = \frac{1}{n - k_{\max}} \sum_{k=k_{\max}}^{n-1} Q_{NX}(k). \tag{17}$$

These measure the preservation of local and global distances respectively. The original authors advised using Q_{local} over Q_{global}, but this depends on the application.

LCMC(k) can be normalized to a maximum of 1, yielding the following measure for a quality embedding (Lee et al., 2013):

$$R_{NX}(k) = \frac{(n-1)Q_{NX}(k) - k}{n - 1 - k}, \tag{18}$$

where a value of 0 corresponds to a random embedding and a value of 1 to a perfect embedding into the k-ary neighborhood. To transform $R_{NX}(k)$ into a parameterless measure, the area under the curve can be used:

$$\mathrm{AUC}_{\ln k}\left(R_{NX}(k)\right) = \left(\sum_{k=1}^{n-2} R_{NX}(k)\right) \Big/ \left(\sum_{k=1}^{n-2} 1/k\right). \tag{19}$$

This measure is normalized to one and takes k at a log-scale. Therefore it prefers methods that preserve local distances.

In R, the co-ranking matrix can be calculated using the the `coRanking::coranking` function. The **dimRed** package contains the functions `Q_local`, `Q_global`, `Q_NX`, `LCMC`, and `R_NX` to calculate the above quality measures in addition to `AUC_lnK_R_NX`.

Calculating the co-ranking matrix is a relatively expensive operation because it requires sorting every row of the distance matrix twice. It therefore scales with $\mathcal{O}(n^2 \log n)$. There is also a plotting function `plot_R_NX`, which plots the R_{NX} values with log-scaled K and adds the $AUC_{\ln K}$ to the legend (see Figure 2).

There are a number of other measures that can be computed from a co-ranking matrix, e.g., see Lueks et al. (2011); Lee and Verleysen (2009), or Babaee et al. (2013).

Cophenetic correlation

An old measure originally developed to compare clustering methods in the field of phylogenetics is cophenetic correlation (Sokal and Rohlf, 1962). This method consists simply of the correlation between the upper or lower triangles of the distance matrices (in dendrograms they are called cophenetic matrices, hence the name) in a high and low dimensional space. Additionally the distance measure and correlation method can be varied. In the **dimRed** package this is implemented in the

`cophenetic_correlation` function.

Some studies use a measure called "residual variance" (Tenenbaum et al., 2000; Mahecha et al., 2007), which is defined as

$$1 - r^2(D, \hat{D}),$$

where r is the Pearson correlation and D, $\hat{D}$ are the distances matrices consisting of elements d_{ij} and $\hat{d}_{ij}$ respectively.

Reconstruction error

The fairest and most common way to assess the quality of a dimensionality reduction when the method provides a inverse mapping is the reconstruction error. The **dimRed** package includes a function to calculate the root mean squared error which is defined as:

$$\text{RMSE} = \sqrt{\frac{1}{n}\sum_{i=1}^{n} d(x_i', x_i)^2} \tag{20}$$

with $x_i' = f^{-1}(y_i)$, f^{-1} being the function that maps an embedded value back to feature space.

The **dimRed** package provides the `reconstruction_rmse` and `reconstruction_error` functions.

Test data sets

There are a number of test data sets that are often used to showcase a dimensionality reduction technique. Common ones being the 3d S-curve and the Swiss roll, among others. These data sets have in common that they usually have three dimensions, and well defined manifolds. Real world examples usually have more dimensions and often are much noisier, the manifolds may not be well sampled and exhibit holes and large pieces may be missing. Additionally, we cannot be sure if we can observe all the relevant variables.

The **dimRed** package implements a number of test datasets that are being used in literature to benchmark methods with the function `dimRed::loadDataSet()`. For artificial datasets the number of points and the noise level can be adjusted, the function also returns the internal coordinates.

The dimRed Package

The **dimRed** package collects DR methods readily implemented in R, implements missing methods and offers means to compare the quality of embeddings. The package is open source and available under the GPL3 license. Released versions of the package are available through CRAN (https://cran.r-project.org/package=dimRed) and development versions are hosted on GitHub (https://github.com/gdkrmr/dimRed). The **dimRed** package provides a common interface and convenience functions for a variety of different DR methods so that it is made easier to use and compare different methods. An overview of the packages main functions can be found in Table 1.

Function	Description
`embed`	Embed data using a DR method.
`quality`	Calculate a quality score from the result of `embed`.
`plot`	Plot a `"dimRedData"` or `"dimRedResult"` object, colors the points automatically, for exploring the data.
`plot_R_NX`	Compares the quality of various embeddings.
`dimRedMethodList`	Returns a character vector that contains all implemented DR methods.
`dimRedQualityList`	Returns a character vector that contains all implemented quality measures.

Table 1: The main interface functions of the **dimRed** package.

Internally, the package uses S4 classes but for normal usage the user does not need to have any knowledge on the inner workings of the S4 class system in R (cf. table 2). The package contains simple conversion functions from and to standard R-objects like a data.frame or a matrix. The `"dimRedData"` class provides an container for the data to be processed. The slot `data` contains a matrix

with dimensions in columns and observations in rows, the slot `meta` may contain a data frame with additional information, e.g., categories or other information of the data points.

Class Name	Function
`"dimRedData"`	Holds the data for a DR. Fed to `embed()`. An `as.dimRedData()` methods exists for `"data.frame"`, `"matrix"`, and `"formula"` exist.
`"dimRedMethod"`	Virtual class, ancestor of all DR methods.
`"dimRedResult"`	The result of `embed()`, the embedded data.

Table 2: The S4 classes used in the **dimRed** package.

Each embedding method is a class which inherits from `"dimRedMethod"` which means that it contains a function to generate `"dimRedResult"` objects and a list of standard parameters. The class `"dimRedResult"` contains the data in reduced dimensions, the original meta information along with the original data, and, if possible, functions for the forward and inverse mapping.

From a user-perspective the central function of the package is `embed` which is called in the form `embed(data,method,...)`, `data` can take standard R objects such as instances of `"data.frame"`, `"matrix"`, or `"formula"`, as input. The `method` is given as a character vector. All available methods can be listed by calling '`dimRedMethodList()`'. Method-specific parameters can be passed through `...`; when no method-specific parameters are given, defaults are chosen. The `embed` function returns an object of class `"dimRedResult"`.

For comparing different embeddings, **dimRed** contains the function `quality` which relies on the output of `embed` and a method name. This function returns a scalar quality score; a vector that contains the names of all quality functions is returned by calling '`dimRedQualityList()`'.

For easy visual examination, the package contains `plot` methods for `"dimRedData"` and `"dimRedResult"` objects in order to plot high dimensional data using parallel plots and pairwise scatter plots. Automatic coloring of data points is done using the available metadata.

Examples

The comparison of different DR methods, choosing the right parameters for a method, and the inspection of the results is simplified by **dimRed**. This section contains a number of examples to highlight the use of the package.

To compare methods of dimensionality reduction, first a test data set is loaded using `loadDataSet`, then the `embed` function is used for DR (`embed` can also handle standard R types like `matrix` and `data.frame`). This makes it very simple to apply different methods of DR to the same data e.g., by defining a character vector of method names and then iterating over these, say with `lapply`. For inspection, **dimRed** provides methods for the `plot` function to visualize the resulting embedding (Figure 2 b and d), internal coordinates of the manifold are represented by color gradients. To visualize how well embeddings represent different neighborhood sizes, the function `plot_R_NX` is used on a list of embedding results (Figure 2 c). The plots in figure 2 are produced by the following code:

```
## define which methods to apply
embed_methods <- c("Isomap", "PCA")
## load test data set
data_set <- loadDataSet("3D S Curve", n = 1000)
## apply dimensionality reduction
data_emb <- lapply(embed_methods, function(x) embed(data_set, x))
names(data_emb) <- embed_methods
## figure 2a, the data set
plot(data_set, type = "3vars")
## figures 2b (Isomap) and 2d (PCA)
lapply(data_emb, plot, type = "2vars")
## figure 2c, quality analysis
plot_R_NX(data_emb)
```

The function `plot_R_NX` produces a figure that plots the neighborhood size (k at a log-scale) against the quality measure $R_{NX}(k)$ (see Equation 18). This gives an overview of the general behavior of methods: if R_{NX} is high for low values of K, then local neighborhoods are maintained well; if R_{NX} is

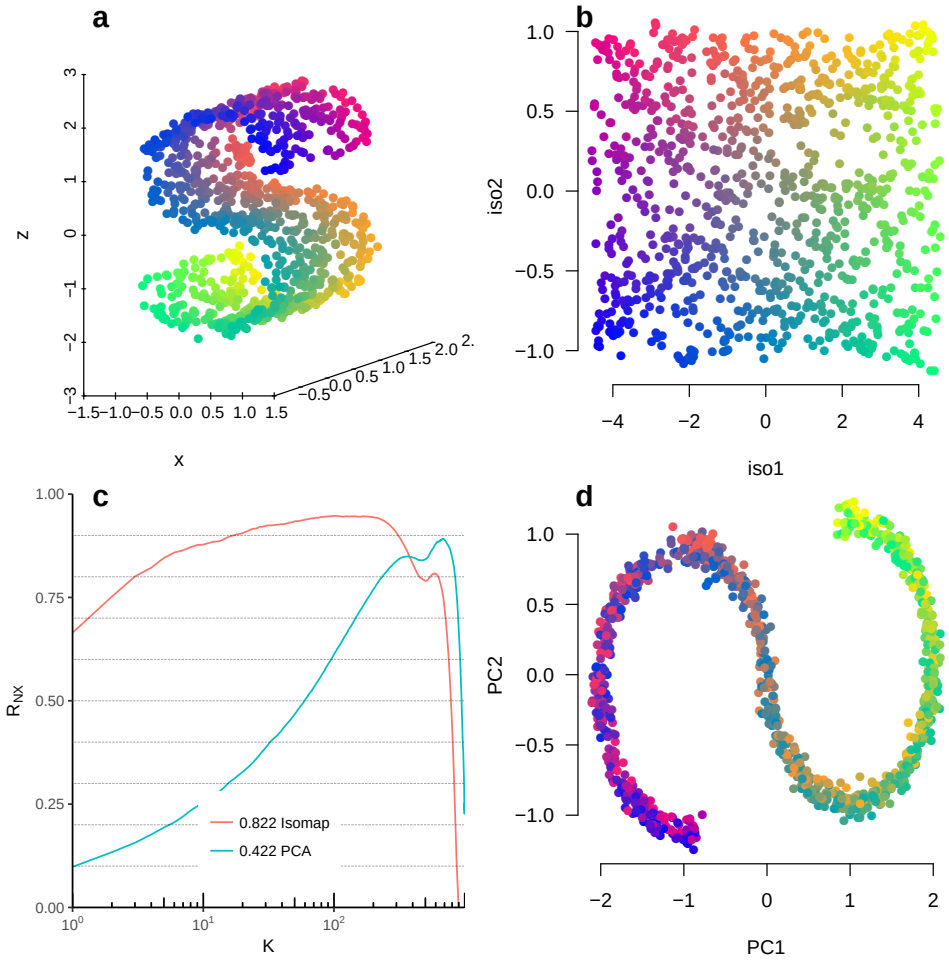


Figure 2: Comparing PCA and Isomap: (a) An S-shaped manifold, colors represent the internal coordinates of the manifold. (b) Isomap embedding, the S-shaped manifold is unfolded. (c) R_{NX} plotted agains neighborhood sizes, Isomap is much better at preserving local distances and PCA is better at preserving global Euclidean distances. The numbers on the legend are the $\text{AUC}_{1/K}$. (d) PCA projection of the data, the directions of maximum variance are preserved.

high for large values of K, then global gradients are maintained well. It also provides a way to directly compare methods by plotting more than one R_{NX} curve and an overall quality of the embedding by taking the area under the curve as an indicator for the overall quality of the embedding (see fig 19) which is shown as a number in the legend.

Therefore we can see from Figure 2c that t-SNE is very good a maintaining close and medium distances for the given data set, whereas PCA is only better at maintaining the very large distances. The large distances are dominated by the overall bent shape of the S in 3D space, while the close distances are not affected by this bending. This is reflected in the properties recovered by the different methods, the PCA embedding recovers the S-shape, while t-SNE ignores the S-shape and recovers the inner structure of the manifold.

Often the quality of an embedding strongly depends on the choice of parameters, the interface of **dimRed** can be used to facilitate searching the parameter space.

Isomap has one parameter k which determines the number of neighbors used to construct the k-NNG. If this number is too large, then Isomap will resemble an MDS (Figure 3 e), if the number is too small, the resulting embedding contains holes (Figure 3 c). The following code finds the optimal value, k_{max}, for k using the Q_{local} criterion, the results are visualized in Figure 3 a:

```
## Load data
ss <- loadDataSet("3D S Curve", n = 500)
## Parameter space
kk <- floor(seq(5, 100, length.out = 40))
## Embedding over parameter space
emb <- lapply(kk, function(x) embed(ss, "Isomap", knn = x))
```

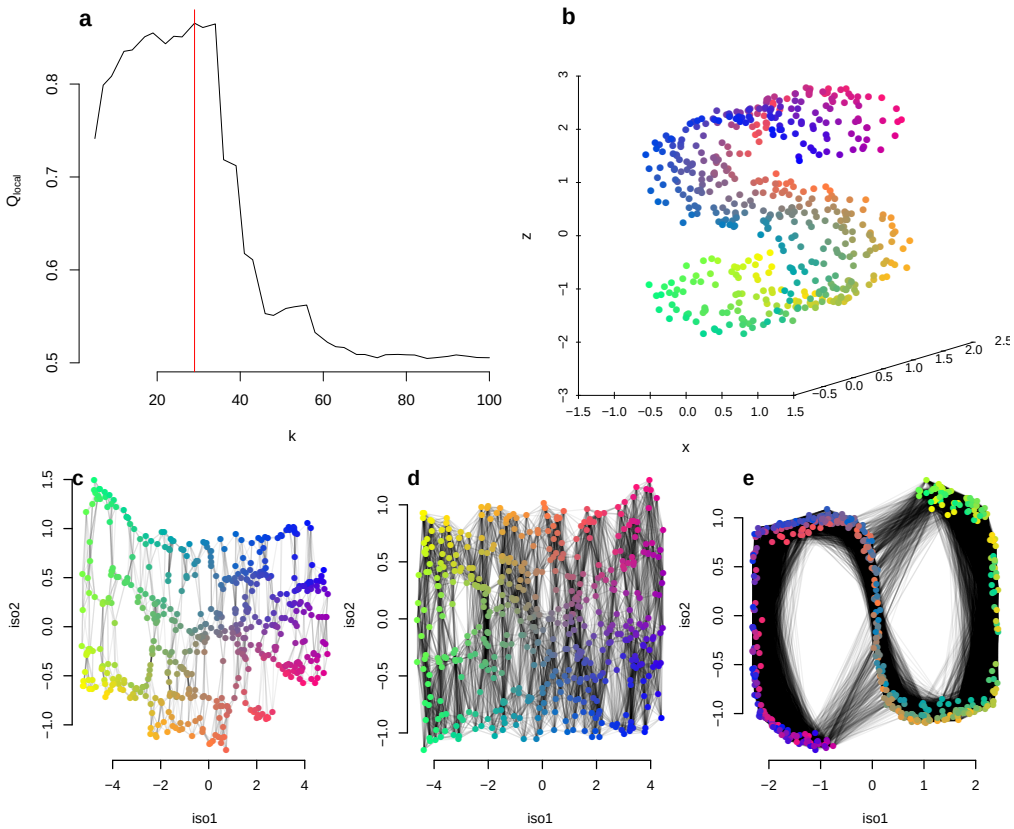


Figure 3: Using **dimRed** and the Q_{local} indicator to estimate a good value for the parameter k in Isomap. (a) Q_{local} for different values of k, the vertical red line indicates the maximum $k_{\max}$. (b) The original data set, a 2 dimensional manifold bent in an S-shape in 3 dimensional space. Bottom row: Embeddings and k-NNG for different values of k. (c) When $k = 5$, the value for k is too small resulting in holes in the embedding, the manifold itself is still unfolded correctly. (d) Choose $k = k_{\max}$, the best representation of the original manifold in two dimensions achievable with Isomap. (e) $k = 100$, too large, the k-NNG does not approximate the manifold any more.

```
## Quality over embeddings
qual <- sapply(emb, function(x) quality(x, "Q_local"))
## Find best value for K
ind_max <- which.max(qual)
k_max <- kk[ind_max]
```

Figure 3a shows how the Q_{local} criterion changes when varying the neighborhood size k for Isomap, the gray lines in Figure 3 represent the edges of the k-NN Graph. If the value for k is too low, the inner structure of the manifold will still be recovered, but it will be imperfect (Figure 3c, note that the holes appear in places that are not covered by the edges of the k-NN Graph), therefore the Q_{local} score is lower than optimal. If k is too large, the error of the embedding is much larger due to short circuiting and we observe a very steep drop in the Q_{local} score. The short circuiting can be observed in Figure 3e with the edges that cross the gap between the tips and the center of the S-shape.

It is also very easy to compare across methods and quality scores. The following code produces a matrix of quality scores and methods, where `dimRedMethodList` returns a character vector with all methods. A visualization of the matrix can be found in Figure 4.

```
embed_methods <- dimRedMethodList()
quality_methods <- c("Q_local", "Q_global", "AUC_lnK_R_NX",
                     "cophenetic_correlation")
scurve <- loadDataSet("3D S Curve", n = 2000)
quality_results <- matrix(
  NA, length(embed_methods), length(quality_methods),
  dimnames = list(embed_methods, quality_methods)
)
```

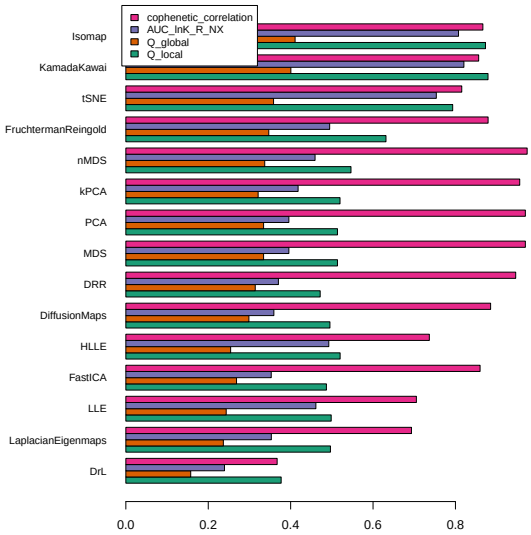


Figure 4: A visualization of the `quality_results` matrix. The methods are ordered by mean quality score. The reconstruction error was omitted, because a higher value means a worse embedding, while in the present methods a higher score means a better embedding. Parameters were not tuned for the example, therefore it should not be seen as a general quality assessment of the methods.

```
embedded_data <- list()
for (e in embed_methods) {
  embedded_data[[e]] <- embed(scurve, e)
  for (q in quality_methods)
    try(quality_results[e, q] <- quality(embedded_data[[e]], q))
}
```

This example showcases the simplicity with which different methods and quality criteria can be combined. Because of the strong dependencies on parameters it is not advised to apply this kind of analysis without tuning the parameters for each method separately. There is no automatized way to tune parameters in **dimRed**.

Conclusion

This paper presents the **dimRed** and **coRanking** packages and it provides a brief overview of the methods implemented therein. The **dimRed** package is written in the R language, one of the most popular languages for data analysis. The package is freely available from CRAN. The package is object oriented and completely open source and therefore easily available and extensible. Although most of the DR methods already had implementations in R, **dimRed** adds some new methods for dimensionality reduction, and **coRanking** adds methods for an independent quality control of DR methods to the R ecosystem. DR is a widely used technique. However, due to the lack of easily usable tools, choosing the right method for DR is complex and depends upon a variety of factors. The **dimRed** package aims to facilitate experimentation with different techniques, parameters, and quality measures so that choosing the right method becomes easier. The **dimRed** package wants to enable the user to objectively compare methods that rely on very different algorithmic approaches. It makes the life of the programmer easier, because all methods are aggregated in one place and there is a single interface and standardized classes to access the functionality.

Acknowledgments

We thank Dr. G. Camps-Valls and an anonymous reviewer for many useful comments. This study was supported by the European Space Agency (ESA) via the Earth System Data Lab project (http://earthsystemdatacube.org) and the EU via the H2020 project BACI, grant agreement No 640176.

Bibliography

P. J. M. V. D. Aart. Distribution Analysis of Wolfspiders (Araneae, Lycosidae) in a Dune Area By Means of Principal Component Analysis. *Netherlands Journal of Zoology*, 23(3):266–329, 1972. ISSN 1568-542X. URL https://doi.org/10.1163/002829673x00076. [p342]

J. Arenas-Garcia, K. B. Petersen, G. Camps-Valls, and L. K. Hansen. Kernel Multivariate Analysis Framework for Supervised Subspace Learning: A Tutorial on Linear and Kernel Multivariate Methods. *IEEE Signal Processing Magazine*, 30(4):16–29, 2013. ISSN 1053-5888. URL https://doi.org/10.1109/msp.2013.2250591. [p343]

M. Babaee, M. Datcu, and G. Rigoll. Assessment of dimensionality reduction based on communication channel model; application to immersive information visualization. In *Big Data 2013*, pages 1–6. IEEE Xplore, 2013. URL https://doi.org/10.1109/bigdata.2013.6691726. [p350]

G. H. Bakir, J. Weston, and P. B. Schölkopf. Learning to Find Pre-Images. In S. Thrun, L. K. Saul, and P. B. Schölkopf, editors, *Advances in Neural Information Processing Systems 16*, pages 449–456. MIT Press, 2004. URL https://doi.org/10.1007/978-3-540-28649-3_31. [p345]

M. Belkin and P. Niyogi. Laplacian Eigenmaps for Dimensionality Reduction and Data Representation. *Neural Computation*, 15(6):1373, 2003. ISSN 08997667. URL https://doi.org/10.1162/089976603321780317. [p347]

Y. Bengio, O. Delalleau, N. L. Roux, J.-F. Paiement, P. Vincent, and M. Ouimet. Learning Eigenfunctions Links Spectral Embedding and Kernel PCA. *Neural Computation*, 16(10):2197–2219, 2004. ISSN 0899-7667. URL https://doi.org/10.1162/0899766041732396. [p345, 347]

L. Chen and A. Buja. Local multidimensional scaling for nonlinear dimension reduction, graph drawing, and proximity analysis. *Journal of the American Statistical Association*, 104(485):209–219, 2009. URL https://doi.org/10.1198/jasa.2009.0111. [p350]

R. R. Coifman and S. Lafon. Diffusion maps. *Applied and Computational Harmonic Analysis*, 21(1):5–30, 2006. ISSN 10635203. URL https://doi.org/10.1016/j.acha.2006.04.006. [p347]

P. Comon. Independent component analysis, A new concept? *Signal Processing*, 36(3):287–314, 1994. ISSN 01651684. URL https://doi.org/10.1016/0165-1684(94)90029-9. [p348]

J. de Leeuw and P. Mair. Multidimensional scaling using majorization: Smacof in r. *Journal of Statistical Software, Articles*, 31(3):1–30, 2009. ISSN 1548-7660. doi: 10.18637/jss.v031.i03. URL https://www.jstatsoft.org/v031/i03. [p347]

V. De Silva and J. B. Tenenbaum. Sparse multidimensional scaling using landmark points. Technical report, Stanford University, 2004. [p346]

S. Díaz, J. Kattge, J. H. C. Cornelissen, I. J. Wright, S. Lavorel, S. Dray, B. Reu, M. Kleyer, C. Wirth, I. Colin Prentice, E. Garnier, G. Bönisch, M. Westoby, H. Poorter, P. B. Reich, A. T. Moles, J. Dickie, A. N. Gillison, A. E. Zanne, J. Chave, S. Joseph Wright, S. N. Sheremet'ev, H. Jactel, C. Baraloto, B. Cerabolini, S. Pierce, B. Shipley, D. Kirkup, F. Casanoves, J. S. Joswig, A. Günther, V. Falczuk, N. Rüger, M. D. Mahecha, and L. D. Gorné. The global spectrum of plant form and function. *Nature*, 529(7585):167–171, 2016. ISSN 0028-0836. URL https://doi.org/10.1038/nature16489. [p342]

Elsevier. Scopus - Advanced search, 2017. URL https://www.scopus.com/. [p343]

T. M. J. Fruchterman and E. M. Reingold. Graph drawing by force-directed placement. *Softw: Pract. Exper.*, 21(11):1129–1164, 1991. ISSN 1097-024X. URL https://doi.org/10.1002/spe.4380211102. [p348]

Y. Han, J. Kim, and K. Lee. Deep Convolutional Neural Networks for Predominant Instrument Recognition in Polyphonic Music. *IEEE-ACM TRANSACTIONS ON AUDIO SPEECH AND LANGUAGE PROCESSING*, 25(1):208–221, 2017. ISSN 2329-9290. [p342]

G. E. Hinton and S. T. Roweis. Stochastic Neighbor Embedding. In S. Becker, S. Thrun, and K. Obermayer, editors, *Advances in Neural Information Processing Systems 15*, pages 857–864. MIT Press, 2003. URL http://papers.nips.cc/paper/2276-stochastic-neighbor-embedding.pdf. [p343, 348]

W. W. Hsieh. Nonlinear multivariate and time series analysis by neural network methods. *Rev. Geophys.*, 42(1):RG1003, 2004. ISSN 1944-9208. URL https://doi.org/10.1029/2002rg000112. [p342]

A. Hyvarinen. Fast and robust fixed-point algorithms for independent component analysis. *IEEE Transactions on Neural Networks*, 10(3):626–634, 1999. ISSN 1045-9227. URL https://doi.org/10.1109/72.761722. [p348]

T. Kamada and S. Kawai. An algorithm for drawing general undirected graphs. *Information Processing Letters*, 31(1):7–15, 1989. ISSN 0020-0190. URL https://doi.org/10.1016/0020-0190(89)90102-6. [p348]

J. B. Kruskal. Multidimensional scaling by optimizing goodness of fit to a nonmetric hypothesis. *Psychometrika*, 29(1):1–27, 1964a. ISSN 0033-3123, 1860-0980. URL https://doi.org/10.1007/bf02289565. [p347]

J. B. Kruskal. Nonmetric multidimensional scaling: A numerical method. *Psychometrika*, 29(2):115–129, 1964b. ISSN 0033-3123, 1860-0980. URL https://doi.org/10.1007/bf02289694. [p347]

V. Laparra, J. Malo, and G. Camps-Valls. Dimensionality Reduction via Regression in Hyperspectral Imagery. *IEEE Journal of Selected Topics in Signal Processing*, 9(6):1026–1036, 2015. ISSN 1932-4553. URL https://doi.org/10.1109/jstsp.2015.2417833. [p342, 349]

J. A. Lee and M. Verleysen. Quality assessment of dimensionality reduction: Rank-based criteria. *Neurocomputing*, 72(7–9):1431–1443, 2009. ISSN 0925-2312. URL https://doi.org/10.1016/j.neucom.2008.12.017. [p349, 350]

J. A. Lee, E. Renard, G. Bernard, P. Dupont, and M. Verleysen. Type 1 and 2 mixtures of Kullback–Leibler divergences as cost functions in dimensionality reduction based on similarity preservation. *Neurocomputing*, 112:92–108, 2013. ISSN 0925-2312. URL https://doi.org/10.1016/j.neucom.2012.12.036. [p348, 350]

W. Lueks, B. Mokbel, M. Biehl, and B. Hammer. How to Evaluate Dimensionality Reduction? - Improving the Co-ranking Matrix. *arXiv:1110.3917 [cs]*, 2011. URL http://arxiv.org/abs/1110.3917. arXiv: 1110.3917. [p350]

U. v. Luxburg. A tutorial on spectral clustering. *Statistics and Computing*, 17(4):395–416, Dec. 2007. ISSN 0960-3174, 1573-1375. URL https://doi.org/10.1007/s11222-007-9033-z. [p347]

M. D. Mahecha, A. Martínez, G. Lischeid, and E. Beck. Nonlinear dimensionality reduction: Alternative ordination approaches for extracting and visualizing biodiversity patterns in tropical montane forest vegetation data. *Ecological Informatics*, 2(2):138–149, 2007. ISSN 1574-9541. URL https://doi.org/10.1016/j.ecoinf.2007.05.002. [p347, 351]

S. Martin, W. M. Brown, and B. N. Wylie. Dr.l: Distributed Recursive (graph) Layout. Technical Report dRl; 002182MLTPL00, Sandia National Laboratories, 2007. URL http://www.osti.gov/scitech/biblio/1231060-dr-distributed-recursive-graph-layout. [p348]

R. A. A. Morrall. Soil microfungi associated with aspen in Saskatchewan: Synecology and quantitative analysis. *Can. J. Bot.*, 52(8):1803–1817, 1974. ISSN 0008-4026. URL https://doi.org/10.1139/b74-233. [p342]

K. Pearson. On lines and planes of closest fit to systems of points in space. *Philosophical Magazine*, 2(6):559–572, 1901. URL https://doi.org/10.1080/14786440109462720. [p345]

F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011. [p343]

S. T. Roweis and L. K. Saul. Nonlinear Dimensionality Reduction by Locally Linear Embedding. *Science*, 290(5500):2323–2326, 2000. ISSN 0036-8075, 1095-9203. URL https://doi.org/10.1126/science.290.5500.2323. [p346]

C. Saunders, A. Gammerman, and V. Vovk. Ridge regression learning algorithm in dual variables. In *Proceedings of the Fifteenth International Conference on Machine Learning*, ICML '98, pages 515–521, San Francisco, CA, USA, 1998. Morgan Kaufmann Publishers Inc. ISBN 1-55860-556-8. URL http://dl.acm.org/citation.cfm?id=645527.657464. [p349]

B. Schölkopf, A. Smola, and K.-R. Müller. Nonlinear Component Analysis as a Kernel Eigenvalue Problem. *Neural Computation*, 10(5):1299–1319, 1998. ISSN 08997667. URL https://doi.org/10.1162/089976698300017467. [p345]

B. Schölkopf, R. Herbrich, and A. J. Smola. A Generalized Representer Theorem. In *Computational Learning Theory*, pages 416–426. Springer-Verlag, 2001. URL https://doi.org/10.1007/3-540-44581-1_27. [p345]

R. R. Sokal and F. J. Rohlf. The Comparison of Dendrograms by Objective Methods. *Taxon*, 11(2):33–40, 1962. ISSN 0040-0262. URL https://doi.org/10.2307/1217208. [p350]

S. Sonnenburg, H. Strathmann, S. Lisitsyn, V. Gal, F. J. I. García, W. Lin, S. De, C. Zhang, frx, tklein23, E. Andreev, JonasBehr, sploving, P. Mazumdar, C. Widmer, P. D. . Zora, G. D. Toni, S. Mahindre, A. Kislay, K. Hughes, R. Votyakov, khalednasr, S. Sharma, A. Novik, A. Panda, E. Anagnostopoulos, L. Pang, A. Binder, serialhex, and B. Esser. Shogun-toolbox/shogun: Shogun 6.1.0, 2017. URL https://doi.org/10.5281/zenodo.1067840. [p343]

J. B. Tenenbaum, V. de Silva, and J. C. Langford. A Global Geometric Framework for Nonlinear Dimensionality Reduction. *Science*, 290(5500):2319–2323, 2000. ISSN 0036-8075, 1095-9203. URL https://doi.org/10.1126/science.290.5500.2319. [p346, 351]

W. S. Torgerson. Multidimensional scaling: I. Theory and method. *Psychometrika*, 17(4):401–419, 1952. ISSN 0033-3123, 1860-0980. URL https://doi.org/10.1007/bf02288916. [p346]

L. van der Maaten and G. Hinton. Visualizing Data using t-SNE. *J. Mach. Learn. Res.*, 9:2579–2605, 2008. ISSN 1532-4435. WOS:000262637600007. [p343, 348]

L. Van Der Maaten, E. Postma, and J. Van den Herik. Dimensionality reduction: a comparative review. *J Mach Learn Res*, 10:66–71, 2009. [p343, 344]

J. Venna, J. Peltonen, K. Nybo, H. Aidos, and S. Kaski. Information Retrieval Perspective to Nonlinear Dimensionality Reduction for Data Visualization. *J. Mach. Learn. Res.*, 11:451–490, 2010. ISSN 1532-4435. WOS:000277186500001. [p348]

Guido Kraemer
Max Planck Institute for Biogeochemisty
Hans-Knöll-Str. 10, 07745 Jena
Jena
gkraemer@bgc-jena.mpg.de

Markus Reichstein
Max Planck Institute for Biogeochemisty
Hans-Knöll-Str. 10, 07745 Jena
Jena
mreichstein@bgc-jena.mpg.de

Miguel D. Mahecha
Max Planck Institute for Biogeochemisty
Hans-Knöll-Str. 10, 07745 Jena
Jena
mmahecha@bgc-jena.mpg.de

GrpString: An R Package for Analysis of Groups of Strings

by Hui Tang, Elizabeth L. Day, Molly B. Atkinson, and Norbert J. Pienta

Abstract The R package **GrpString** was developed as a comprehensive toolkit for quantitatively analyzing and comparing groups of strings. It offers functions for researchers and data analysts to prepare strings from event sequences, extract common patterns from strings, and compare patterns between string vectors. The package also finds transition matrices and complexity of strings, determines clusters in a string vector, and examines the statistical difference between two groups of strings.

Introduction

In domains such as psychology and social science, participants' actions can be listed as sequences of events. These event sequences can be recorded as strings in which a single character represents an event or a state. For example, in an eye-tracking study, the eye gazes on some regions in the order of "Question-Figure-Answer1-Answer2-Answer3-Answer4-Figure-Answer3" can be recorded as a string "QF1234F3" (also called a scanpath in eye-tracking studies). Another example is using the log files of an online learning system to generate event sequences from student actions. Processing and analyzing strings helps researchers explore the features of event sequences. In R, a string is a sequence of characters. It is generated using single or double quotes, and can contain zero or more characters. The R **base** package offers various functions for string operations including character count, case conversion, pattern detection, substring extraction and replacement, and string split and concatenation. These methods are also provided by packages **stringr** (Wickham, 2010, 2017), **stringb** (Meissner, 2016), and **stringi** (Gagolewski and Tartanus, 2017), which are built for the purpose of string manipulation. In addition, other packages, such as **gsubfn** (Grothendieck, 2014) and **uniqtag** (Jackman, 2015), focus on particular functionalities of handling strings. Despite a number of packages for string manipulation, there are few packages designed for the analysis of strings, especially groups of character strings. A common method for string analysis in the existing packages is string comparison by using distances between two strings. For example, the function `adist()` in the R **utils** package calculates a generalized Levenshtein distance between two strings, and the function `stringdist()` in package **stringdist** (van der Loo, 2014, 2016) provides options of computing ten different types of distances.

Broadly speaking, the text of natural language and DNA sequences are also strings. However, these two types of strings differ from simple character strings, which are continuous and generally short (at most several hundred characters in a string). Compared to simple character strings, DNA sequences usually are much longer while natural language text may be segmented by spaces or punctuation. There has been a relatively large collection of packages for processing and analyzing these two complex types of sequences (`https://cran.r-project.org/web/views/NaturalLanguageProcessing.html`; `https://www.bioconductor.org/`). Other packages for manipulating and analyzing event or text sequences include **TraMineR** (Gabadinho et al., 2011, 2017) and **informR** (Marcum and Butts, 2015). On the other hand, there is no package built specifically for analyzing groups of typical character strings from the quantitative perspective. Therefore, a comprehensive R package will fill this gap by providing functions to deal with one or multiple groups of simple character strings, including quantitatively describing and statistically examining differences between groups of strings ("string groups" or "string vectors" are also used in this paper).

Package **GrpString** (Tang and Pienta, 2017) is developed with this purpose, and it emphasizes quantifying the features of a string group as well as the differences between two string groups. First, the package provides functions for the users to convert raw event sequences to strings and then to "collapsed" strings. Next, there are functions that extract common patterns shared by the strings in a vector. The functions return the frequency of each pattern, as well as the number and starting position of each pattern in each string. When patterns from two string vectors are compared, featured patterns for each vector are listed to distinguish the two vectors of strings. The package also contains functions for finding the transition matrices of a single string or a group of strings and for calculating the complexity of strings based on transitions. A transition is a two-character sub-sequence within a string or group of strings. A transition matrix in this package is a two-dimensional matrix that lists the numbers of transitions. Cluster analysis is also included with both hierarchical and k-means methods. Lastly, users can employ a permutation test to statistically compare and visualize the difference between two string vectors. Figure 1 shows the main functions in the current version (0.3.2) with a detailed description for each function.

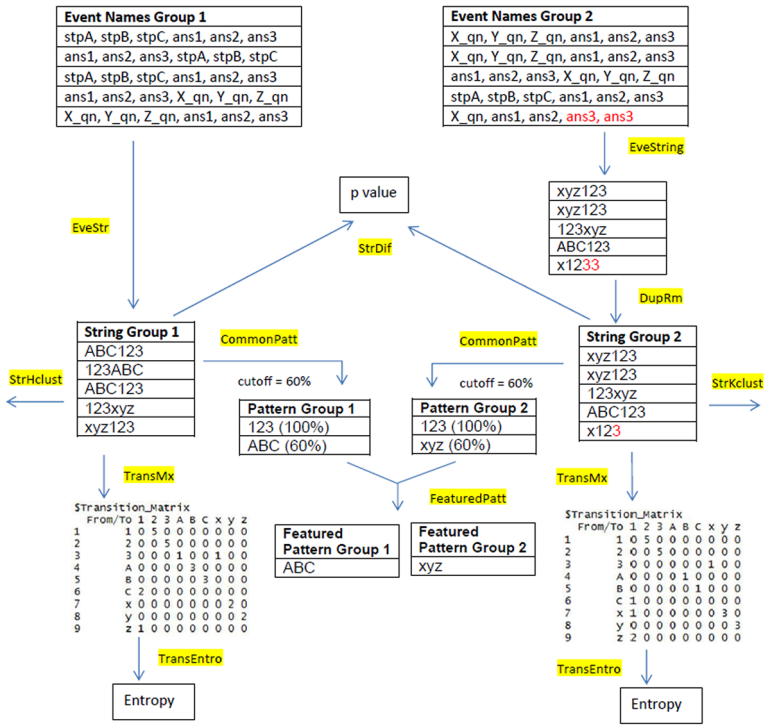


Figure 1: Flow chart of the main functions in the package **GrpString**. There are alternatives for some functions in the flow chart. For example, `EveStr()` can be replaced with `EveString()`, `CommonPatt()` can be replaced with `CommonPattern()`, and `StrHclust()` and `StrKclust()` are interchangeable. The cutoff 60% is chosen as an example showing how to obtain common patterns and then featured patterns. In practice, the users may select different cutoffs when using `CommonPatt()` or `CommonPattern()`.

Functions in GrpString

Converting event sequences to strings

Encoding each event name to its corresponding character is often a necessary step before string analysis. Automatic and simultaneous conversion of event names to string vectors can reduce the users' workload, especially when multiple sequences of event names need to be converted repeatedly. The **GrpString** package offers three functions to accomplish this task, based on a conversion key created by the user. The first function, `EveS()`, converts a sequence of event names to a single string. Here, the two input vectors, `eve.names` and `labels`, which must have the same length, form a conversion key. An element in `labels` can be a letter, digit, or a special character.

```
> event.vec <- c("aoi_1", "aoi_2", "aoi_3", "aoi_2", "aoi_1")
> eve.names <- c("aoi_1", "aoi_2", "aoi_3")
> labels <- c("a", "b", "c")
> EveS(event.vec, eve.names, labels)

[1] "abcba"
```

The second function, `EveStr()`, is applicable when event names are stored in a data frame in which each row has the same number of event names, and hence each converted string has the same length in the exported vector. In this example, `event.name.vec` and `label.vec` form a conversion key.

```
> event.df <- data.frame(c("aoi_1", "aoi_2"),
                         c("aoi_1", "aoi_3"),
                         c("aoi_3", "aoi_5"))
> event.name.vec <- c("aoi_1", "aoi_2", "aoi_3", "aoi_4", "aoi_5")
> label.vec <- c("a", "b", "c", "d", "e")
> EveStr(event.df, event.name.vec, label.vec)
```

```
[1] "aac" "bce"
```

The third function, `EveString()`, is a generalized version of `EveStr()`. It deals with the situation when the user stores event names in a file (e.g., '.txt' or '.csv') in which different rows may have different numbers of elements. It is generally not convenient to read such a file into a data frame. Thus, a '.txt' or '.csv' file for event names is used directly in the function to save the user's effort. The following command converts an array of event names to a vector containing 45 strings with different lengths. The conversion key is stored in data frame `eventChar.df`, in which the first column contains event names and the second column contains the characters to be converted. The object event1d holds the directory of file 'eve1d.txt' (located in the user's R library in this example), which has 45 rows, each with different numbers of event names. Note that it is common to use a file name (and its file path if the file is not in the current directory) directly instead of an object (like `event1d`) in the function, and the users should not forget the quote sign.

```
> data(eventChar.df)
> event1d <- paste(path.package("GrpString"), "/extdata/eve1d.txt", sep = "")
> EveString(event1d, eventChar.df$event, eventChar.df$char)

[1] "D02F0E20DEDC0C30BDC0E45G050A0B5050A06BG0BA5607BA"
[2] "A1ABC21EF0230E03G032C30CBABGBA5G5A7G"
[3] "B1G1GEG10CEF2BAC3DEBA404B5G6F6A"
. . .
[45] "DCB010A0Q1EF21F0FGF0G0GF0B0BC20D0D0303DF04030CF45050B0CBCB05607B0"
```

After the event-string conversion, a string may contain consecutive repeating characters, but sometimes the users are only interested in "collapsed" strings in which successively duplicated characters are removed. Function `DupRm()` allows users to obtain collapsed strings.

```
> dup1 <- "000<<<<<DDDFFF333333qqqqqKKKKK33FFF"
> dup3 <- "aaBB111^^~~~555667777!!!###$$$$$$&&&(((((***)))))@@@@@>>>>99"
> dup13 <- c(dup1, dup3)
> DupRm(dup13)

[1] "0<DF3qK3F"           "aB1^~5670!#$&(*)@>9"
```

Detecting patterns

In **GrpString**, a pattern (also called a common pattern) is defined as a substring with a minimum length of three that occurs at least twice among a group of strings. For a single string of length m, the total number of substrings of length 3 or more is n = (m - 1) × (m - 2) / 2. Note that a string itself is also considered as a substring. By using exhaustive search of substrings in the string vector, this package provides two functions to detect and organize patterns. The simplified version, `CommonPatt()`, returns one data frame. Because there could be thousands of substrings in a string vector, the function utilizes a cutoff to display more frequent or important patterns. The cutoff is the minimum percentage of the occurrence of patterns or substrings and is selected by the user. If the number of strings in a group is num, and the cutoff is selected as c, then only patterns with the minimum number of occurrence $f_{min} = num \times c\%$ will be returned by function `CommonPatt()`. In the following example, $num = 6$, $c = 30$. Thus, $f_{min} = 6 \times 30\% = 1.8$. Patterns with frequency (i.e., number of occurrence) ≥ 2 are displayed in column `Freq_grp`.

```
> strs.vec <- c("ABCDdefABCDa", "def123DC", "123aABCD", "ACD13", "AC1ABC",
                "3123fe")
> CommonPatt(strs.vec, low = 30)

   Pattern Freq_grp Percent_grp Length Freq_str Percent_str
      ABCD        3      50.00%      4        2      33.33%
       ABC        4      66.67%      3        3      50.00%
       123        3      50.00%      3        3      50.00%
       BCD        3      50.00%      3        2      33.33%
       def        2      33.33%      3        2      33.33%
```

The exported data frame is sorted by the pattern length and then by the frequency or percentage of the pattern. Note that the result contains two sets of pattern frequencies and percentages; one is of the overall occurrence (`Freq_grp` and `Percent_grp`) in a string group and the other excludes duplicated occurrences in the same string (`Freq_str` and `Percent_str`). As a consequence, in some cases Percent_grp can be larger than 100%, while Percent_str will not exceed 100%. The full version of

this pair, `CommonPattern()`, offers more options. In addition to the lowest minimum cutoff as described in function `CommonPatt()`, the user can select the highest minimum cutoff and the interval between the two cutoffs. Furthermore, the user can choose using a conversion key to convert patterns back to sequences of event names, which makes it easier to interpret the patterns. This function exports a set of '.txt' files in the current directory instead of a data frame.This is because the number of files and the numbers of rows in some files could be large, which might be difficult for the user to view the results in R directly. The names of these '.txt' files consist of the name of the input string vector and the percentages resulted from the cutoffs and the interval in the function. Patterns that occur at least twice are exported in a separate '.txt' file with `"_f2up"` appended to the name of the input string vector. For example, the following command exports three files: 'strs.vec_30up.txt', 'strs.vec_50up.txt', and 'strs.vec_f2up.txt'. Each file contains a table that has the same columns as shown in the above result from function `CommonPatt()`.

```
> strs.vec <- c("ABCDdefABCDa", "def123DC", "123aABCD", "ACD13", "AC1ABC",
                "3123fe")
> CommonPattern(strs.vec, low = 30, high = 50, interval = 20)
```

Pattern information and featured patterns

The first function in this pair, `PatternInfo()`, lists some basic information about patterns in each string. The information includes the length of each string and the starting position of each pattern in a string. If a pattern does not appear in a string, `"-1"` will be returned. If a pattern occurs at least twice in a string the default position is for the first occurrence, although there is an option for the users to choose the last occurrence of duplicated patterns in the same string.

```
> strs.vec <- c("ABCDdefABCDa", "def123DC", "123aABCD", "ACD13", "AC1ABC",
                "3123fe")
> patts <- c("ABC", "123")
> PatternInfo(patts, strs.vec)

             length ABC 123
ABCDdefABCDa     12   1  -1
def123DC          8  -1   4
123aABCD          8   5   1
ACD13             5  -1  -1
AC1ABC            6   4  -1
3123fe            6  -1   2
```

The second function in this pair, `FeaturedPatt()`, is developed to distinguish the pattern characteristics of two string groups. It compares two groups of strings and discovers featured patterns in each of the groups. It also lists basic information about the featured patterns in each string. In this function, "featured" means patterns in a resulting pattern group can only exist in one of the two input-pattern groups. Note that "featured patterns" shared by strings in both string groups are allowed. This is because, in practice, it is difficult to find a pattern that is exclusively present in all the strings within only one group. As a result, in this function, "featured patterns" are probably obtained from two pattern vectors, each of which contains patterns that are shared by a certain percentage of strings in a group. The following simple example uses two groups (or vectors) of strings, `s_grp1` and `s_grp2`, which are very similar to those in Figure 1. The two pattern vectors, `p1` and `p2`, are obtained from `s_grp1` and `s_grp2`, respectively by applying function `CommonPatt()` with cutoff = 60%. Thus, `p1` contains patterns that are shared by at least 60% of the strings in `s_grp1` and `p2` contains patterns that are shared by at least 60% of the strings in `s_grp2`. To apply `FeaturedPatt()`, the first two arguments are `p1` and `p2`, which serve as the "input-pattern groups", and the last two arguments are the original string groups `s_grp1` and `s_grp2`. The function exports five text files: 'uni_p1-p2.txt', 'p1-vs-p2_in_s_grp1.txt', 'p1-vs-p2_in_s_grp2.txt', 'p2-vs-p1_in_s_grp1.txt', and 'p2-vs-p1_in_s_grp2.txt'. Figure 2 shows the content of the first three files. The resulting featured patterns are listed in File 1 ('uni_p1_p2.txt'). It can be seen that "ABC" (featured pattern group 1) does not appear in `p2` (input-pattern group 2), but can be found in `s_grp2` (20% of string group 2); "xyz" (featured pattern group 2) does not appear in `p1` (input-pattern group 1), but can be found in `s_grp1` (40% of string group 1). In File 2 ('p1-vs-p2_in_s_grp1.txt') and File 3 ('p1-vs-p2_in_s_grp2.txt'), the four columns are (original) string group, length of string, number of featured patterns in string, and the starting position of featured pattern in string. If `p1` had n patterns instead of one, the number of columns in File 2 and File 3 should have $3 + n$ columns; starting from the 4th column, each column lists the starting position of a featured pattern.

```
> s_grp1 <- c("ABC123", "123ABC", "ABCx123", "123xyz", "xyz123")
> s_grp2 <- c("xyz123", "xyzA123", "123xyz", "ABC123", "x123")
```

```
#File 1. 'uni_p1_p2.txt'
  onlyIn_s_grp1 onlyIn_s_grp2
1         ABC         xyz

#File 2. 'p1-vs-p2_in_s_grp1.txt'
   s_grp1 Length numPattern ABC
1  ABC123     6          1   1
2  123ABC     6          1   4
3 ABCx123     7          1   1
4  123xyz     6          0  -1
5  xyz123     6          0  -1

#File 3. 'p1-vs-p2_in_s_grp2.txt'
   s_grp2 Length numPattern ABC
1  xyz123     6          0  -1
2 xyzA123     7          0  -1
3  123xyz     6          0  -1
4  ABC123     6          1   1
5    x123     4          0  -1
```

Figure 2: Three of the five files exported using function `featuredPatt()`. File 1. 'uni_p1_p2.txt': featured patterns in the original string groups. File 2. 'p1-vs-p2_in_s_grp1.txt': information of featured pattern "ABC" (which is from original pattern group p1) in original string group s_grp1. File 3. 'p1-vs-p2_in_s_grp2.txt': information of featured pattern "ABC" (which is from original pattern group p1) in original string group s_grp2.

```
> p1 <- c("123", "ABC")
> p2 <- c("123", "xyz")

> FeaturedPatt(p1, p2, s_grp1, s_grp2)
```

Ideally, a featured pattern presents in 100% of the strings of one string group but in none of the strings of the other group. However, obtaining featured patterns based on cutoffs smaller than 100% as described above still has significance. If two groups of strings are different, then patterns in featured pattern vector 1 are likely to present more frequently in string group 1 than in string group 2. For example, although the featured pattern "ABC" exists in both string groups, it can be found in 60% of the strings in String Group 1, but in only 20% in string group 2 (Figure 1 and Figure 2). Note that Figure 1 and Figure 2 only show a simplified example. In reality, each string may contain multiple featured patterns. Therefore, featured patterns, including their numbers and positions in a string (Figure 2), in turn can be used to categorize the string, i.e., to classify or predict to which group the string belongs.

Transition matrix and information

In addition to patterns, transitions are also often reported in string analysis. For example, researchers in eye-tracking studies may be interested in the gaze transitions within a specific location or between two different locations. A transition is a substring with length of 2. For a single string of length m, the total number of transitions n = m - 1. According to the definition of transition, transitions are not applied to strings with length < 2. The first function in this pair, `TransInfo()`, returns the numbers of two types of transitions — letter and digit by default — in a group of strings. Letters and digits are used as default because they are common components in strings and can represent two different types of events. The function also reports the number of transitions that do not belong to either of the two types. To be more flexible, the users can define any two types of transitions (see the following example).

```
> strs.vec <- c("ABCDdefABCDa", "def123DC", "123aABCD", "ACD13", "AC1ABC",
                "3123fe")
> TransInfo(strs.vec)

  transition_name transition_number
1           type1                24
```

```
2          type2                8
3          mixed                7
```

The second function in this pair, `TransMx()`, returns grand transition matrices in a group of strings. The first matrix contains the numbers of transitions between two characters in a string vector. Normalized transition numbers are stored in the second matrix. A data frame that contains transitions sorted by frequency is also returned. Moreover, this function provides an option to export a transition matrix for each individual string into the current directory. The following shows the usage and results of `TransMx()`. The results are stored in a list. Only one of the components in the list, `Transition_Matrix`, is shown. The matrix of normalized transitions `Transition_Normalized_Matrix` and the data frame of the sorted transitions `Transition_Organized` are not shown. As an example, the readers can understand transition matrix $Transition_Matrix by looking at the transition "12". In vector `strs.vec`, the transition "12" occurs three times (in the 2nd, 3rd and 6th strings). Thus, the value is 3 in the cell of row 1, column 2 of `Transition_Matrix`, which is the total number of the transition "From 1 To 2".

```
> strs.vec <- c("ABCDdefABCDa", "def123DC", "123aABCD", "ACD13", "AC1ABC",
               "3123fe")
> TransMx(strs.vec)

$Transition_Matrix
   From/To 1 2 3 a A B C d D e f
1        1 0 3 1 0 1 0 0 0 0 0 0
2        2 0 0 3 0 0 0 0 0 0 0 0
3        3 1 0 0 1 0 0 0 0 1 0 1
4        a 0 0 0 0 1 0 0 0 0 0 0
5        A 0 0 0 0 0 4 2 0 0 0 0
6        B 0 0 0 0 0 0 4 0 0 0 0
7        C 1 0 0 0 0 0 0 0 4 0 0
8        d 0 0 0 0 0 0 0 0 0 2 0
9        D 1 0 0 1 0 0 1 1 0 0 0
10       e 0 0 0 0 0 0 0 0 0 0 2
11       f 1 0 0 0 1 0 0 0 0 1 0
```

Transition entropy

Transition entropy measures the diversity of the transitions in a string or a group of strings. It can be used as an estimate of string complexity. Larger entropy reflects more evenly distributed transitions and smaller entropy reflects more biased distribution of transitions. Entropy in the current version of package **GrpString** is calculated using the Shannon entropy formula (Shannon, 1948):

$$H = -\sum_{i=1}^{n} freqs_i \times \log_2 (freqs_i) \tag{1}$$

Here, $freqs_i$ is the ith frequency of a transition in a string or a string vector. These are normalized transition numbers, which can be obtained in the normalized transition matrix exported by function `TransMx()` in this package. The formula is equivalent to the function `entropy.empirical()` in the **entropy** package (Hausser and Strimmer, 2014) when setting $unit = "\log_2"$. Strings with length < 2 are not counted for calculating entropies because transitions are not applied to those strings.

One function in this pair, `TransEntro()`, computes the overall transition entropy for a group of strings. The other function, `TransEntropy()`, returns the transition entropy for each of the strings in a group. Note that the third string (`"A"`) is skipped in the result because the length of this string is 1. This function provides another way to quantitatively and/or statistically compare two groups of strings. For example, with the entropy values obtained from `TransEntropy()`, the users can conduct a t-test to examine whether the difference in complexity of two string groups is statistically significant.

```
> stra.vec <- c("ABCDdefABCDa", "def123DC", "A", "123aABCD", "ACD13", "AC1ABC",
               "3123fe")
> TransEntro(stra.vec)

[1] 4.272331

> TransEntropy(stra.vec)
```

```
  String  Entropy
1      1 2.913977
2      2 2.807355
3      4 2.807355
4      5 2.000000
5      6 2.321928
6      7 2.321928
```

Statistical difference between two groups of strings and distribution of differences

`StrDif()` employs a permutation test to statistically compare the difference between two groups of strings based on normalized Levenshtein distances (LDs). `StrDif()` is the main function, and `HistDif()` is the auxiliary function that optimizes the histogram generated by the former. A Levenshtein distance between two strings is the minimum number of operations to transform one string into the other by inserting, deleting, or replacing characters. LDs in `StrDif()` are calculated directly using R's `adist()` function in the package **utils**. A normalized LD between two strings is the LD value divided by the length of the longer string. For two groups of strings, there are two types of LDs. One is within-group LD that is computed by comparing each string with all the others in the same group; the other is between-group LD that is computed by comparing each string in one group with all strings in the other group.

When comparing two groups of strings statistically, the null hypothesis is that the average normalized between-group LD is equal to the average normalized within-group LD. Under this null hypothesis, the difference (d^*) between the two average normalized LDs should be zero. We define $d^* = d_{between} - d_{within}$ as the difference for the original two groups of strings. In a permutation test, strings are re-allocated randomly between the two new groups, each with the same number of strings as the original groups. Every new combination of two new groups results in a new distance $d^{*'} = d'_{between} - d'_{within}$. If one group has n strings and the other group has m strings, the total number of all possible permutations (NP) is

$$NP = \frac{(n+m)!}{n!m!} \tag{2}$$

The p-value is then calculated using equation:

$$p = \frac{\sum_{i=1}^{NP} \left(d_i^{*'} \geq d_i^* \right)}{NP} \tag{3}$$

In practice, NP can be very large, which will make the running time of computing all $d^{*'}$ values very long. Therefore, a smaller subset of all the possible re-allocations of two groups of strings (i.e., permutations) is usually selected using the Monte Carlo method to reduce the burden of computations. `StrDif()` allows the users to choose the number of permutations. The default value is 1000, which has been proved to be a reasonable number (Tang et al., 2012).

`StrDif()` prints $d_{between}$, d_{within}, d^*, and the p-value. It also returns the $d^{*'}$ (including d^*) vector, in which the total number of elements is the sample size in the Monte Carlo simulation. The users should use a vector object to store the $d^{*'}$ values. Furthermore, the function generates a histogram demonstrating the distribution of $d^{*'}$ (Figure 3). The x-axis is the value of $d^{*'}$ and the y-axis is the absolute value of the frequency of $d^{*'}$. In the graph, d^* will be marked as "Observed Difference" with the p-value labeled beside. Because the positions of the legend and text in the histogram may vary due to different distributions and p-values resulted from different string groups, `StrDif()` provides options for the users to define these two positions in order to obtain the graph with optimal appearance. However, since the users usually do not know the ideal positions in advance, they may have to re-run the function at least once to adjust the positions. To avoid rerunning `StrDif()`, which can take a long time, `HistDif()` allows the users to adjust the positions by directly using the vector containing all $d^{*'}$ generated from `StrDif()`. This is another reason we suggest that the users save $d^{*'}$ values in a vector when performing `StrDif()`.

```
> strs1.vec <- c("ABCDdefABCDa", "def123DC", "123aABCD", "ACD13", "AC1ABC", "3123fe")
> strs2.vec <- c("xYZdkfAxDa", "ef1563xy", "BC9Dzy35X", "AkeC1fxz", "65CyAdC",
                 "Dfy3f69k")
> ld.dif.vec <- StrDif(strs1.vec, strs2.vec, num_perm = 500, p.x = 0.025)
```

```
For the initial two groups of strings,
   the average normalized between-group Levenshtein Distance is:  0.85056
   the average normalized within-group Levenshtein Distance is:  0.84306
   the difference in the average normalized Levenshtein Distance between
 between-group and within-group is:  0.00751.
   The p value of the permutation test is:   0.41000
```

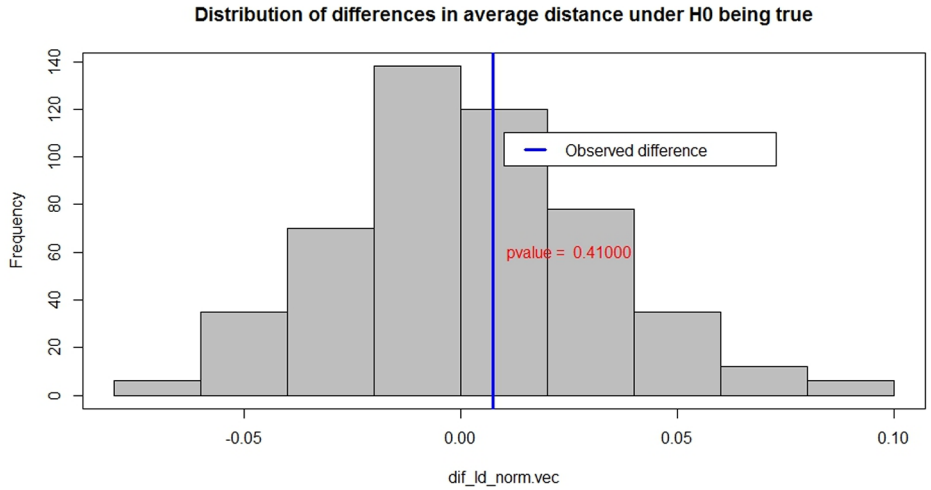


Figure 3: Histogram from function `StrDif()`: distribution of the differences in the average normalized Levenshtein Distance between between-group and within-group strings. The original difference ("Observed difference") d^* is marked in a blue line with the p-value labeled beside. Because a subset of all the possible permutations are selected randomly, the actual p-value may be slightly different each time the user runs the function.

Cluster analysis

The pair of functions, `StrHclust()` and `StrKclust()`, perform string clustering based on Levenshtein distance matrices. Function `StrHclust()` utilizes hierarchical clustering and exports a hierarchical dendrogram (Figure 4), which may suggest a number of clusters. When the user selects a number of clusters, the function assigns each string to a corresponding cluster.

```
> strs3.vec <- c("ABCDdefABCDa", "AC3aABCD", "ACD1AB3", "xYZfgAxZY", "gf56xZYx",
                 "AkfxzYZg")
> StrHclust(strs3.vec)
```

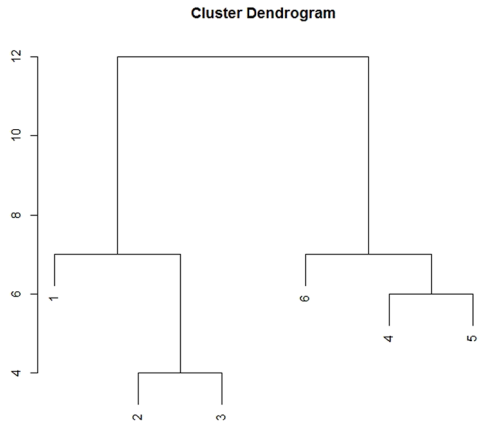


Figure 4: Dendrogram from function `StrHclust()`.

Function `StrKclust()` utilizes k-means clustering and assigns strings to their clusters based on the number of clusters the user chooses. In addition, a cluster plot is produced to visualize the clusters

(Figure 5). In the plot, each labelled point represents a string in the input vector. Points are clustered by ellipses, which are also labelled. Note that by default, the ellipses are not shaded; but the users can shade the ellipses when setting *"shade = TRUE"* in the function.

```
> strs3.vec <- c("ABCDdefABCDa", "AC3aABCD", "ACD1AB3", "xYZfgAxZY", "gf56xZYx",
                 "AkfxzYZg")
> StrKclust(strs3.vec)

  Cluster      Strings
1       1 ABCDdefABCDa
2       1     AC3aABCD
3       1      ACD1AB3
4       2    xYZfgAxZY
5       2     gf56xZYx
6       2     AkfxzYZg
```

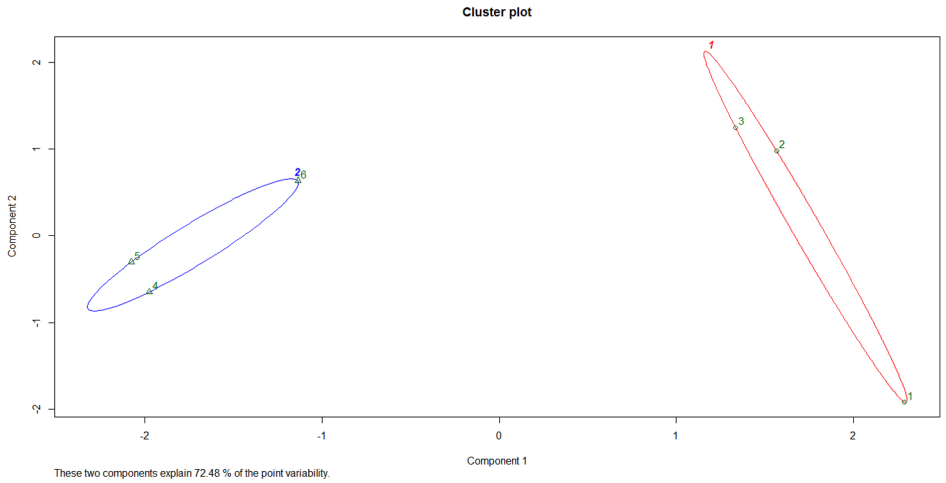


Figure 5: Cluster plot from function `StrKclust()`. All the points (1 to 6) and ellipses (1 and 2) are labelled.

Summary

This article describes the **GrpString** package for analyzing and comparing groups of strings. Most functions in the package **GrpString** were initially developed for analyzing groups of scanpaths (i.e., sequences of eye gazes) in eye-tracking studies. Nevertheless, as described above, this R package can be applied in analysis of any type of character strings, and the strings do not have to be associated with any event or state data. It should be noted that the sole usage of one function may not be sufficient to draw conclusion when answering a research question. An example is `StrDif()`. There could be various factors, such as string lengths, that can affect the p-value of a permutation test for string comparison. Therefore, when the users claim a statistically significant or non-significant difference between two string groups, the results from `CommonPattern()` and/or `TransEntropy()` may be used to support the conclusion. One limitation of the current version of **GrpString** is that it only provides basic and common options in some functions. For instance, only Levenshtein distance is available when distances between strings are computed (`StrDif()`, `StrHclust()` and `StrKclust()`); the updated versions should include an argument for different types of distance. Finally, there are many other advanced analytical methods that have not been included in this package, such as determining the centroid or median string in a string vector (de la Higuera and Casacuberta, 2000; Martínez-Hinarejos et al., 2000) or using Markov chains for string modeling and classification (Krejtz et al., 2014). We plan to build more functions into **GrpString** in the future and welcome feedback from the users to improve this package.

Acknowledgement

This work was supported by a Startup Grant from the Office of Research at the University of Georgia (No. 1026AR168004) and the Journal of Chemical Education Grants. We thank Dr. Rhonda DeCook

in the Department of Statistics and Actuarial Science at the University of Iowa, who wrote the first version of function `StrDif()`.

Bibliography

C. de la Higuera and F. Casacuberta. Topology of strings: Median string is np-complete. *Theoretical computer science*, 230(1-2):39–48, 2000. URL https://doi.org/10.1016/S0304-3975(97)00240-5. [p367]

A. Gabadinho, R. Gilbert, M. Nicolas, and S. Matthias. Analyzing and visualizing state sequences in R with TraMineR. *Journal of Statistical Software*, 40:1–37, 2011. doi: 10.18637/jss.v040.i04. URL https://www.jstatsoft.org/article/view/v040i04. [p359]

A. Gabadinho, R. Gilbert, M. Nicolas, and S. Matthias. *TraMineR: Trajectory Miner: a Toolbox for Exploring and Rendering Sequences*, 2017. URL https://CRAN.R-project.org/package=TraMineR. R package version 2.0.7. [p359]

M. Gagolewski and B. Tartanus. *stringi: Character String Processing Facilities*, 2017. URL https://CRAN.R-project.org/package=stringi. R package version 1.1.5. [p359]

G. Grothendieck. *gsubfn: Utilities for Strings and Function Arguments*, 2014. URL https://CRAN.R-project.org/package=gsubfn. R package version 0.6-6. [p359]

J. Hausser and K. Strimmer. *entropy: Estimation of Entropy, Mutual Information and Related Quantities*, 2014. URL https://CRAN.R-project.org/package=entropy. R package version 1.2.1. [p364]

S. Jackman. *uniqtag: Abbreviate Strings to Short, Unique Identifiers*, 2015. URL https://CRAN.R-project.org/package=uniqtag. R package version 1.0. [p359]

K. Krejtz, T. Szmidt, A. T. Duchowski, and I. Krejtz. Entropy-based statistical analysis of eye movement transitions. In *Proceedings of the Symposium on Eye Tracking Research and Applications*, pages 159–166. ACM, 2014. URL https://doi.org/10.1145/2578153.2578176. [p367]

C. S. Marcum and C. T. Butts. Constructing and modifying sequence statistics for relevent using informR in R. *Journal of Statistical Software*, 64:1–36, 2015. URL https://www.jstatsoft.org/article/view/v064i05. [p359]

C. D. Martínez-Hinarejos, A. Juan, and F. Casacuberta. Use of median string for classification. In *Pattern Recognition, 2000. Proceedings. 15th International Conference on*, volume 2, pages 903–906. IEEE, 2000. URL https://doi.org/10.1109/ICPR.2000.906220. [p367]

P. Meissner. *stringb: Convenient Base R String Handling*, 2016. URL https://CRAN.R-project.org/package=stringb. R package version 0.1.13. [p359]

C. E. Shannon. A mathematical theory of communication. *Bell System Technical Journal*, 27:623–656, 1948. URL http://math.harvard.edu/~ctm/home/text/others/shannon/entropy/entropy.pdf. [p364]

H. Tang and N. J. Pienta. *GrpString: Patterns and Statistical Differences Between Two Groups of Strings*, 2017. URL https://CRAN.R-project.org/package=GrpString. R package version 0.3.2. [p359]

H. Tang, J. J. Topczewski, A. M. Topczewski, and N. J. Pienta. Permutation test for groups of scanpaths using normalized levenshtein distances and application in nmr questions. In *Proceedings of the Symposium on Eye Tracking Research and Applications*, pages 169–172. ACM, 2012. URL https://doi.org/10.1145/2168556.2168584. [p365]

M. van der Loo. The stringdist package for approximate string matching. *The R Journal*, 6:111–122, 2014. URL https://journal.r-project.org/archive/2014-1/loo.pdf. [p359]

M. van der Loo. *stringdist: Approximate String Matching and String Distance Functions*, 2016. URL https://CRAN.R-project.org/package=stringdist. R package version 0.9.4.4. [p359]

H. Wickham. stringr: modern, consistent string processing. *The R Journal*, 2:38–40, 2010. URL https://journal.r-project.org/archive/2010/RJ-2010-012/RJ-2010-012.pdf. [p359]

H. Wickham. *stringr: Simple, Consistent Wrappers for Common String Operations*, 2017. URL https://CRAN.R-project.org/package=stringr. R package version 1.2.0. [p359]

Hui Tang
Department of Chemistry, University of Georgia
140 Cedar Street, Athens, GA 30602-2556
United States
huitang@uga.edu

Elizabeth L. Day
Department of Chemistry, University of Georgia
140 Cedar Street, Athens, GA 30602-2556
United States
elday@uga.edu

Molly B. Atkinson
Department of Chemistry, University of Georgia
140 Cedar Street, Athens, GA 30602-2556
United States
matkin@uga.edu

Norbert J. Pienta
Department of Chemistry, University of Georgia
140 Cedar Street, Athens, GA 30602-2556
United States
npienta@uga.edu

Epistemic Game Theory: Putting Algorithms to Work

by Bilge Başer and Nalan Cinemre

Abstract The aim of this study is to construct an epistemic model in which each rational choice under common belief in rationality is supplemented by a type which expresses such a belief. In practice, the finding of type depends on manual solution approach with some mathematical operations in scope of the theory. This approach becomes less convenient with the growth of the size of the game. To solve this difficulty, a linear programming model is constructed for two-player, static and non-cooperative games to find the type that is supporting that player's rational choice is optimal under common belief in rationality and maximizing the utility of the game. Since the optimal choice would only be made from rational choices, it is first necessary to eliminate all strictly dominated choices. In real life, the games are usually large sized. Therefore, the elimination process should be performed in a computer environment. Since software related to game theory was mostly prepared with a result-oriented approach for some types of games, it was necessary to develop software to execute the iterated elimination method. With this regard, a program has been developed that determines the choices that are strictly dominated by pure and randomized choices in two-player games. Two functions named "esdc" and "type" are created by using R statistical programming language for the operations performed in both parts, and these functions are added to the content of an R package after its creation with the name **EpistemicGameTheory**.

Introduction

In order to evaluate the possible results of the decision, it is very important to constitute a belief about opponents' feasible preferences which can affect their choices. In addition to this, the precondition of making a good choice requires having a reasonable belief about opponents' choices. However, in general each belief of players may not be reasonable according to their opponents. The player should determine the opponent's possible idea about his opponent with putting himself into his opponent's shoes, to decide which choice is reasonable for his opponent or which choice would not be preferred by him. In other words, before surmising an idea about opponents' choices, it is compulsory to reason their system of thought. Indeed, Oskar Morgenstern who is one of the earliest founders of game theory, has highlighted this subject in his article "Perfect Foresight and Economic Equilibrium" which was published in 1935. In his article, Morgenstern has explained the significance of having idea about beliefs of the opponents, analyzing the opponents' systems of thought properly and establishing a reasonable relation to make a good decision (Morgenstern, 1935). However, the importance of this concept frequently has been underestimated in the studies on game theory, which have been published in last sixty years. Morgenstern's bold idea of using the tools of formal logic to talk about how members of a social system think, about how they think about what other members think, and so on, was far ahead of its time. However now, in the form of epistemic game theory, it has found a home (Brandenburger, 2010).

The discipline that studies these patterns of reasoning, and how they influence the eventual choices of the players, is called epistemic game theory (Perea, 2012).[1]

Approximately twenty-five years ago, conceptual changes have emerged with the introduction of epistemic game theory. This new branch of science has brought game theory back to its fundamental concepts, its background. In other words, it has brought game theory back to reasonable modeling of players' beliefs about their opponents. At the core of epistemic game theory, there is the fact that people have different tendencies to reason under same circumstances in a game. Therefore, it is not true reasoning in a unique way and claiming that it is the best option. Under such conditions, it can be said that there are only different reasoning ways, and it should be avoided that claiming on which one is better. In epistemic approach, the aim is to define the methods of reasoning, which can be used in the game, and to examine how the method affects the result of the game.

[1] In this study, (Perea, 2012) is used on a large scale for explaining the concepts of Epistemic Game Theory.

Concepts Of Epistemic Game Theory

Belief about opponents' choices

The belief of a player about choices of his opponents is a probability distribution which is defined over the set $C_{-i} = C_1 \times \ldots \times C_{i-1} \times C_{i+1} \times \ldots \times C_n$ where C_i is the set of player i's choices. The probability value which is assigned by player i for his opponents' each choice combination, is obtained by $b_i(c_1, \ldots, c_{i-1}, c_{i+1}, \ldots, c_n)$.

Let us symbolize the utility function of player i with u_i and the belief about the choices of his opponents with b_i. Accordingly, expected utility of player i from choosing the choice c_i is calculated by the equation below.

$$u_i(c_i, b_i) = \sum_{(c_1,\ldots,c_{i-1},c_{i+1},\ldots,c_n) \in C_{-i}} b_i(c_1, \ldots, c_{i-1}, c_{i+1}, \ldots, c_n) \times u_i(c_1, \ldots, c_{i-1}, c_i, c_{i+1}, \ldots, c_n)$$

Belief hierarchies and type

The concept of belief hierarchies is the basic element of epistemic approach. In an n-player game, the belief hierarchies for player i is as follows:

1. The belief that player i has about his opponents' choice-combinations,
2. The belief that player i has about the beliefs that his opponents have about their opponents' choice-combinations,
3. The belief that player i has about the beliefs that his opponents have about the beliefs that their opponents have about the other players' choice-combinations,

and so on, ad infinitum. The first belief for player i is his first-order belief, the second belief is his second-order belief, and so on.

The belief hierarchies have some disadvantages both theoretically and practically. In theory, it is difficult to make a mathematical description of the hierarchy. In practice, it is often impossible to write and express each stage of the infinite hierarchy (first-order belief, second-order belief, etc.). For this reason, an approach that can express the hierarchy in a shorter and more formal way is needed.

The concept of infinite belief hierarchy has brought to game theory by John Harsanyi. He has worked on incomplete information games, also called Bayesian games, where players have incomplete information about the parameters of the game. It is aimed to model the beliefs of each player about the unknown parameters of the game, each player's beliefs about the other players' beliefs about these parameters, and so on ad infinitum. This may be called the explicit approach and is in fact feasible. However, the explicit approach is mathematically rather cumbersome and hardly manageable. Indeed, this was a major obstacle to the development of the theory of games with incomplete information at its early stages. The breakthrough was provided by John Harsanyi in a seminal work (Harsanyi, 1982) that awarded him the Nobel Prize in 1994 after thirty years. While Harsanyi actually formulated the problem verbally, in an explicit way, he suggested a solution that 'avoided' the difficulty of having to deal with infinite hierarchies of beliefs, by providing a much more workable implicit, encapsulated model (Zamir, 2013).

The concept of a type is the basis of the Harsanyi model. As a concept, type can be considered as a precise definition of players' belief hierarchies about unknown parameters of the game. It is a special representation of the player's beliefs about the actual parameters involved and the answers to the types of other players. This characteristic of the types gives the player the ability to self-reference inevitably that is, the ability to decide the types of other players through their own types in an interactive decision-making environment.

The construction that Harsanyi proposed in the context of a game with incomplete information on the preferences of the players was very simple: For every player, define a set of types, and for every type define a utility function, together with a probabilistic belief about the opponents' types (Harsanyi, 1982).

The content of the type in epistemic game theory differs from that of Harsanyi. While Harsanyi forms belief hierarchies for parameters of the game, epistemic game theory deals with the choices of the players. In the studies (Armbruster and Boge, 1979) and (Boge and Eisele, 1979), belief hierarchies have been used to describe the beliefs of players about their opponents' choices.

Let us denote the belief hierarchy with t_i^c which supports player i's any choice c. In the literature, belief hierarchy is also used as epistemic type (or briefly type). Therefore, t_i represents an epistemic type of the player i.

Epistemic model

Let us consider an n-player game. The epistemic model, firstly, indicates possible types for each player. Let us symbolize the set of all possible types for every player i with T_i. Each type t_i stores information about the beliefs on the choices of player i's opponents and their types. The problem that arises here is how this belief can be expressed mathematically.

As it is known, every player i's belief about choices of his opponents is a probability distribution b_i which has been defined on the set $C_{-i} = C_1 \times \ldots \times C_{i-1} \times C_{i+1} \times \ldots \times C_n$.$t_i$ should have information about not only the belief about choices of his opponents but also information about his opponents' types. Thus, t_i represents the choice-type combinations of player i's opponents.

The set that contains all possible choice-type combinations of any opponent j of player i is $C_j \times T_j$. In parallel with this definition, the set of all choice-type combinations of player i's opponents' becomes $(C_1 \times T_1) \times \ldots \times (C_{i-1} \times T_{i-1}) \times (C_{i+1} \times T_{i+1}) \times \ldots \times (C_n \times T_n)$. This set includes all possible combinations $((c_1, t_1), \ldots, (c_{i-1}, t_{i-1}), (c_{i+1}, t_{i+1}), \ldots, (c_n, t_n))$.

An epistemic model specifies probability distribution $b_i(t_i)$ defined on the set of all possible choice-type combinations $(C_1 \times T_1) \times \ldots \times (C_{i-1} \times T_{i-1}) \times (C_{i+1} \times T_{i+1}) \times \ldots \times (C_n \times T_n)$ for every player i and each type $t_i \in T_i$. The probability distribution of $b_i(t_i)$ represents the belief that type t_i has about the opponents' choices and types.

Consider a type t_i for player i within an epistemic model. The choice c_i is rational for type t_i if it maximizes the expected utility for the belief that t_i holds about the opponents' choice-type combinations.

The entire belief hierarchy can be expressed with an epistemic model. Constructing an epistemic model is easier than establishing a belief hierarchy. This is an important advantage of epistemic model. Another advantage of the epistemic model is that it can be defined by a mathematical expression conveniently.

Deciding Under Common Belief In Rationality

Type t_i is said to believe in the opponents' rationality if for every opponent j, and every choice-type pair $(c_j, t_j) \in C_j \times T_j$ to which t_i assigns positive probability, the choice c_j is rational for type t_j.

In order to define common belief in rationality, firstly, the definition of k-fold belief in rationality is needed and explained as follows.

k-fold belief in rationality

Consider an epistemic model.

1. Type t_i expresses 1-fold belief in rationality if t_i believes in the opponents' rationality.
2. Type t_i expresses 2-fold belief in rationality if t_i only assigns positive probability to the opponents' types that express 1-fold belief in rationality.
3. Type t_i expresses 3-fold belief in rationality if t_i only assigns positive probability to the opponents' types that express 2-fold belief in rationality.

And so on. Thus, k-fold belief in rationality can be recursively defined for every number k.

Rational choice under belief in the opponents' rationality

The choice c_1^* is the optimal choice for player i, if the expected utility of the choice c_i^* is maximum $(u_i(c_i^*, b_i) \geq u_i(c_i, b_i))$.

If the choice c_i^* is optimal with reference to player i's belief about the opponents' behavior patterns, then c_i^* is called as a rational choice.

Player i believes that his opponents are rational if he assigns positive probability for only his opponent's rational choices.

If the choice c_i of player i is optimal for some belief b_i about the choices of his opponents' while believing in the opponents' rationality, then the choice c_i is a rational choice for player i under belief in the opponents' rationality.

The concept of rational choice in game theory sometimes causes confusion. In the context of this study, the word "rational" is used as follows: If a player has built a belief about his opponent's choices, and has made the optimal choice for himself under this belief, he has made a rational choice. However,

this player may have had an unreasonable belief about the opponent, and making rational choice does not guarantee making the reasonable choice. Reasonability carries a subjective meaning and depends on the mindset of the person. Something that is reasonable for one may not be reasonable for another. Therefore, it is impossible to make a single definition of reasonable choice.

The reasonable choice should not only be rational under the belief in the opponents' rationality, but at the same time, it should be optimal according to a reasonable belief about the opponents' choices. What is open to debate is when does a belief about the opponent be reasonable? Epistemic game theory examines the answer to this question.

Rational choice under common belief in rationality

In an epistemic model, if the type t_i expresses k-fold belief in rationality for every k, it can be said that t_i expresses common belief in rationality. If t_i expresses common belief in rationality and the choice c_i is optimal for the type t_i, then c_i is a rational choice under common belief in rationality.

Common belief in rationality does not only mean that you believe that your opponents choose rationally, but you also believe that your opponents believe that their opponents will choose rationally, and that your opponents believe that their opponents believe that the other players will choose rationally, and so on.

Consider an epistemic model, which contains type sets $T_1, \ldots, T_n$. If the player i's choice c_i is not supported to express common belief in rationality by any type in T_i, this does not mean that the choice c_i cannot be chosen under common belief in rationality. There may be another epistemic model that has a type supporting the choice c_i to express common belief in rationality. The purpose of this work is to seek an answer to the question that how this epistemic model can be detected.

Algorithm 1 *(Choices that can be rationally chosen under common belief in rationality)*:

An algorithm has been required to use in order to find the choices that can be rationally chosen under common belief in rationality. *Algorithm 1* is based on the following theorem.

Theorem 1: A choice c_i is irrational if and only if it is strictly dominated by another pure or randomized choice. In other words, a choice c_i is rational if and only if it cannot be strictly dominated by another pure or randomized choice [2] (Pearce, 1984).

By *Theorem 1*, the algorithm is explained as follows.

1) Eliminate all strictly dominated choices in the original game.

2) Eliminate all strictly dominated choices in the reduced game obtained after the step (1).

3) Eliminate all strictly dominated choices in the reduced game obtained after the step (2).

$\vdots$

Repeat this process until no strategy can be eliminated.

This algorithm ends with a finite number of steps and gives a set of choices that are not empty for each player if the game is finite. The order of elimination and speed do not affect the result.

Theorem 2: (Brandenburger and Dekel, 1987) and (Tan and Werlang, 1988) proved that, if the choices can be rationally made under k-fold belief in rationality for each $k \geq 1$, then these choices are also survived $(k+1)$-fold elimination. This can be generalized as; the rational choices under common belief in rationality are the choices that survive the iterated elimination of strictly dominated choices.

Algorithms For Finding Types That Express Common Belief In Rationality For Optimal Choices

In practice, the finding of type depends on non-computer based approach with some mathematical operations in scope of the theory. This approach becomes less convenient with the growth of the size of the game. For this reason, we construct a linear programming model for two-player, static and non-cooperative games to find the type that is supporting that player i's rational choice c_i is optimal under common belief in rationality and maximizing the utility of the game. Since the optimal choice would only be made from rational choices, it is first necessary to eliminate all strictly dominated choices. By the reason of software related to game theory was mostly prepared with a result-oriented approach for some solution methods and some types of games, it was necessary to develop software to execute the iterated elimination method. With this regard, we developed a computer program that determines the choices that are strictly dominated by pure and randomized choices in two-player

[2] A randomized choice means that a player, before making a choice, uses a randomization device and bases the actual choice on the outcome of the randomization device.

games. Başer transformed the operations performed in both parts to software by using R Statistical Programming Language and created a package with the name **EpistemicGameTheory** (Baser, 2017a).

The **EpistemicGameTheory** R package containing functions named `esdc` and `type` for both purposes explained above. The package **roxygen2** was used to prepare the documentation when the R package was created (Wickham and et al, 2015).

`esdc` function

As it is known, since the optimal choice is made only from rational choices, it is first necessary to make iterated elimination of strictly dominated choices. For this purpose, the steps given in *Algorithm 1* for two-player games need to be coded into software. We developed *Algorithm 2* to make the steps of this algorithm suitable for programming architecture.

Let n be the number of choices of the first player; and m be the number of choices of the second player; the utility matrix of the first player would be;

$$A = \begin{bmatrix} a_{11} & a_{12} & \dots & a_{1m} \\ a_{21} & a_{22} & \dots & a_{2m} \\ \vdots & \vdots & \ddots & \vdots \\ a_{n1} & a_{n2} & \dots & a_{nm} \end{bmatrix}$$

the utility matrix of the second player would be;

$$B = \begin{bmatrix} b_{11} & b_{12} & \dots & b_{1m} \\ b_{21} & b_{22} & \dots & b_{2m} \\ \vdots & \vdots & \ddots & \vdots \\ b_{n1} & b_{n2} & \dots & b_{nm} \end{bmatrix}$$

Each entry of matrix A represents a utility level of the first player while each entry of matrix B represents a utility level of the second player. The two subscripts of these entries indicate the strategies chosen by the two players where the first subscript refers to the strategy chosen by the first player and the second subscript refers to the strategy chosen by the second player.

Algorithm 2 *(Creating the Algorithm to be Used in Comparing the Choices of the First Player)*:

1) A combination matrix (C_A) is generated that contains all combinations of $(n, (n-t))$ with $t = 1$ at the initial point. Each row of C_A shows how to compare the choices.

It is aimed to obtain a randomized choice for each row by using its elements and then compare with the utility level of the choice which does not exist in that row. For instance, let c_1^* be the randomized choice that is obtained by randomization of the choices $(c_{11}, c_{12}, \dots, c_{1(n-t)})$. Then the utility level of c_1^* is compared with the utility level of the choice that does not exist in the first row. The randomization process is made with the procedures in the following steps.

$$C_A = \begin{bmatrix} c_{11} & c_{12} & \dots & c_{1(n-t)} \\ c_{21} & c_{22} & \dots & c_{2(n-t)} \\ \vdots & \vdots & \ddots & \vdots \\ c_{n1} & c_{n2} & \dots & c_{n(n-t)} \end{bmatrix}$$

2) An index matrix (N) with rows equal to the number of rows of the matrix C_A is generated. Each row of the index matrix is equal and consists of a number sequence of 1 to n.

$$N = \begin{bmatrix} 1 & 2 & \dots & n \\ 1 & 2 & \dots & n \\ \vdots & \vdots & \ddots & \vdots \\ 1 & 2 & \dots & n \end{bmatrix}$$

3) A comparison is made between the rows of C_A and N matrices. The elements existed in N and not in C_A are assigned to the corresponding row of the difference matrix. The purpose of this step is identifying the choice for each row which is compared with the randomized choice that is obtained in step 1.

$$D = \begin{bmatrix} d_{11} & d_{12} & \dots & d_{1(n-(n-t))} \\ d_{21} & d_{22} & \dots & d_{2(n-(n-t))} \\ \vdots & \vdots & \ddots & \vdots \\ d_{n1} & d_{n2} & \dots & d_{n(n-(n-t))} \end{bmatrix}$$

4) For every row of C_A, a probability vector P is generated from a uniform distribution on the $(n-t-1)-$ dimensional simplex.

$$P = (p_1, p_2, \dots, p_{(n-t)}), \qquad p_1 + p_2 + \dots + p_{(n-t)} = 1$$

According to this probability vector, the first player chooses the first choice with the probability p_1, the second choice with the probability p_2, ..., $(n-t)$th choice with the probability $p_{(n-t)}$.

5) Using the obtained probability values, the expected utility for the first player as a result of choosing the randomized choice is calculated and compared with the utility provided by the choice not existed in the related row of the combination matrix.

If the first player believes that second player will choose his first strategy, the expected utility for the first player would be;

$$p_1 \times a_{11} + p_2 \times a_{21} + \dots + p_{(n-t)} a_{(n-t)1} = E_1,$$

If the first player believes that second player will choose his second strategy, the expected utility for the first player is calculated as;

$$p_1 \times a_{12} + p_2 \times a_{22} + \dots + p_{(n-t)} a_{(n-t)2} = E_2,$$

$$\vdots$$

If the first player believes that second player will choose his last strategy, the expected utility for the first player is as below;

$$p_1 \times a_{1m} + p_2 \times a_{2m} + \dots + p_{(n-t)} a_{(n-t)m} = E_m,$$

If all expected utilities $(E_1, E_2, \dots, E_m)$ obtained are greater than the expected utility of the choice pointed by the element in the corresponding row of matrix D, then the strategy is strictly dominated by that randomized choice. This process is made for every row of C_A.

Two situations can arise here:

Elimination occurs: In this case, the reduced utility matrix is obtained. The new $C(n^*, (n^* - t))$ combination matrix is created (where n^* is the number of rows of the reduced utility matrix) and the above steps are repeated.

Elimination does not occur: In this case, a new probability vector is generated, and the above steps are repeated. Here it is very important to decide the number of iterations that determine how many times the probability vector will be generated and it depends on the dimension of the game. Therefore, the number of iterations must be sufficiently large to be able to determine the strictly dominated choices (iteration $\longrightarrow \infty$). If there is no strictly dominated choice according to the utility matrix, the value of t is increased by "1" to form a new combination matrix $C(n, (n-t))$ and the above steps are repeated.

6) When $(n^* - t) < 1$ the algorithm ends and the last reduced utility matrix is obtained.

The last reduced utility matrix determined by the above algorithm consists of the rational choices of the first player. The same steps are performed for the second player. The specified steps are written in the R statistical programming language, and a function named "esdc" is created. This function gives the reduced game after the iterated elimination of all strictly dominated choices. The properties of the function are shown in Figure 1.

esdc	*Eliminating strictly dominated choices*
Description	
	This function eliminates strictly dominated choices.
Usage	
	`esdc(n,m,A,choices.A,B,choices.B,iteration)`
Arguments	
`n`	an integer representing the number of choices of player 1
`m`	an integer representing the number of choices of player 2
`A`	an nxm matrix representing the payoff matrix of player 1
`choices.A`	a vector of length n representing the names of player 1's choices
`B`	an nxm matrix representing the payoff matrix of player 2
`choices.B`	a vector of length m representing the names of player 2's choices
`iteration`	an integer representing the iteration number of algorithm
Details	
	This function works for the games with two players.
Value	
	This function works for the games with two players.

Figure 1: `esdc` Function

The `esdc` function uses seven arguments. These are; the choice number of the first player (n), the choice number of the second player (m), the utility matrix of the first player (A), the vector consisting of the choice names of the first player (choices.A), the utility matrix of the second player (B), the vector consisting of the choice names of the second player (choices.B), and the number of repetitions of the algorithm (iteration). As a result of the execution of this function, the reduced utility matrices of the players' that are obtained after eliminating strictly dominated choices as output.

`type` function

It is aimed to show that for every player i and every rational choice $c_i (i = 1, 2, \ldots, n)$, there is a type t_i, which supports that c_i is optimal under common belief in rationality.

At the beginning of the game, a type is defined for each element in the player's choice set. However, the "type" function generates types only for rational choices because the player do not choose irrational strategies in practice.

Construction of Epistemic Model:

Let A^* be the first player's reduced utility matrix.

$$A^* = \begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1m} \\ a_{21} & a_{22} & \cdots & a_{2m} \\ \vdots & \vdots & \ddots & \vdots \\ a_{n1} & a_{n2} & \cdots & a_{nm} \end{bmatrix}$$

The first player believes that the second player will likely choose the first choice with the probability q_1, the second choice with the probability $q_2, \ldots$, and the final choice with the probability q_m. In this case, according to the preferences of the second player, the utilities of the first player can be expressed by a linear equation system.

Under this belief, first player will get the utility U_1, if he chooses his first strategy;

$a_{11}q_1 + a_{12}q_2 + \cdots + a_{1m}q_m = U_1$,

he will get the utility U_2, if he chooses his second strategy;

$a_{21}q_1 + a_{22}q_2 + \cdots + a_{2m}q_m = U_2$

$\vdots$

he will get the utility U_i, if he chooses his ith strategy,

$a_{i1}q_1 + a_{i2}q_2 + \cdots + a_{im}q_m = U_i$

$\vdots$

he will get the utility U_n, if he chooses his nth strategy,

$a_{n1}q_1 + a_{n2}q_2 + \cdots + a_{nm}q_m = U_n$

The first player prefers his ith choice if and only if the inequality $U_i \geq U_1, U_2, \ldots, U_{(i-1)}$, $U_{(i+1)}, \ldots, U_n$ is satisfied. Then, when is this inequality satisfied?

Let's express this inequality in pairwise comparison;

$U_i \geq U_1 \Longleftrightarrow U_i - U_1 \geq 0$

$U_i \geq U_2 \Longleftrightarrow U_i - U_2 \geq 0$

$\vdots$

$U_i \geq U_n \Longleftrightarrow U_i - U_n \geq 0$

Then the linear equation system is obtained by substitution of the equals of the utilities explicitly, where q is a uniform distribution on the $(m-1)-$ dimensional simplex.

$(a_{i1} - a_{11})q_1 + (a_{i2} - a_{12})q_2 + \cdots + (a_{im} - a_{1m})q_m = U_i - U_1 \geq 0$

$(a_{i1} - a_{21})q_1 + (a_{i2} - a_{22})q_2 + \cdots + (a_{im} - a_{2m})q_m = U_i - U_2 \geq 0$

$\vdots$

$(a_{i1} - a_{n1})q_1 + (a_{i2} - a_{n2})q_2 + \cdots + (a_{im} - a_{nm})q_m = U_i - U_n \geq 0$

Suppose that the first player's ith strategy is optimal. In this case, all the points located in the convex region consisting of the intersection of the closed half spaces and the hyperplane are those, which make the i th strategy optimal under common belief in rationality.

The mathematical model above allows us to find all types that c_i is optimal under the common belief in rationality for every player i and every rational choice $c_i(i = 1, 2, \ldots, n)$. By solving this model, an infinite number of points are obtained. Instead of dealing with infinite number of points in practice, constructing an epistemic model that there is a type t_i for each player i and for each rational choice $c_i(i = 1, 2, \ldots, n)$ that supports c_i to be optimal under the common belief in rationality that maximizes the utility is making more sense. For this reason, a linear programming model has been established for each choice. Thus, while showing at least, there is one type making the relevant choice optimal under common belief in rationality, and it can also be found the type that maximizing the utility of the player. In this regard, the linear programming model for the ith choice is established as follows.

$$
\begin{aligned}
Z_{max} &= a_{i1}q_1 + a_{i2}q_2 + \cdots + a_{im}q_m \\
(a_{i1} - a_{11})q_1 + (a_{i2} - a_{12})q_2 + \cdots + (a_{im} - a_{1m})q_m &\geq 0 \\
(a_{i1} - a_{21})q_1 + (a_{i2} - a_{22})q_2 + \cdots + (a_{im} - a_{2m})q_m &\geq 0 \\
&\vdots \\
(a_{i1} - a_{n1})q_1 + (a_{i2} - a_{n2})q_2 + \cdots + (a_{im} - a_{nm})q_m &\geq 0 \\
q_1 + q_2 + \cdots + q_m &= 1 \\
q_1, q_2, \ldots, q_m &\geq 0
\end{aligned}
$$

Similar models can be set up for first player's other choices and for the second player as well. A function named "type" has been created in R programming language for solving the linear programming model described above. The features of this function are shown in Figure 2.

The "type" function uses four arguments, named A, B, choices.A and choices.B. "A" is the reduced utility matrix of the first player, and "B" is the reduced utility matrix of the second player. "choices.A" and "choices.B" represent the name of the choices of players as previously defined. This function uses the "lp" function from the **lpSolve** package, which provides the Simplex method for solving linear programming problems (Berkelaar and et al, 2015). As a result of the execution of the "type" function, probabilities for the types are obtained, which ensure that a rational strategy is optimal under common belief in rationality and maximizes the utility of the player.

Since in epistemic game theory, it is important to comprehend why and how a choice is strictly dominated, the "esdc" function becomes crucial. Indeed, it is advisable that to execute the "esdc" function before the types are specified. In other words, the "esdc" function should be integrated into the "type" function as a prepended operation (Baser, 2017b).

type	*Finding types that express common belief in rationality for optimal choices*
Description	This function takes the reduced payoff matrices and finds out the probabilities for the types that expresses common belief in rationality for optimal choices.
Usage	`type(A,B,choices.A,choices.B)`
Arguments	
`A`	an nxm matrix representing the reduced payoff matrix of player 1
`B`	an nxm matrix representing the reduced payoff matrix of player 2
`choices.A`	a vector of length n representing the names of player 1's choices
`choices.B`	a vector of length m representing the names of player 2's choices
Details	This function works for the games with two players. It returns infeasible solution for the irrational choices.
Value	Probabilities of the types that expresses common belief in rationality for optimal choices

Figure 2: `type` Function

Application of the Traveler's Dilemma

We applied the two functions to Basu's well-known The Traveler's Dilemma (Basu, 1994) game to underscore the practical usefulness of this work. Each player has 99 strategies between 2 and 100. The utility matrices for the players are created according to the utility function of the Traveler's Dilemma game.

Applying `esdc` Function

The arguments for `esdc` function are assigned as given below. For this numerical example, the "iteration" argument is taken as 500. The name of choices of both players ("choices.A" and "choices.B") are denominated by the numbers between 2 to 100.

```
n = 99
m = 99
iteration = 500
esdc(n,m,A,choices. A,B,choices. B,iteration)
```

The Reduced Utility Matrices

With the execution of "esdc" function the algorithm eliminates all strictly dominated choices from the utility matrices A and B. For these matrices 196 times elimination occurred. The reduced utility matrices are displayed below. In the last reduced game, the players have only one choice left, which is choosing a price of "2". Therefore, they can only rationally choose a price of "2" under common belief in rationality.

```
[1] "ELIMINATION IS OVER."
[1] "The Last Reduced Matrix For Player 1:"
   2   3
[1,] 2   4
[1] "The Last Reduced Matrix For Player 2:"
   [,1]
[1,]   2
```

Applying `type` Function

The `type` function is executed for finding types for rational choices under common belief in rationality given in the reduced matrices for both players. The definitions of arguments are shown below.

```
A<-matrix(c(2),1,1)
B<-matrix(c(2),1,1)
choices.A = c("2")
choices.B = c("2")
type(A,B,choices. A,choices. B)
```

The Output of `type` **Function**

The output of `type` function includes the coefficients of the linear equation system, the types that supports relevant choice under common belief in rationality and the maximum utility of the player. For this example, the output of `type` function is displayed below.

```
type(A,B,choices.A,choices.B)
[1] "The utility matrix of Player 1:"
2
2 2
Player 1's type for the strategy 2 : 1
Success: the objective function is 2
[1] "The utility matrix of Player 2:"
2
2 2
Player 2's type for the strategy 2 : 1
Success: the objective function is 2
```

The epistemic model of the first player was created as follows using belief probabilities from the output of `type` function that make every rational choice of the first player optimal under common belief in rationality.

Type: $T_1 = \{t_1^2\}$

Belief for the first player:

$$b_1(t_1^2) = (2, t_2^2)$$

The epistemic model is constructed by using the values for the second player from the output of `type` function as follows:

Type: $T_2 = \{t_2^2\}$

Belief for the second player:

$$b_2(t_2^2) = (2, t_1^2)$$

Consequently, the first player will choose his choice "2", if he believes that second player will choose "2" with the probability 1. In a similar way, the second player will choose "2", if he believes that first player will choose "2" with the probability 1 and they will end up getting two units of money each.

Conclusion

Game theory has been investigated with epistemic approaches in recent years. Theorists and practitioners do research for analyzing the logic underlying game theory in a broader and more realistic perspective. Although epistemic game theory has a strong theoretical background, there is a lack of tools to work on large-scale problems in practice. This study based on common belief in rationality which can be considered as the heart of epistemic game theory. In two-player games, we developed a systematic way to find out types that optimize their rational choices under common belief in rationality for every player together with maximizing their utility. Thus, this study brings flexibility to producing solutions of large-scale problems within the scope of epistemic game theory. The algorithms which have been used are transformed into software. R is preferred because of its open-source software feature, and an R package has been created to bring the program into use. It is thought that this study will serve as an example for future work in computational epistemic game theory field and it is planned to adapt this approach to n-person games as well.

Bibliography

W. Armbruster and W. Boge. Bayesian game theory. *Game theory and related topics*, 17:28, 1979. [p371]

B. Baser. *EpistemicGameTheory: Constructing an Epistemic Model for the Games with Two Players.*,

2017a. URL https://CRAN.R-project.org/package=EpistemicGameTheory. R Package version 0.1.2. [p374]

B. Baser. *Epistemik Oyun Teorisi Algoritmalari:"EpistemicGameTheory" R Package. Dissertation*. Mimar Sinan Fine Arts University, 2017b. [p377]

K. Basu. The travelers dilemma: Paradoxes of rationality in game theory. *The American Economic Review*, 84(2):391–395, 1994. [p378]

M. Berkelaar and et al. *lpSolve*, 2015. URL https://CRAN.R-project.org/package=lpSolve. R Package version 5.6.13. [p377]

W. Boge and T. H. Eisele. On solutions of bayesian games. *International Journal of Game Theory*, 8(4): 193–215, 1979. URL http://dx.doi.org/10.1007/BF01766706. [p371]

A. Brandenburger. Origins of epistemic game theory. *Epistemic Logic*, 5:59–69, 2010. [p370]

A. Brandenburger and E. Dekel. Rationalizability and correlated equilibria. *Econometrica*, 55(6): 1391–1402, 1987. URL http://dx.doi.org/10.2307/1913562. [p373]

J. C. Harsanyi. Games with incomplete information played by "bayesian" players part i. papers in game theory, theory and decision library I-III. *Management Science*, 14(28):115–138, 1982. URL https://doi.org/10.1007/978-94-017-2527-9_6. [p371]

O. Morgenstern. Perfect foresight and economic equilibrium. *Selected Economic Writings of Oskar Morgenstern*, pages 169–183, 1935. [p370]

D. Pearce. Rationalizable strategic behavior and the problem of perfection. *Econometrica*, (52):1029–1050, 1984. URL http://dx.doi.org/10.2307/1911197. [p373]

A. Perea. *Epistemic Game Theory: Reasoning and Choice*. Cambridge University Press, Cambridge, 2012. URL https://doi.org/10.1017/CBO9780511844072. [p370]

T. Tan and S. Werlang. The bayesian foundations of solution concepts of games. *Journal of Economic Theory*, (45):370–391, 1988. URL http://dx.doi.org/10.1016/0022-0531(88)90276-1. [p373]

H. Wickham and et al. *Roxygen2*, 2015. URL https://CRAN.R-project.org/package=roxygen2. R Package version 6.0.1. [p374]

S. Zamir. *Bayesian Games: Games with Incomplete Information. In: Meyers R. (eds) Encyclopedia of Complexity and Systems Science*. Springer, New York, 2013. URL https://doi.org/10.1007/978-3-642-27737-5_29-3. [p371]

Bilge Başer
Department of Statistics
Mimar Sinan Fine Arts University
Istanbul, Turkey
bilge.baser@msgsu.edu.tr

Nalan Cinemre
Department of Statistics
Mimar Sinan Fine Arts University
Istanbul, Turkey
nalan.cinemre@msgsu.edu.tr

Residuals and Diagnostics for Binary and Ordinal Regression Models: An Introduction to the sure Package

by Brandon M. Greenwell, Andrew J. McCarthy, Bradley C. Boehmke, and Dungang Liu

Abstract Residual diagnostics is an important topic in the classroom, but it is less often used in practice when the response is binary or ordinal. Part of the reason for this is that generalized models for discrete data, like cumulative link models and logistic regression, do not produce standard residuals that are easily interpreted as those in ordinary linear regression. In this paper, we introduce the R package **sure**, which implements a recently developed idea of **SU**rrogate **RE**siduals. We demonstrate the utility of the package in detection of cumulative link model misspecification with respect to mean structures, link functions, heteroscedasticity, proportionality, and interaction effects.

Introduction

Categorical outcomes are encountered frequently in practice across different fields. For example, in medical studies, the outcome of interest is often binary (e.g., presence or absence of a particular disease after applying a treatment). In other studies, the outcome may be an ordinal variable; that is, a categorical outcome having a natural ordering. For instance, in an opinion poll, the response may be satisfaction with categories low, medium, and high. In this case, the response is ordered: low $<$ medium $<$ high.

Logistic and probit regression are popular choices for modeling a binary outcome. Although this paper focuses on models for ordinal responses, the surrogate approach to constructing residuals actually applies to a wide class of general models of the form

$$\mathcal{Y} \sim F_a\left(y; \boldsymbol{X}, \boldsymbol{\beta}\right),$$

where $F_a\left(\cdot\right)$ is a discrete cumulative distribution function, $\boldsymbol{X}$ is an $n \times p$ model matrix, and $\boldsymbol{\beta}$ is a $p \times 1$ vector of unknown regression coefficients. This functional form includes binary regression as a special case. For example, the probit model has

$$\mathcal{Y} \sim Bernoulli\left[\Phi\left(\boldsymbol{X}^\top \boldsymbol{\beta}\right)\right],$$

where $\Phi\left(\cdot\right)$ is the cumulative distribution function for the standard normal distribution.

The *cumulative link model* is a natural choice for modeling a binary or ordinal outcome. Consider an ordinal categorical outcome $\mathcal{Y}$ with ordered categories $1 < 2 < \cdots < J$. In a cumulative link model, the cumulative probabilities are linked to the predictors according to

$$G^{-1}\left(\Pr\left\{\mathcal{Y} \leq j\right\}\right) = \alpha_j + f\left(\boldsymbol{X}, \boldsymbol{\beta}\right), \tag{1}$$

where $G\left(\cdot\right)$ is a continuous cumulative distribution function, α_j are the category-specific intercepts, $\boldsymbol{X}$ is a matrix of covariates, and $\boldsymbol{\beta}$ is a vector of fixed regression coefficients. The intercept parameters satisfy $-\infty = \alpha_0 < \alpha_1 < \cdots < \alpha_{J-1} < \alpha_J = \infty$. We should point out that some authors (and software) use the alternate formulation

$$G^{-1}\left(\Pr\left\{\mathcal{Y} \geq j\right\}\right) = \alpha_j^\star + f\left(\boldsymbol{X}, \boldsymbol{\beta}^\star\right). \tag{2}$$

This formulation provides coefficients that are consistent with the ordinary logistic regression model. The estimated coefficients from model (2) will have the opposite sign as those in model (1); see, for example, Agresti (2010).

Another way to interpret the cumulative link model is through a *latent* continuous random variable $\mathcal{Z} = -f\left(\boldsymbol{X}, \boldsymbol{\beta}\right) + \epsilon$, where ϵ is a continuous random variable with location parameter 0, scale parameter 1, and cumulative distribution function $G\left(\cdot\right)$. We then construct an ordered factor according to the rule

$$y = j \quad if \quad \alpha_{j-1} < z \leq \alpha_j.$$

For $\epsilon \sim N\left(0, 1\right)$, this leads to the usual probit model for ordinal responses,

$$\Pr\left\{\mathcal{Y} \leq j\right\} = \Pr\left\{\mathcal{Z} \leq \alpha_j\right\} = \Pr\left\{-f\left(\boldsymbol{X}, \boldsymbol{\beta}\right) + \epsilon \leq \alpha_j\right\} = \Phi\left(\alpha_j + f\left(\boldsymbol{X}, \boldsymbol{\beta}\right)\right).$$

Common choices for the link function $G^{-1}(\cdot)$ and the implied (standard) distribution for ϵ are described in Table 1.

Link	Distribution of ϵ	$G(y)$	$G^{-1}(p)$
logit	logistic	$\exp(y) / [1 + \exp(y)]$	$\log[p/(1-p)]$
probit	standard normal	$\Phi(y)$	$\Phi^{-1}(p)$
log-log	Gumbel (max)	$\exp[-\exp(-y)]$	$-\log[-\log(p)]$
complementary log-log	Gumbel (min)	$1 - \exp[-\exp(y)]$	$\log[-\log(1-p)]$
cauchit	Cauchy	$\pi^{-1}\arctan(y) + 1/2$	$\tan(\pi p - \pi/2)$

Table 1: Common link functions. Note: the logit is typically the default link function used by most statistical software.

There are a number of R packages that can be used to fit cumulative link models (1) and (2). The recommended package **MASS** (Venables and Ripley, 2002) contains the function `polr` (proportional odds logistic regression) which, despite the name, can be used with all of the link functions described in Table 1. The **VGAM** package (Yee, 2017) has the `vglm` function for fitting vector generalized linear models, which includes the broad class of cumulative link models. By default, `vglm` uses the same parameterization as in Equation (1), but provides the option of using the parameterization seen in Equation (2); this will result in the estimated coefficients having the opposite sign. Package **ordinal** (Christensen, 2015) has the `clm` function for fitting cumulative link models. The popular **rms** package (Harrell, 2017) has two functions: `lrm` for fitting logistic regression and cumulative link models using the logit link, and `orm` for fitting ordinal regression models. Both of these functions use the parameterization seen in Equation (2).

The remainder of the paper is organized as follows. In the next section, we discuss the idea of *surrogate residuals* (Liu and Zhang, 2017) and talk about some important properties. Next, we briefly discuss jittering, bootstrapping, and how they apply to the surrogate approach. The Section "Surrogate residuals in R" introduces the **sure** package and discusses the various modeling packages it supports. The sections following demonstrate how **sure** can be used to detect misspecified mean structures, heteroscedasticity, misspecified link functions, and interaction effects, as well as check the proportionality assumption. The Section "Bitterness of wine" provides a real data analysis example. We end with a closing summary.

Surrogate residuals

For a continuous outcome $\mathcal{Y}$, the residual is traditionally defined as the difference between the observed and fitted values. For ordinal outcomes, the residuals are more difficult to define, and few definitions have been proposed in the literature. Liu et al. (2009) propose using the cumulative sums of residuals derived from collapsing the ordered categories into multiple binary outcomes. Unfortunately, this method leads to multiple residuals for the ordinal outcome and therefore is difficult to interpret. Li and Shepherd (2012) show that the sign-based statistic (SBS)

$$R_{SBS} = E\{sign(y - \mathcal{Y})\} = Pr\{y > \mathcal{Y}\} - Pr\{y < \mathcal{Y}\} \tag{3}$$

can be used as a residual for ordinal outcomes. Though Li and Shepherd refer to these as *probability-based residuals*, we will follow Liu and Zhang (2017) and refer to them as SBS residuals. For an overview of the theoretical and graphical properties of the SBS residual (3), see Liu and Zhang (2017) and the **PResiduals** package (Dupont et al., 2016). A limitation with the SBS residuals is that they are based on a discrete outcome and are discrete themselves, which makes them less useful in diagnostic plots.

Liu and Zhang (2017) propose a new type of residual that is based on a continuous variable S that acts as a surrogate for the ordinal outcome $\mathcal{Y}$. This surrogate residual is defined as

$$R_S = S - E(S|\boldsymbol{X}), \tag{4}$$

where S is a continuous variable based on the conditional distribution of the latent variable $\mathcal{Z}$ given $\mathcal{Y}$. In particular, given $\mathcal{Y} = y$, Liu and Zhang (2017) show that S follows a truncated distribution obtained by truncating the distribution of $\mathcal{Z} = -f(\boldsymbol{X}, \boldsymbol{\beta}) + \epsilon$ using the interval (α_{y-1}, α_y). The benefit of the surrogate residual (4) is that it is based on a continuous variable S, such that R_S is also continuous.

Furthermore, it can be shown (Liu and Zhang, 2017) that if the hypothesized model agrees with the true model, then R_S will have the following properties:

(a) **symmetry around zero** $E\left(R_S|\boldsymbol{X}\right) = 0$;

(b) **homogeneity** $Var\left(R_S|\boldsymbol{X}\right) = c$, a constant that is independent of $\boldsymbol{X}$;

(c) **reference distribution** the empirical distribution of R_S approximates an explicit distribution that is related to the link function $G^{-1}\left(\cdot\right)$. In particular, independent of $\boldsymbol{X}$, $R_S \sim G\left(c + \int udG(u)\right)$, where c is a constant.

According to property (a), if $\int udG\left(u\right) = 0$, then $R_S \sim G\left(\cdot\right)$. Properties (a)–(c) allow for a thorough examination of the residuals to check model adequacy and misspecification of the mean structure and link function.

Jittering for general models

The latent method discussed in Section "Surrogate residuals" applies to cumulative link models for ordinal outcomes. For more general models, we can define a surrogate using a technique called *jittering*. Suppose the true model for an ordinal outcome $\mathcal{Y}$ is

$$\mathcal{Y} \sim F_a\left(y; \boldsymbol{X}, \boldsymbol{\beta}\right), \tag{5}$$

where $F\left(\cdot\right)$ is a discrete cumulative distribution function. This model is general enough to cover the cumulative link models (1) and (2), and nearly any parametric or nonparametric model for ordinal outcomes.

Liu and Zhang (2017) suggest defining the surrogate S using either of the following two approaches:

1. jittering on the outcome scale: $S|\mathcal{Y} = y \sim \mathcal{U}\left[y, y+1\right]$;
2. jittering on the probability scale: $S|\mathcal{Y} = y \sim \mathcal{U}\left[F_a\left(y-1\right), F_a\left(y\right)\right]$.

Once a surrogate is obtained, we define the surrogate residuals in the same way as Equation (4). In either case, if the hypothesized model is correct, then symmetry around zero still holds; that is $E\left(R_S|\boldsymbol{X}\right) = 0$. For the latter case, if the hypothesized model is correct then $R_S|\boldsymbol{X} \sim \mathcal{U}\left(-1/2, 1/2\right)$. In other words, jittering on the probability scale has the additional property that the conditional distribution of R_S given $\boldsymbol{X}$ has an explicit form which allows for a full examination of the distributional information of the residual.

Bootstrapping

Since the surrogate residuals are based on random sampling, additional variability is introduced. One way to account for this sample variability and help stabilize any patterns in diagnostic plots is to use the bootstrap (Efron, 1979).

The procedure for bootstrapping surrogate residuals is similar to the model-based bootstrap algorithm used in linear regression. To obtain the b-th bootstrap replicate of the residuals, Liu and Zhang (2017) suggest the following algorithm:

Step 1 Perform a standard case-wise bootstrap of the original data to obtain the bootstrap sample $\left\{\left(\boldsymbol{X}^{\star}_{1b}, \mathcal{Y}^{\star}_{1b}\right), \ldots, \left(\boldsymbol{X}^{\star}_{nk}, \mathcal{Y}^{\star}_{nk}\right)\right\}$.

Step 2 Using the procedure outlined in the previous section, obtain a sample of surrogate residuals $R^{\star}_{S_{1b}}, \ldots, R^{\star}_{S_{nb}}$ using the bootstrap sample obtained in **Step 1**.

This procedure is repeated a total of B times. For residual-vs-covariate (i.e., R-vs-x) plots and residual-vs-fitted value (i.e., R-vs-$f\left(\boldsymbol{X}, \widehat{\boldsymbol{\beta}}\right)$) plots, we simply scatter all $B \times n$ residuals on the same plot. This approach is valid since the bootstrap samples are drawn independently. For large data sets, we find it useful to lower the opacity of the data points to help alleviate any issues with overplotting. For Q-Q plots, on the other hand, Liu and Zhang (2017) suggest using the median of the B bootstrap distributions, which is the implementation used in the **sure** package (Greenwell et al., 2017).

Surrogate residuals in R

The **sure** package supports a variety of R packages for fitting cumulative link and other types of models. The supported packages and their corresponding functions are described in Table 2.

The **sure** package currently exports four functions:

- `resids`—for constructing surrogate residuals;

Package	Function(s)	Model	Parameterization
stats	`glm`	binary regression	NA
MASS	`polr`	cumulative link	$Pr\{\mathcal{Y} \leq j\}$
rms	`lrm`	cumulative link	$Pr\{\mathcal{Y} \geq j\}$
	`lrm`	logistic regression	NA
	`orm`	cumulative link	$Pr\{\mathcal{Y} \geq j\}$
ordinal	`clm`	cumulative link	$Pr\{\mathcal{Y} \leq j\}$
VGAM	`vglm`	cumulative link	$Pr\{\mathcal{Y} \leq j\}$
	`vgam`	cumulative link	$Pr\{\mathcal{Y} \leq j\}$

Table 2: Ordinal regression modeling packages supported by **sure** and the corresponding parameterization they use for fitting cumulative link models. Note: by default, `vglm` uses the same parameterization as in Equation (1). This can be reversed by setting `reverse = TRUE` in the `family` argument.

- `surrogate`—for generating the surrogate response values used in the residuals;
- `autoplot`—for producing various diagnostic plots using **ggplot2** graphics (Wickham, 2009);
- `gof`—for simulating p-values from various goodness-of-fit tests.

In addition, the package also includes five simulated data sets: `df1`, `df2`, `df3`, `df4`, and `df5`. These data sets are used throughout the paper to demonstrate how the surrogate residual can be useful as a diagnostic tool for cumulative link models. The R code used to generate these data sets is available on the projects GitHub page: https://github.com/koalaverse/sure/blob/master/data-raw/data.R.

Detecting a misspecified mean structure

For illustration, the data frame `df1` contains $n = 2000$ observations from the following cumulative link model:

$$Pr\{\mathcal{Y} \leq j\} = \Phi\left(\alpha_j + \beta_1 X + \beta_2 X^2\right), \quad j = 1, 2, 3, 4, \tag{6}$$

where $\alpha_1 = -16$, $\alpha_2 = -12$, $\alpha_3 = -8$, $\beta_1 = -8$, $\beta_2 = 1$, and $X \sim \mathcal{U}(1, 7)$. These parameters were chosen to ensure that 1) the sample from the latent variable $\mathcal{Z}$ is spread out, rather than clustering in a small interval, and 2) each category of $\mathcal{Y}$ is well represented in the sample; we follow these guidelines throughout the simulated examples. The simulated data for this example are available in the `df1` data frame from the **sure** package and are loaded automatically with the package; see `?df1` for details. Below, we fit a (correctly specified) probit model using the `polr` function from the **MASS** package.

```
# Fit a cumulative link model with probit link
library(sure)  # for residual function and sample data sets
library(MASS)  # for polr function
fit.polr <- polr(y ~ x + I(x ^ 2), data = df1, method = "probit")
```

The code chunk below obtains the SBS residuals (3) from the previously fitted probit model `fit.polr` using the **PResiduals** package and constructs a couple of diagnostic plots. The results are displayed in Figure 1.

```
# Obtain the SBS/probability-scale residuals
library(PResiduals)
pres <- presid(fit.polr)

# Residual-vs-covariate plot and Q-Q plot
library(ggplot2)  # for plotting
p1 <- ggplot(data.frame(x = df1$x, y = pres), aes(x, y)) +
  geom_point(color = "#444444", shape = 19, size = 2, alpha = 0.5) +
  geom_smooth(color = "red", se = FALSE) +
  ylab("Probability-scale residual")
p2 <- ggplot(data.frame(y = pres), aes(sample = y)) +
  stat_qq(distribution = qunif, dparams = list(min = -1, max = 1), alpha = 0.5) +
  xlab("Sample quantile") +
  ylab("Theoretical quantile")
grid.arrange(p1, p2, ncol = 2)  # Figure 1
```

(Note: the reference distribution for the SBS residual is the $\mathcal{U}(-1,1)$ distribution.) As can be seen in the left side of Figure 1, the SBS residuals are inherently discrete and often display unusual patterns in diagnostic plots, making them less useful as a diagnostic tool.

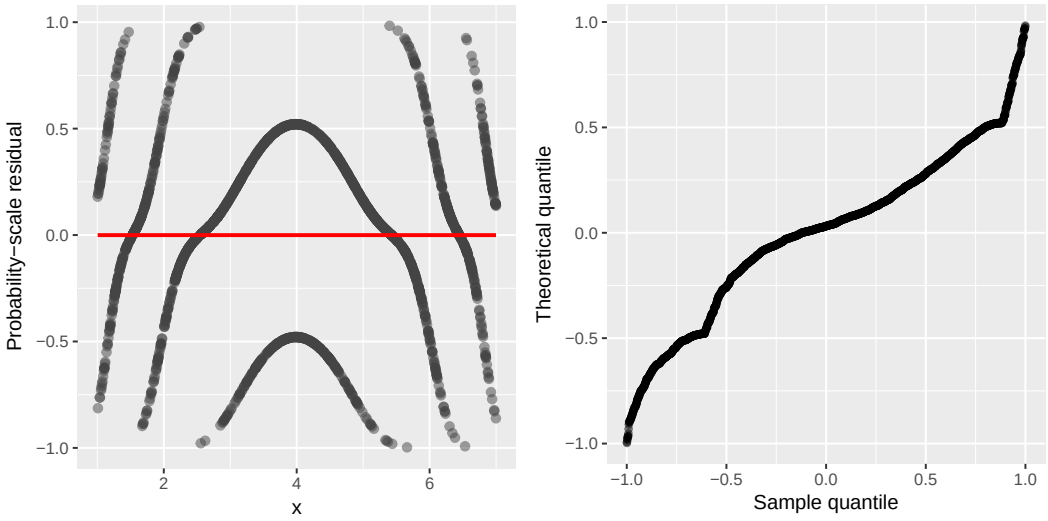


Figure 1: SBS residual plots for the (correctly specified) probit model fit to the `df1` data set. *Left*: Residual-vs-covariate plot with a nonparametric smooth (red curve). *Right*: Q-Q plot of the residuals.

Similarly, we can use the `resids` function in package **sure** to obtain the surrogate residuals discussed in Section "Surrogate residuals." This is illustrated in the following code chunk; the results are displayed in Figure 2. (Note: since the surrogate residuals are based on random sampling, we specify the seed via the `set.seed` function throughout this paper for reproducibility.)

```
# Obtain surrogate residuals
library(sure)
set.seed(101)  # for reproducibility
sres <- resids(fit.polr)

# Residual-vs-covariate plot and Q-Q plot
library(ggplot2)  # needed for autoplot function
p1 <- autoplot(sres, what = "covariate", x = df1$x, xlab = "x")
p2 <- autoplot(sres, what = "qq", distribution = qnorm)
grid.arrange(p1, p2, ncol = 2)  # Figure 2
```

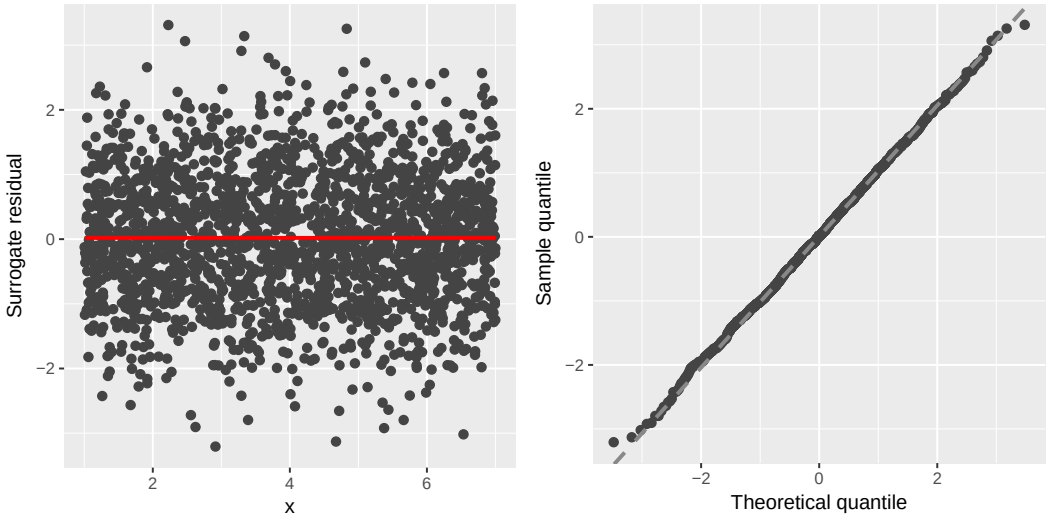


Figure 2: Surrogate residual plots for the (correctly specified) probit model fit to the `df1` data set. *Left*: Residual-vs-covariate plot with a nonparametric smooth (red curve). *Right*: Q-Q plot of the residuals.

The **sure** package also includes `autoplot` methods for the various classes of models listed in Table 2, so that the user can give `autoplot` the fitted model directly. The benefit of this approach is that the fitted values and reference distribution (used in Q-Q plots) are automatically extracted. For example, to reproduce the Q-Q plot in Figure 2, we could have used the following:

```
set.seed(101)  # for reproducibility
autoplot(fit.polr, what = "qq")  # same as top right of Figure 1
```

Suppose that we did not include the quadratic term in our fitted model. We would expect a residual-vs-x plot to indicate that such a quadratic term is missing. Below, we update the previously fitted model by removing the quadratic term, then update the residual-vs-covariate plots (code not shown). The updated residual plots are displayed in Figure 3.

```
fit.polr <- update(fit.polr, y ~ x)  # remove quadratic term
```

The SBS residuals gives some indication of a misspecified mean structure, but this only becomes more clear with increasing J, and the plot is still discrete. This is overcome by the surrogate residuals which produces a residual plot not unlike those seen in ordinary linear regression models.

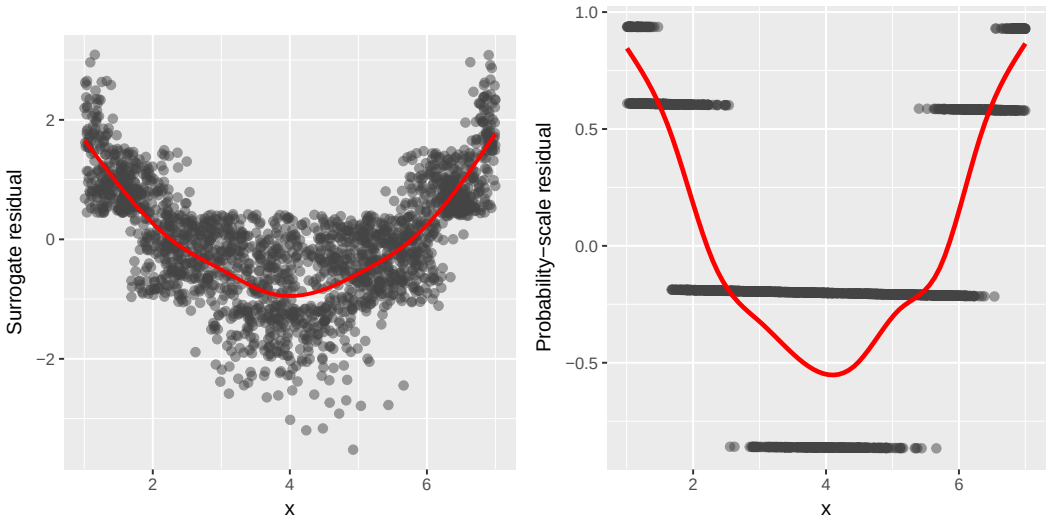


Figure 3: Residual-vs-covariate plots with nonparametric smooths (red curves) for a probit model with a misspecified mean structure fit to the simulated data from model (6). *Left*: Surrogate residuals. *Right*: SBS residuals.

Detecting heteroscedasticty

One issue that often raises concerns in statistical inference is that of heteroscedasticity; that is, when the error term has non constant variance. Heteroscedasticity can bias the statistical inference and lead to improper standard errors, confidence intervals, and p-values. Therefore, it is imperative to identify heteroscedasticity whenever present and take appropriate action (e.g., transformations). In ordinary linear regression, this topic has been covered extensively. Not much has been proposed in the literature for categorical models.

As discussed in Section "Surrogate residuals," one of the properties of the surrogate residual R_S is that, if the model is specified correctly, then $Var\left(R_S|X\right) = c$, where c is a constant.

For this example, we generated $n = 2000$ observations from the following ordered probit model:

$$Pr\left\{\mathcal{Y} \leq j\right\} = \Phi\left\{\left(\alpha_j + \beta X\right)/\sigma_X\right\}, \quad j = 1,2,3,4,5,$$

where $\alpha_1 = -36$, $\alpha_2 = -6$, $\alpha_3 = 34$, $\alpha_4 = 64$, $\beta = -4$, $X \sim \mathcal{U}\left(2,7\right)$, and $\sigma_X = X^2$. Notice how the variability is an increasing function of X. These data are available in the `df2` data frame loaded with the **sure** package; see `?df2` for details.

The following block of code uses the `orm` function from the popular **rms** package to fit a probit model to the simulated data. Note: we set `x = TRUE` in the call to `orm` in order to use the `presid` function later.

```
# Fit a cumulative link model with probit link
library(rms)  # for orm function
fit.orm <- orm(y ~ x, data = df2, family = "probit", x = TRUE)
```

If heteroscedasticity is present, we would expect this to show up in various diagnostic plots, such as a residual-vs-covariate plot. Below we obtain the SBS and surrogate residuals as before and plot them against X. The results are displayed in Figure 4.

```
set.seed(102)  # for reproducibility
p1 <- autoplot(resids(fit.orm), what = "covariate", x = df2$x, xlab = "x")
p2 <- ggplot(data.frame(x = df2$x, y = presid(fit.orm)), aes(x, y)) +
  geom_point(color = "#444444", shape = 19, size = 2, alpha = 0.25) +
  geom_smooth(col = "red", se = FALSE) +
  ylab("Probability scale residual")
grid.arrange(p1, p2, ncol = 2)  # Figure 4
```

In Figure 4, it is clear from the plot of the surrogate residuals (left side of Figure 4) that the variance increases with X, a sign of heteroscedasticity. As a matter of fact, the plot suggests that the true link function has a varying scale parameter, $\sigma = \sigma(\boldsymbol{X})$. The plot of the SBS residuals (right side of Figure 4), on the other hand, gives no indication of an issue with nonconstant variance.

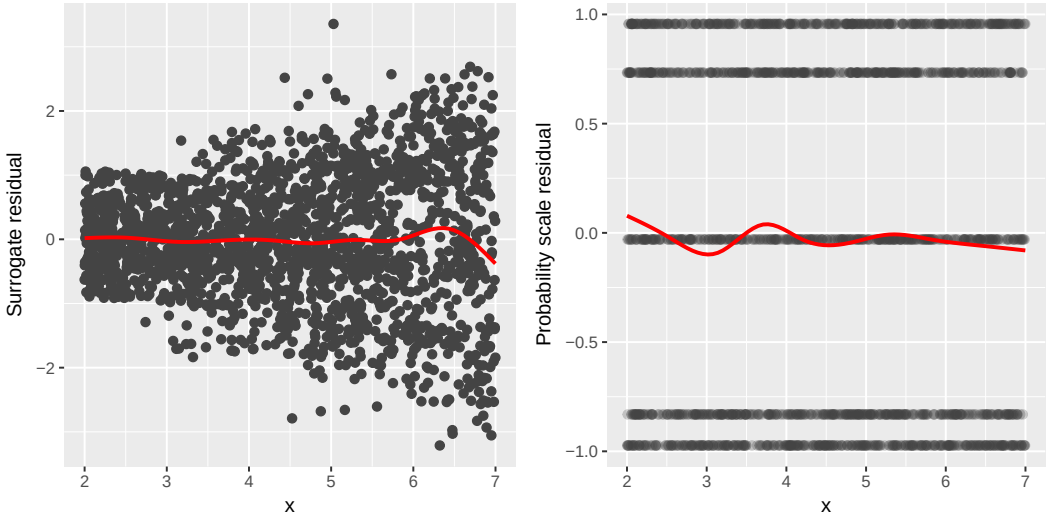


Figure 4: Residual-vs-covariate plots with nonparametric smooths (red curves) for the simulated heteroscedastic data. *Left*: Surrogate residuals. *Right*: SBS residuals.

As outlined in Section "Jittering for general models," the jittering technique is broadly applicable to virtually all parametric and nonparametric models for ordinal responses. To illustrate, the code chunk below uses the **VGAM** package to fit a *vector generalized additive model* to the same data using a nonparametric smooth for x.

```
library(VGAM)  # for vgam and vglm functions
fit.vgam <- vgam(y ~ s(x), family = cumulative(link = probit, parallel = TRUE),
                 data = df2)
```

To obtain a surrogate residual using the jittering technique, we can set `method = "jitter"` in the call to `resids` or `autoplot`. There is also the option `jitter.scale` which can be set to either `"probability"`, for jittering on the probability scale (the default), or `"response"`, for jittering on the response scale. In the code chunk below, we use the `autoplot` function to obtain residual-by-covariate plots using both types of jittering. The results (Figure 5) indicate that the variance increases with increasing x.

```
set.seed(103)  # for reproducibility
p1 <- autoplot(fit.vgam, what = "covariate", x = df2$x, method = "jitter",
               xlab = "x")
p2 <- autoplot(fit.vgam, what = "covariate", x = df2$x, method = "jitter",
               jitter.scale = "response", xlab = "x")
grid.arrange(p1, p2, ncol = 2)  # Figure 5
```

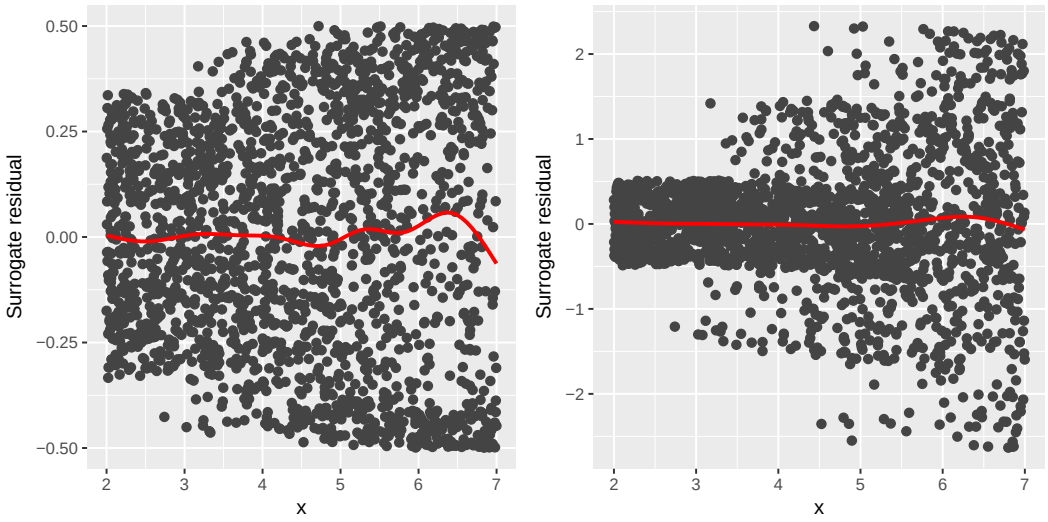


Figure 5: Residual-vs-covariate plots with nonparametric smooths (red curves) from a vector generalized additive model fit to the simulated heteroscedastic data. *Left*: Jittering on the probability scale (default). *Right*: Jittering on the response scale.

Detecting a misspecified link function

For this example, we simulated $n = 2000$ observations from the following model

$$Pr(\mathcal{Y} \leq j) = G\left(\alpha_j - 8X + X^2\right), \quad j = 1, 2, 3, 4,$$

where $G(\cdot)$ is the CDF for the Gumbel (max) distribution (see Table 1), $\alpha_1 = -16$, $\alpha_2 = -12$, $\alpha_3 = -8$, $\beta_1 = -8$, $\beta_2 = 1$, and $X \sim \mathcal{U}(1, 7)$. The data are available in the data frame `df3` within the package; see `?df3` for details.

We fit a model with various link functions, where the true link function is log-log. From these output, we construct Q-Q plots of the residuals using $R = 100$ bootstrap replicates. The results are displayed in Figure 6.

```
# Fit models with various link functions to the simulated data
fit.probit <- polr(y ~ x + I(x ^ 2), data = df3, method = "probit")
fit.logistic <- polr(y ~ x + I(x ^ 2), data = df3, method = "logistic")
fit.loglog <- polr(y ~ x + I(x ^ 2), data = df3, method = "loglog")  # correct link
fit.cloglog <- polr(y ~ x + I(x ^ 2), data = df3, method = "cloglog")

# Construct Q-Q plots of the surrogate residuals for each model
set.seed(1056)  # for reproducibility
p1 <- autoplot(fit.probit, nsim = 100, what = "qq")
p2 <- autoplot(fit.logistic, nsim = 100, what = "qq")
p3 <- autoplot(fit.loglog, nsim = 100, what = "qq")
p4 <-  autoplot(fit.cloglog, nsim = 100, what = "qq")

# Figure 6
grid.arrange(p1, p2, p3, p4, ncol = 2)  # bottom left plot is correct model
```

From the Q-Q plots in Figure 6, it is clear the the model with the log-log link (which corresponds to Gumbel (max) errors in the latent variable formulation) is the most appropriate, while the other plots indicate deviations from the hypothesized model.

Alternatively, we could also use the surrogate residuals to make use of existing distance-based goodness-of-fit (GOF) tests; for example, the Kolmogorov-Smirnov distance. The `gof` function in **sure** can be used to produce simulated p-values from such tests.

Currently, the `gof` function supports three goodness-of-fit tests: the Kolmogorov-Smirnov test (`test = "ks"`), the Anderson-Darling test (`test = "ad"`), and the Cramer-Von Mises test (`test = "cvm"`). Below, we use the `gof` function to simulate p-values from the Anderson-Darling test for each of the four models; we also set `nsim` to 100 to produce smoother plots and reduce the sampling error induced by the surrogate procedure. The `plot` method is then used to display the empirical distribution function (EDF) of the simulated p-values. A good fit would imply uniformly distributed

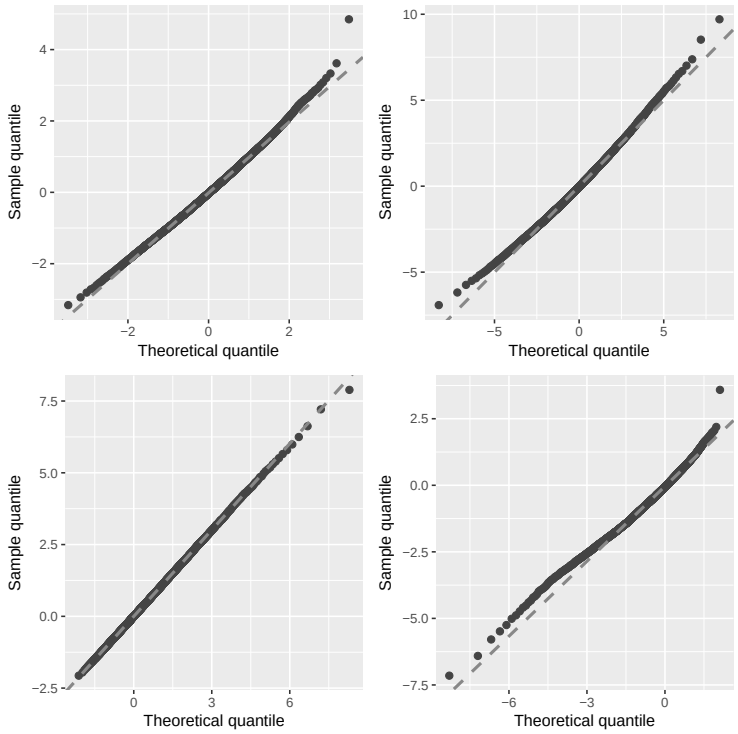


Figure 6: Q-Q plots of the residuals for various cumulative link models fit to simulated data with Gumbel (max) errors. *Top left*: A model with probit link. *Top right*: A model with logit link. *Bottom left*: A model with log-log link (i.e., the correct model). *Bottom right*: A model with complementary log-log link.

p-values; hence, the EDF would be relatively straight with a slope of one. The results in Figure 7 agree with the Q-Q plots from Figure 6 in that the log-log link is the most appropriate for these data. (Note: the plotting method for `"gof"` objects uses base R graphics; hence, we can use the `par` function to set various graphical parameters.)

```
# Figure 7
par(mfrow = c(2, 2), mar = c(2, 4, 2, 2) + 0.1)
set.seed(8491)  # for reproducibility
plot(gof(fit.probit, nsim = 100, test = "ad"), main = "")
plot(gof(fit.logistic, nsim = 100, test = "ad"), main = "")
plot(gof(fit.loglog, nsim = 100, test = "ad"), main = "")
plot(gof(fit.cloglog, nsim = 100, test = "ad"), main = "")
```

Checking the proportionality assumption

An important feature of the cumulative link model (1) is the proportional odds assumption, which assumes that the mean structure, $f(\boldsymbol{X}, \boldsymbol{\beta})$, remains the same for each of the J categories; for the logit case (see row one of Table 1), this is also referred to as the proportional odds assumption. Harrell (2001, pp. 334–335) suggests computing each observation's contribution to the first derivative of the log likelihood function with respect to $\boldsymbol{\beta}$, averaging them within each of the J categories, and examining any trends in the residual plots, but these plots can be difficult to interpret. Fortunately, it is relatively straightforward to use the simulated surrogate response values S to check the proportionality assumption.

To illustrate, we generated 2000 observations from each of the following probit models

$$Pr(\mathcal{Y} \leq j) = \Phi\left(\alpha_j + \beta_1 X\right), \quad j = 1,2,3, \quad \text{and} \quad Pr(\mathcal{Y} \leq j) = \Phi\left(\alpha_j + \beta_2 X\right), \quad j = 4,5,6,$$

where $\alpha_1 = -1.5$, $\alpha_2 = 0$, $\alpha_3 = 1$, $\alpha_4 = 3$, $\beta_1 = 1$, $\beta_2 = 1.5$, and $X \sim \mathcal{U}(-3,3)$. The data are available in the data frame `df4` within the package; see `?df4` for details.

Checking the proportionality assumption here amounts to checking whether or not $\beta_1 - \beta_2 = 0$. As outlined in Liu and Zhang (2017), we can generate surrogates $S_1 \sim \mathcal{N}(-\beta_1 X, 1)$ and $S_2 \sim$

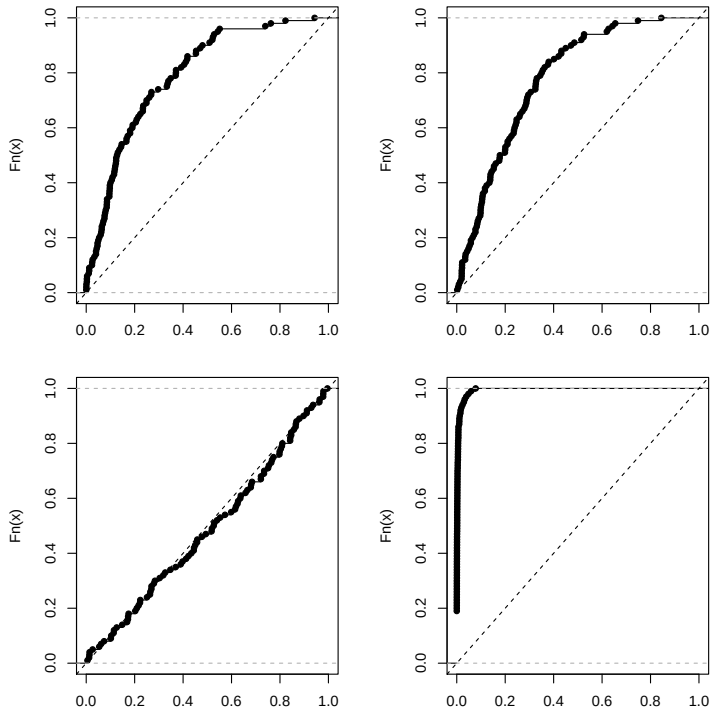


Figure 7: EDFs of the simulated p-values from an Anderson-Darling GOF test for various cumulative link models fit to simulated data with gumbel errors. *Top left*: A model with probit link. *Top right*: A model with logit link. *Bottom left*: A model with log-log link (i.e., the correct model). *Bottom right*: A model with complementary log-log link.

$\mathcal{N}(-\beta_2 X, 1)$, both conditional on X. We then define the difference $D = S_2 - S_1$ which, conditional on X, follows a $\mathcal{N}((\beta_1 - \beta_2) X, 1)$ distribution. If $\beta_1 - \beta_2 = 0$, then D should be independent of X. This can be easily checked by plotting D against X. Below, we use the `surrogate` function to generate the surrogate response values directly (as opposed to the residuals) and generate the D-vs-X plot shown in Figure 8.

```
# Fit separate models (VGAM should already be loaded)
fit1 <- vglm(y ~ x, data = df4[1:2000, ],
             cumulative(link = probit, parallel = TRUE))
fit2 <- update(fit1, data = df4[2001:4000, ])

# Generate surrogate response values
set.seed(8671)  # for reproducibility
s1 <- surrogate(fit1)
s2 <- surrogate(fit2)

# Figure 8
ggplot(data.frame(D = s1 - s2, x = df4[1:2000, ]$x) , aes(x = x, y = D)) +
  geom_point(color = "#444444", shape = 19, size = 2) +
  geom_smooth(se = FALSE, size = 1.2, color = "red")
```

From Figure 8, it is clear that $\beta_1 - \beta_2 \neq 0$; hence, the proportionality assumption does not hold.

Detecting interaction effects

A common challenge in model building is determining whether or not there are important interactions between the predictors in the data. Using the surrogate residuals, it is rather straightforward to determine if such an interaction effect is missing from the assumed model.

For illustration, we generated $n = 2000$ observations from the following ordered probit model

$$Pr(\mathcal{Y} \leq j) = \Phi\left(\alpha_j + \beta_1 x_1 + \beta_2 x_2 + \beta_3 x_1 x_2\right), \quad j = 1, 2, 3, 4,$$

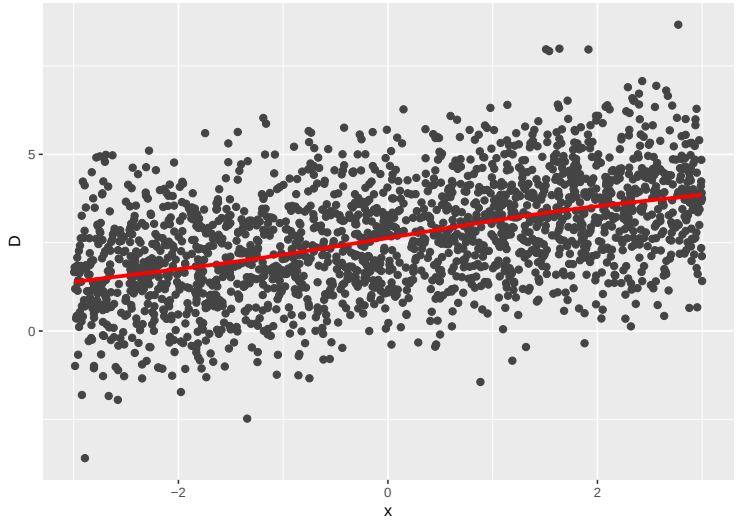


Figure 8: Scatterplot of $D = S_1 - S_2$ vs. x with a nonparametric smooth (red curve).

where $\alpha_1 = -16$, $\alpha_2 = -12$, $\alpha_3 = -8$, $\beta_1 = -5$, $\beta_2 = 3$, $\beta_3 = 10$, $x_1 \sim \mathcal{U}(1,7)$, and x_2 is a factor with levels `Treatment` and `Control`. The simulated data are available in the `df5` data frame loaded with the **sure** package; see `?df5` for details. Below, we fit two probit models using the `clm` function from the **ordinal** package. The first model corresponds to the simulated control group, while the second model corresponds to the treatment group.

```
library(ordinal)  # for clm function
fit1 <- clm(y ~ x1, data = df5[df5$x2 == "Control", ], link = "probit")
fit2 <- clm(y ~ x1, data = df5[df5$x2 == "Treatment", ], link = "probit")
```

If the true model contains an interaction term $x_1 x_2$ but the fitted model does not include it, we can detect this misspecification using the surrogate residuals. We simply plot R_S versus x_1 for the treatment group, and compare it to the plot of R_S versus x_1 for the controls—or better yet, we can just use the surrogate response S instead of R_S. The trends in these two plots should be different. Below, we use `ggplot` along with the `sure` function to construct such a plot in Figure 9. The plot indicates a negative association between x_1 and the outcome within the control group, and a positive association between x_1 and the treatment group (i.e., an interaction between x_1 and x_2).

```
# Figure 9
set.seed(1105)  # for reproducibility
df5$s <- c(surrogate(fit1), surrogate(fit2))  # surrogate response values
ggplot(df5, aes(x = x1, y = s)) +
  geom_point(color = "#444444", shape = 19, size = 2, alpha = 0.5) +
  geom_smooth(se = FALSE, size = 1.2, color = "red2") +
  facet_wrap( ~ x2) +
  ylab("Surrogate response") +
  xlab(expression(x[1]))
```

Bitterness of wine

Randall (1989) performed an experiment on factors determining the bitterness of wine. Two binary treatment factors, temperature and contact (between juice and skin), were controlled while crushing the grapes during wine production. Nine judges each assessed wine from two bottles from each of the four treatment conditions; for a total of $n = 72$. The response is an ordered factor with levels $1 < 2 < 3 < 4 < 5$. The data are available in the **ordinal** package; see `?ordinal::wine` for details.

```
data(wine, package = "ordinal")  # load wine data set
wine.clm <- clm(rating ~ temp + contact, data = wine, link = "probit")
```

Since both of the covariates in this model are binary factors, scatterplots are not appropriate for displaying the residual-by-covariate relationships. Instead, the `autoplot` function in `sure` uses boxplots; a future release is likely to include the additional option for producing nonparametric densities for each level of a factor. The code chunk below uses `autoplot` along with `grid.arrange` to

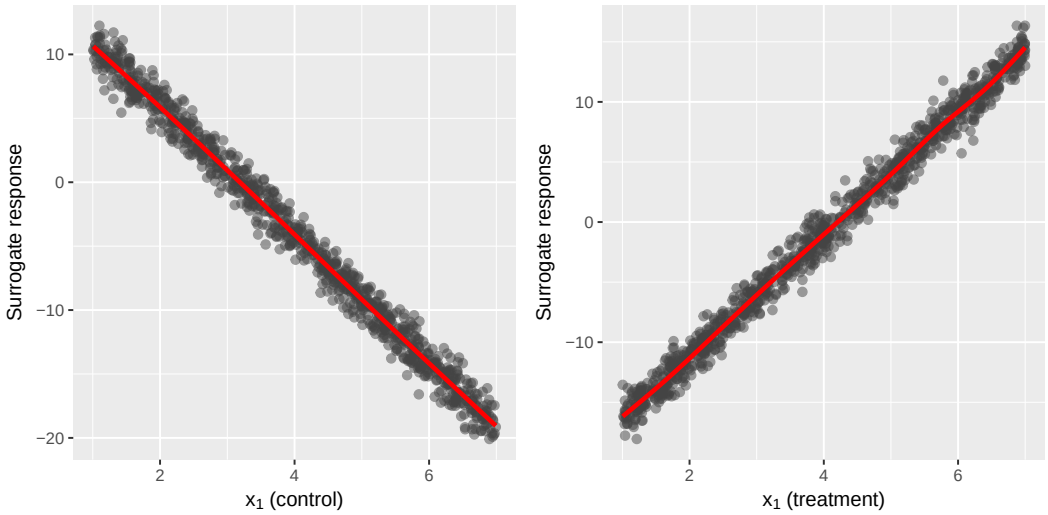


Figure 9: Scatterplot of the surrogate response S versus x_1 with a nonparametric smooth (red line). *Left*: Control group. *Right*: Treatment group.

produce some standard residual diagnostic plots. The results are displayed in Figure 10. The Q-Q plot and residual-vs-fitted value plot do not indicate any serious model misspecifications. Furthermore, the boxplots reveal that the medians of the surrogate residuals are very close to zero, and the distribution of the residuals within each level appear to be symmetric and have approximately the same variability (with the exception of a few outliers).

```
set.seed(1225)  # for reproducibility
grid.arrange(  # Figure 10
  autoplot(wine.clm, nsim = 10, what = "qq"),
  autoplot(wine.clm, nsim = 10, what = "fitted", alpha = 0.5),
  autoplot(wine.clm, nsim = 10, what = "covariate", x = wine$temp,
           xlab = "Temperature"),
  autoplot(wine.clm, nsim = 10, what = "covariate", x = wine$contact,
           xlab = "Contact"),
  ncol = 2
)
```

Summary

In this paper, we introduce the **sure** package, which implements the surrogate approach to residuals for ordinal regression models described in Liu and Zhang (2017). Using simulated data sets, we illustrate the various properties of these residuals and how they can be used to check assumptions in the ordinal regression model (e.g., heteroscedasticity, misspecified link functions, etc.) using continuous residuals which produce diagnostic plots not unlike those seen in ordinary linear regression. This offers a new way of performing typical diagnostic checks for ordinal regression models that are easily interpreted by the analyst. Furthermore, this new approach to constructing residuals for ordinal regression models is still under active development and new functionality will be added to **sure** in the future.

Acknowledgments

The authors would like to thank the anonymous reviewers and the Editor for their helpful comments and suggestions.

Bibliography

A. Agresti. *Analysis of Ordinal Categorical Data*. Wiley Series in Probability and Statistics. John Wiley & Sons, 2010. ISBN 9781118209998. [p381]

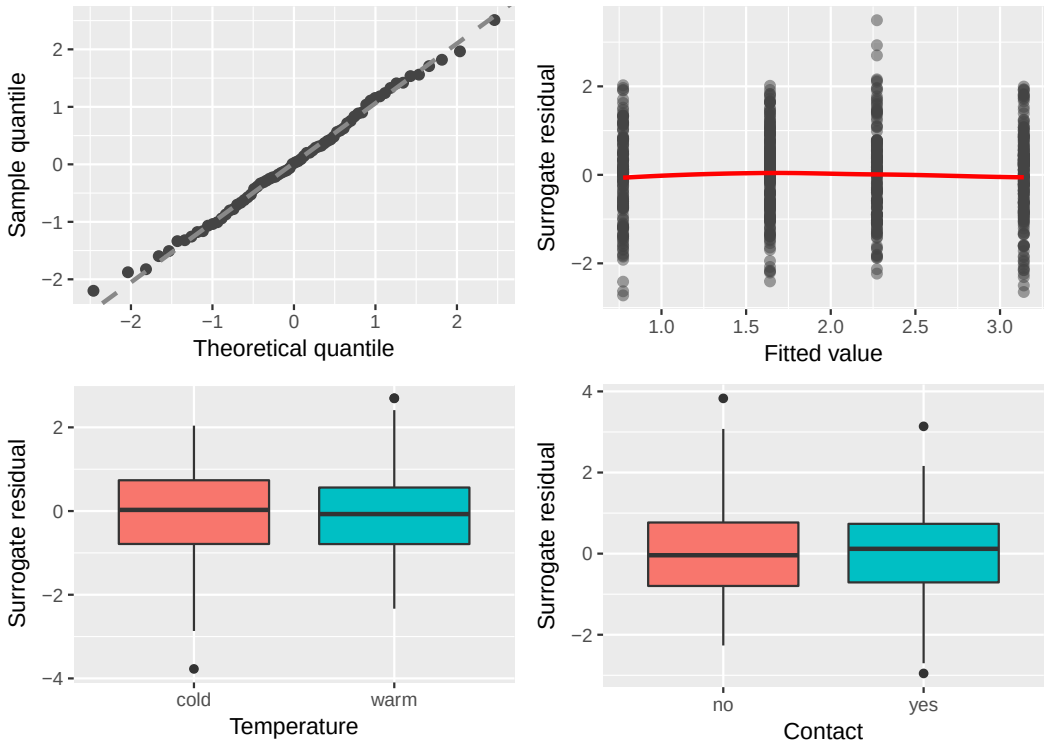


Figure 10: Residual diagnostic plots for the quality of wine example.

R. H. B. Christensen. *Ordinal—Regression Models for Ordinal Data*, 2015. URL http://www.cran.r-project.org/package=ordinal. R package version 2015.6-28. [p382]

C. Dupont, J. Horner, C. Li, Q. Liu, and B. Shepherd. *PResiduals: Probability-Scale Residuals and Residual Correlations*, 2016. URL https://CRAN.R-project.org/package=PResiduals. R package version 0.2-4. [p382]

B. Efron. Bootstrap methods: Another look at the jackknife. *Annals fo Statistics*, 7(1):1–26, 1979. URL http://dx.doi.org/10.1214/aos/1176344552. [p383]

B. Greenwell, A. McCarthy, and B. Boehmke. *Sure: Surrogate Residuals for Ordinal and General Regression Models*, 2017. URL https://CRAN.R-project.org/package=sure. R package version 0.1.2.9000. [p383]

F. E. Harrell. *Regression Modeling Strategies: With Applications to Linear Models, Logistic Regression, and Survival Analysis*. Graduate Texts in Mathematics. Springer-Verlag, 2001. ISBN 9780387952321. [p389]

F. E. Harrell. *Rms: Regression Modeling Strategies*, 2017. URL https://CRAN.R-project.org/package=rms. R package version 5.1-1. [p382]

C. Li and B. E. Shepherd. A new residual for ordinal outcomes. *Biometrika*, 99(2):473–480, 2012. URL http://dx.doi.org/10.1093/biomet/asr073. [p382]

D. Liu and H. Zhang. Residuals and diagnostics for ordinal regression models: A surrogate approach. *Journal of the American Statistical Association*, 0(ja):0–0, 2017. URL http://dx.doi.org/10.1080/01621459.2017.1292915. [p382, 383, 389, 392]

I. Liu, B. Mukherjee, T. Suesse, D. Sparrow, and S. K. Park. Graphical diagnostics to check model misspecification for the proportional odds regression model. *Statistics in Medicine*, 28(3):412–429, 2009. URL http://dx.doi.org/10.1080/01621459.2017.1292915. [p382]

J. H. Randall. The analysis of sensory data by generalized linear model. *Biometrical Journal*, 31(7): 781–793, 1989. URL http://dx.doi.org/10.1002/bimj.4710310703. [p391]

W. N. Venables and B. D. Ripley. *Modern Applied Statistics with S*. Springer-Verlag, New York, 4th edition, 2002. URL http://www.stats.ox.ac.uk/pub/MASS4. ISBN 0-387-95457-0. [p382]

H. Wickham. *Ggplot2: Elegant Graphics for Data Analysis*. Springer-Verlag, 2009. ISBN 978-0-387-98140-6. URL http://ggplot2.org. [p384]

T. W. Yee. *VGAM: Vector Generalized Linear and Additive Models*, 2017. URL https://CRAN.R-project.org/package=VGAM. R package version 1.0-3. [p382]

Brandon M. Greenwell
Department of Mathematics and Statistics
Wright State University
3640 Colonel Glenn Hwy
Dayton, OH 45435
United States of America
ORCiD: 0000-0002-8120-0084
greenwell.brandon@gmail.com

Andrew McCarthy
The Perduco Group
3610 Pentagon Blvd
Suite 110
Beavercreek, OH 45431
United States of America
andymc82000@yahoo.com

Bradley C. Boehmke
University of Cincinnati
2925 Campus Green Dr
Cincinnati, OH 45221
United States of America
bradleyboehmke@gmail.com

Dungang Liu
University of Cincinnati
2925 Campus Green Dr
Cincinnati, OH 45221
United States of America
dungang.liu@uc.edu

Advanced Bayesian Multilevel Modeling with the R Package brms

by Paul-Christian Bürkner

Abstract The **brms** package allows R users to easily specify a wide range of Bayesian single-level and multilevel models which are fit with the probabilistic programming language Stan behind the scenes. Several response distributions are supported, of which all parameters (e.g., location, scale, and shape) can be predicted. Non-linear relationships may be specified using non-linear predictor terms or semi-parametric approaches such as splines or Gaussian processes. Multivariate models can be fit as well. To make all of these modeling options possible in a multilevel framework, **brms** provides an intuitive and powerful formula syntax, which extends the well known formula syntax of **lme4**. The purpose of the present paper is to introduce this syntax in detail and to demonstrate its usefulness with four examples, each showing relevant aspects of the syntax.

Introduction

Multilevel models (MLMs) offer great flexibility for researchers across sciences (Brown and Prescott, 2015; Demidenko, 2013; Gelman and Hill, 2006; Pinheiro and Bates, 2006). They allow modeling of data measured on different levels at the same time – for instance students nested within classes and schools – thus taking complex dependency structures into account. It is not surprising that many packages have been developed to fit MLMs in R. Usually, the functionality of these implementations is limited insofar as they only predict the mean of the response distribution. Other parameters of the response distribution, such as the residual standard deviation in linear models, are assumed constant across observations, an assumption which may be violated in many applications. Accordingly, it is desirable to allow prediction of *all* response parameters at the same time. Models that perform simultaneous estimation are often referred to as *distributional models* or, more verbosely, as *models for location, scale, and shape* (Rigby and Stasinopoulos, 2005).

Another limitation of basic MLMs is that they only allow for linear predictor terms. While linear predictor terms already offer good flexibility, they are of limited use when relationships are inherently non-linear. Such non-linearity can be handled in at least two ways: (1) by fully specifying a non-linear predictor term with corresponding parameters, each of which can be predicted using MLMs (Lindstrom and Bates, 1990); or (2) estimating the form of the non-linear relationship on-the-fly using splines (Wood, 2004) or Gaussian processes (Rasmussen and Williams, 2006). The former are often simply called *non-linear models*, while models applying splines are referred to as *generalized additive models* (GAMs, see Hastie and Tibshirani, 1990).

Combining all of these modeling options into one framework is a complex task, both conceptually and with regard to model fitting. Maximum likelihood methods, typically applied in classical 'frequentist' statistics, can reach their limit at some point such that fully Bayesian methods become the go-to solutions to fit such complex models (Gelman et al., 2014). In addition to being more flexible, the Bayesian framework comes with other advantages, such as the ability to derive probability statements for every quantity of interest or explicitly incorporate prior knowledge about parameters into the model. The former is particularly relevant in non-linear models, for which classical approaches struggle more often than not in propagating all the uncertainty in the parameter estimates to non-linear functions such as out-of-sample predictions.

Possibly the most powerful program for performing full Bayesian inference available to date is Stan (Stan Development Team, 2017c; Carpenter et al., 2017), which implements Hamiltonian Monte Carlo (Duane et al., 1987; Neal, 2011; Betancourt et al., 2014) and its extension, the No-U-Turn (NUTS) Sampler (Hoffman and Gelman, 2014). These algorithms converge much more quickly than other Markov-Chain Monte-Carlo (MCMC) algorithms, especially for high-dimensional models (Hoffman and Gelman, 2014; Betancourt et al., 2014; Betancourt, 2017). An excellent non-mathematical introduction to Hamiltonian Monte Carlo can be found in Betancourt (2017).

Stan comes with its own programming language, allowing for great modeling flexibility (Stan Development Team, 2017c; Carpenter et al., 2017). Many researchers may still be hesitant to use Stan directly, as every model has to be written, debugged, and possibly also optimized, which may be a time-consuming and error-prone process even for researchers familiar with Bayesian inference. The **brms** package (Bürkner, 2017) presented in this paper aims to remove these hurdles for a wide range of regression models by allowing the user to benefit from the merits of Stan by using extended **lme4**-like formula syntax (Bates et al., 2015), with which many R users are familiar. The **brms** package offers much more than writing efficient and human-readable Stan code. It comes with many post-processing

and visualization functions, including functions for posterior predictive checks, leave-one-out cross-validation, visualization of estimated effects, and prediction of new data. The overarching aim is to have one general framework for regression modeling, which offers everything required to apply regression models to complex data. To date, it replaces and extends the functionality of dozens of other R packages, each of which is restricted to specific regression models. Users may even define their own response distributions and run them via brms (for details, see `vignette("brms_customfamilies")`).

The purpose of the present article is to provide an introduction to the advanced multilevel formula syntax implemented in **brms**, which fits a wide and growing range of non-linear distributional multilevel models. A general overview of the package is given in Bürkner (2017). Accordingly, the present article focuses on more recent developments. We begin by explaining the underlying structure of distributional models. Next, the formula syntax of **lme4** and the extensions implemented in **brms** are explained. Four examples that demonstrate the use of the new syntax are discussed in detail. Afterwards, the functionality of **brms** is compared with that of **rstanarm** (Stan Development Team, 2017a) and **MCMCglmm** (Hadfield, 2010). We end by describing future plans for extending the package.

Model description

The core model implemented in **brms** is the prediction of the response y through predicting all parameters θ_p of the response distribution D, which is also called the model `family` in many R packages. We write

$$y_i \sim D(\theta_{1i}, \theta_{2i}, \dots)$$

to stress the dependency on the i^{th} observation. If desired, every parameter θ_p may be regressed on its own predictor term η_p, transformed by the inverse link function f_p that is $\theta_{pi} = f_p(\eta_{pi})$. Such models are typically referred to as *distributional models*. The models described in Bürkner (2017) are a sub-class of the models described here. Details about the parameterization of each `family` are given in `vignette("brms_families")`.

Suppressing the index p for simplicity, a predictor term η can generally be written as

$$\eta = \mathbf{X}\beta + \mathbf{Z}u + \sum_{k=1}^{K} s_k(x_k)$$

where β and u are the respective coefficients at the population-level and group-level, and $\mathbf{X}, \mathbf{Z}$ are the corresponding design matrices. The terms $s_k(x_k)$ symbolize optional smooth functions of unspecified form based on covariates x_k fitted via splines (see Wood, 2011, for the underlying implementation in the **mgcv** package) or Gaussian processes (Williams and Rasmussen, 1996). The response y as well as $\mathbf{X}$, $\mathbf{Z}$, and x_k make up the data, whereas β, u, and the smooth functions s_k are the model parameters being estimated. The coefficients β and u may be more commonly known as fixed and random effects, but we avoid theses terms following the recommendations of Gelman and Hill (2006). Details about prior distributions for β and u can be found in Bürkner (2017) and under `help("set_prior")`.

As an alternative to the strictly additive formulation described above, predictor terms may also have any form specifiable in Stan. We call it a *non-linear* predictor and write

$$\eta = f(c_1, c_2, \dots, \phi_1, \phi_2, \dots)$$

The structure of the function f is given by the user, c_r are known or observed covariates, and ϕ_s are non-linear parameters, each having its own linear predictor term η_{ϕ_s} of the form specified above. In fact, we should think of non-linear parameters as placeholders for linear predictor terms rather than as parameters themselves. A frequentist implementation of such models, which inspired the non-linear syntax in **brms**, can be found in the **nlme** package (Pinheiro et al., 2016).

Extended multilevel formula syntax

The formula syntax applied in **brms** builds upon the syntax of the R package **lme4** (Bates et al., 2015). First, we will briefly explain the **lme4** syntax used to specify multilevel models and then introduce certain extensions that allow specifying much more complicated models in **brms**.

An **lme4** formula has the general form

```
response ~ pterms + (gterms | group)
```

The `pterms` contain population-level effects, assumed to be the same across observations. The `gterms` contain so called group-level effects, assumed to vary across the grouping variables specified in `group`.

Multiple grouping factors each with multiple group-level effects are possible. Usually, `group` contains only a single variable name pointing to a factor, but users may also use `g1:g2` or `g1/g2` if both `g1` and `g2` are suitable grouping factors. The `:` operator creates a new grouping factor that consists of the combined levels of `g1` and `g2`. The `/` operator indicates nested grouping structures and expands one grouping factor into two or more when using multiple `/` within one term. For instance, if the user were to write `(1 | g1/g2)`, **brms** will expand this to `(1 | g1) + (1 | g1:g2)`. Instead of `|`, users may use `||` in grouping terms to prevent group-level correlations from being modeled. This may be used to reduce model complexity if convergence of the sampler is an issue. One limitation of the `||` operator in **lme4** is that it only splits up terms so that columns of the design matrix originating from the same term are still modeled as correlated (e.g., when coding a categorical predictor; see the `mixed` function of the **afex** package by Singmann et al., 2015, for a way to avoid this behavior).

While intuitive and visually appealing, the classic **lme4** syntax is not flexible enough to specify the more complex models supported by **brms**. In non-linear or distributional models, multiple parameters are predicted, each having their own population and group-level effects. Hence, multiple formulas are necessary to specify such models.[1] Specifying group-level effects of the same grouping factor to be correlated *across* formulas becomes complicated. The solution implemented in **brms** (and currently unique to it) expands the `|` operator into `|<ID>|`, where `<ID>` can be any value. Group-level terms with the same `ID` will then be modeled as correlated if they share same grouping factor(s). For instance, if the terms `(x1|ID|g1)` and `(x2|ID|g1)` appear somewhere in the same or different formulas passed to **brms**, they will be modeled as correlated.

Further extensions of the classical **lme4** syntax refer to the `group` component. In the **lme4** version, the `group` component is rather limited in its flexibility since only variable names combined by `:` or `/` are supported. We propose two extensions of this syntax.

The first extension concerns the `group` element. We propose that `group` can generally be split up in its terms so that, for example, `(1 | g1 + g2)` is expanded to `(1 | g1) + (1 | g2)`. This is fully consistent with the way `/` is handled and provides a natural generalization to the existing syntax.

The second extension concerns special grouping structures that cannot currently be expressed by simply combining grouping variables. For instance, multi-membership models cannot be expressed this way. To overcome this limitation, we propose wrapping terms in `group` within special functions that allow specifying alternative grouping structures: `(gterms | fun(group))`. In **brms**, there are currently two such functions implemented: `gr` for the default behavior, and `mm` for multi-membership terms. To be compatible with the original syntax and to keep formulas short, `gr` is automatically added internally if no function is specified.

While some non-linear relationships, such as quadratic relationships, can be expressed within the basic R formula syntax, more complicated ones cannot. Thus, we have made it possible in **brms** to specify non-linear predictor terms. Similar to the implementation in **nlme**, our extension is fully compatible with the extended multilevel syntax described above. Suppose, for instance, we want to model the non-linear growth curve

$$y = b_1(1 - \exp(-(x/b_2)^{b_3})$$

between y and x with parameters b_1, b_2, and b_3 (see Example 3 in this paper for an implementation of this model with real data). Furthermore, we want all three parameters to vary by a grouping variable g and wish to model the group-level effects as correlated. Additionally, b_1 should be predicted by a covariate z. We can express this in **brms** using multiple formulas, one for the non-linear model itself and one per non-linear parameter:

```
y ~ b1 * (1 - exp(-(x / b2) ^ b3)
b1 ~ z + (1|ID|g)
b2 ~ (1|ID|g)
b3 ~ (1|ID|g)
```

The first formula will not be evaluated using standard R formula parsing, but is instead taken literally. In contrast, the formulas for the non-linear parameters will be evaluated in the usual way and are compatible with all terms supported by **brms**. Note that we have used the above described ID-syntax to model group-level effects correlated across formulas.

There are other syntax extensions implemented in **brms** that do not directly target grouping terms. First, there are terms formally included in the `pterms` part that are handled separately. The most prominent examples are smooth terms specified using the `s` and `t2` functions of the **mgcv** package (Wood, 2011). Other examples are category specific effects `cs`, monotonic effects `mo`, noise-free effects

[1] Actually, it is possible to specify multiple model parts within one formula using interactions terms, as implemented in **MCMCglmm** (Hadfield, 2010). However, this syntax is limited in flexibility and requires a rather deep understanding of the way R parses formulas, thus often being confusing to users.

me, or Gaussian process terms gp. The first is explained in Bürkner (2017), while the latter three are documented in help(brmsformula). Internally, these terms are extracted from pterms and not included in the construction of the population-level design matrix.

Second, since the | operator is unused on the left-hand side of ∼ in the formula, additional information on the response variable may be specified via

```
response | aterms ~ <predictor terms>
```

The aterms part may contain multiple terms of the form fun(<variable>), each separated by + and providing special information on the response variable, including weighting observations, providing known standard errors for meta-analysis, or modeling censored or truncated data. As it is not the main topic of the present paper, we refer users to help("brmsformula") and help("addition-terms") for more details.

To set up the model formulas and combine them into one object, **brms** defines the brmsformula function, or bf for short. Its output can then be passed to the parse_bf function, which splits up the formulas into separate parts and prepares them for the generation of design matrices and related data. Other packages may re-use these functions in their own routines, making it easier to offer support for the above described multilevel syntax.

Examples

The idea of **brms** is to provide one unified framework for multilevel regression models in R. The above described formula syntax and all of its variations can be applied in combination with all response distributions supported by **brms**. Currently about 35 response distributions are supported; see help("brmsfamily") and vignette("brms_families") for an overview.

In this section, we will discuss four examples in detail, each focusing on certain aspects of the syntax. These examples are chosen to provide a broad overview of the modeling options. The first is about the number of fish caught by different groups of people. It does not actually contain any multilevel structure, but helps to understand how to set up formulas for different model parts. The second example is about housing rents in Munich. We model the data using splines and a distributional regression approach. The third example is about cumulative insurance loss payments across several years, which is fitted using a rather complex non-linear multilevel model. Finally, the fourth example is about the performance of school children who change school during the year, thus requiring a multi-membership model.

In addition to the four examples, we wish to discuss briefly a few more modeling options. First, **brms** allows fitting of so-called phylogenetic models. These models are relevant in evolutionary biology when data of many species are analyzed at the same time. Species are not independent if they come from the same phylogenetic tree, implying that different levels of the same grouping-factor (i.e., species) are likely correlated. See vignette("brms_phylogenetics") for an example. Second, there is a canonical way to handle ordinal predictors without falsely assuming they are either categorical or continuous. We call them monotonic effects and discuss them in vignette("brms_monotonic"). Last but not least, it is possible to account for measurement error in both response and predictor variables. Although often ignored in applied regression modeling (Westfall and Yarkoni, 2016), measurement error is common in all scientific fields that employ observational data. There is no vignette yet for this topic, but one will be added in the future. In the meantime, help("brmsformula") is the best place to start learning about such models as well as about other details of the **brms** formula syntax.

Example 1: Catching fish

An important application of the distributional regression framework of **brms** are zero-inflated and hurdle models. These models are helpful whenever there are more zeros in the response variable than one would naturally expect. Here, we consider an example dealing with the number of fish caught by various groups of people. On the UCLA website (https://stats.idre.ucla.edu/stata/dae/zero-inflated-poisson-regression), the data are described as follows: "The state wildlife biologists want to model how many fish are being caught by fishermen at a state park. Visitors are asked how long they stayed, how many people were in the group, were there children in the group and how many fish were caught. Some visitors do not fish, but there is no data on whether a person fished or not. Some visitors who did fish did not catch any fish so there are excess zeros in the data because of the people that did not fish."

```
zinb <- read.csv("http://stats.idre.ucla.edu/stat/data/fish.csv")
zinb$camper <- factor(zinb$camper, labels = c("no", "yes"))
head(zinb)
```

```
  nofish livebait camper persons child         xb         zg count
1      1        0     no       1     0 -0.8963146  3.0504048     0
2      0        1    yes       1     0 -0.5583450  1.7461489     0
3      0        1     no       1     0 -0.4017310  0.2799389     0
4      0        1    yes       2     1 -0.9562981 -0.6015257     0
5      0        1     no       1     0  0.4368910  0.5277091     1
6      0        1    yes       4     2  1.3944855 -0.7075348     0
```

For predictors, we choose the number of people per group, the number of children, and whether or not the group consists of campers. Many groups may not catch any fish because they do not try and so we fit a zero-inflated Poisson model. For now, we assume a constant zero-inflation probability across observations.

```
fit_zinb1 <- brm(count ~ persons + child + camper, data = zinb,
                 family = zero_inflated_poisson("log"))
```

The model is readily summarized via

```
summary(fit_zinb1)
```

```
 Family: zero_inflated_poisson (log)
Formula: count ~ persons + child + camper
   Data: zinb (Number of observations: 250)
Samples: 4 chains, each with iter = 2000; warmup = 1000; thin = 1;
         total post-warmup samples = 4000
   WAIC: Not computed

Population-Level Effects:
          Estimate Est.Error l-95% CI u-95% CI Eff.Sample Rhat
Intercept    -1.01      0.17    -1.34    -0.67       2171    1
persons       0.87      0.04     0.79     0.96       2188    1
child        -1.36      0.09    -1.55    -1.18       1790    1
camper        0.80      0.09     0.62     0.98       2950    1

Family Specific Parameters:
   Estimate Est.Error l-95% CI u-95% CI Eff.Sample Rhat
zi     0.41      0.04     0.32     0.49       2409    1

Samples were drawn using sampling(NUTS). For each parameter, Eff.Sample
is a crude measure of effective sample size, and Rhat is the potential
scale reduction factor on split chains (at convergence, Rhat = 1).
```

Figure 1 shows the graphical summary output by

```
marginal_effects(fit_zinb1)
```

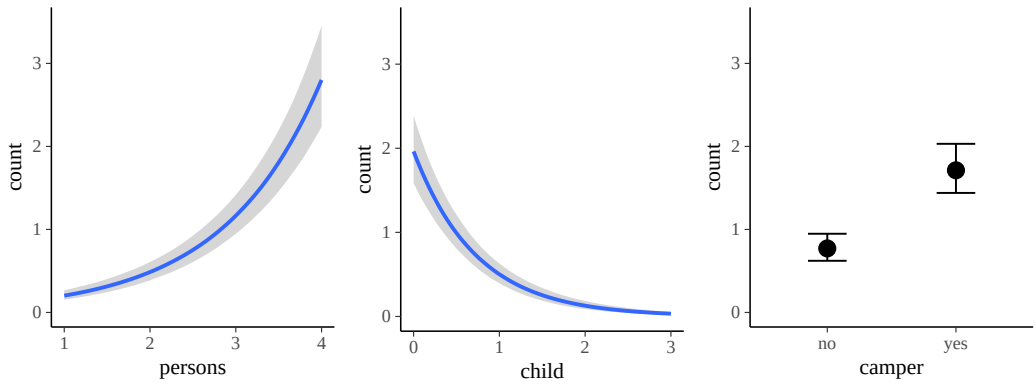


Figure 1: Marginal effects plots of the `fit_zinb1` model.

According to the parameter estimates, larger groups catch more fish, campers catch more fish than non-campers, and groups with more children catch less fish. The zero-inflation probability `zi` is pretty large with a mean of 41%. Note that the probability of catching no fish is actually higher than 41%, since parts of this probability are already modeled by the Poisson distribution itself (hence the name

zero-*inflation*). If the user wants to treat all zeros as originating from a separate process, hurdle models are also supported by **brms** (not shown here).

Since we expect groups with more children to avoid fishing, we next try to predict the zero-inflation probability using the number of children. From a purely statistical perspective, zero-inflated and hurdle distributions are a mixture of two processes. Predicting both parts of the model is natural and often very reasonable to make full use of the data.

```
fit_zinb2 <- brm(bf(count ~ persons + child + camper, zi ~ child),
                 data = zinb, family = zero_inflated_poisson())
```

To transform the linear predictor `zi` into a probability, **brms** applies the logit-link, which takes values in $[0, 1]$ and returns values on the real line. Thus, it allows the transition between probabilities and linear predictors.

```
summary(fit_zinb2)
```

```
 Family: zero_inflated_poisson (log)
Formula: count ~ persons + child + camper
         zi ~ child
   Data: zinb (Number of observations: 250)
Samples: 4 chains, each with iter = 2000; warmup = 1000; thin = 1;
         total post-warmup samples = 4000
   WAIC: Not computed

Population-Level Effects:
             Estimate Est.Error l-95% CI u-95% CI Eff.Sample Rhat
Intercept       -1.07      0.18    -1.43    -0.73       2322    1
persons          0.89      0.05     0.80     0.98       2481    1
child           -1.17      0.10    -1.37    -1.00       2615    1
camper           0.78      0.10     0.60     0.96       3270    1
zi_Intercept    -0.95      0.27    -1.52    -0.48       2341    1
zi_child         1.21      0.28     0.69     1.79       2492    1

Samples were drawn using sampling(NUTS). For each parameter, Eff.Sample
is a crude measure of effective sample size, and Rhat is the potential
scale reduction factor on split chains (at convergence, Rhat = 1).
```

According to the model, trying to fish with children decreases the overall number fish caught (as implied by the Poisson part of the model) and decreases the chance of catching any fish (as implied by the zero-inflation part).

Next we demonstrate how to comparing model fit via leave-one-out cross validation as implemented in the **loo** package (Vehtari et al., 2016a,b):

```
loo(fit_zinb1, fit_zinb2)
```

```
                       LOOIC     SE
fit_zinb1             1639.52 363.30
fit_zinb2             1621.35 362.39
fit_zinb1 - fit_zinb2   18.16  15.71
```

The example above shows that the second model (using the number of children as a predictor) has better fit. However, when considering the standard error of the `LOOIC` difference, improvement in model fit is apparently modest and not substantial. More examples of distributional models can be found in `vignette("brms_distreg")`.

Example 2: Housing rents

Fahrmeir et al. (2013) use an example about housing rents in Munich. The data contains information on roughly 3000 apartments, and includes variables such as the absolute rent (`rent`), rent per square meter (`rentsqm`), size of the apartment (`area`), construction year (`yearc`), and the district in Munich where the apartment is located (`district`). The data can be found in the **gamlss.data** package (Stasinopoulos and Rigby, 2016):

```
data("rent99", package = "gamlss.data")
head(rent99)
```

```
      rent   rentsqm area yearc location bath kitchen cheating district
1 109.9487  4.228797   26  1918        2    0       0        0      916
2 243.2820  8.688646   28  1918        2    0       0        1      813
3 261.6410  8.721369   30  1918        1    0       0        1      611
4 106.4103  3.547009   30  1918        2    0       0        0     2025
5 133.3846  4.446154   30  1918        2    0       0        1      561
6 339.0256 11.300851   30  1918        2    0       0        1      541
```

Here, we wish to predict the rent per square meter using the following explanatory variables: size of the apartment, the construction year, and the district in which the apartment is located. As the effect of apartment size and construction year predictors is of unknown non-linear form, we model these variables using a bivariate tensor spline (Wood et al., 2013). The district is accounted for using a varying intercept.

```
fit_rent1 <- brm(rentsqm ~ t2(area, yearc) + (1|district), data = rent99,
                 chains = 2, cores = 2)
```

We fit the model using just two chains instead of the default of four chains on two processor cores to reduce the model fitting time for the purpose of the present paper. In general, using the default option of four chains (or more) is recommended.

```
summary(fit_rent1)
```

```
 Family: gaussian(identity)
Formula: rentsqm ~ t2(area, yearc) + (1 | district)
   Data: rent99 (Number of observations: 3082)
Samples: 2 chains, each with iter = 2000; warmup = 1000; thin = 1;
         total post-warmup samples = 2000
    ICs: LOO = NA; WAIC = NA; R2 = NA

Smooth Terms:
                   Estimate Est.Error l-95% CI u-95% CI Eff.Sample Rhat
sds(t2areayearc_1)     4.93      2.32     1.61    10.77       1546 1.00
sds(t2areayearc_2)     5.78      2.87     1.58    13.15       1175 1.00
sds(t2areayearc_3)     8.09      3.19     3.66    16.22       1418 1.00

Group-Level Effects:
~district (Number of levels: 336)
              Estimate Est.Error l-95% CI u-95% CI Eff.Sample Rhat
sd(Intercept)     0.60      0.06     0.48     0.73        494 1.01

Population-Level Effects:
              Estimate Est.Error l-95% CI u-95% CI Eff.Sample Rhat
Intercept         7.80      0.11     7.59     8.02       2000 1.00
t2areayearc_1    -1.00      0.09    -1.15    -0.83       2000 1.00
t2areayearc_2     0.75      0.17     0.43     1.09       2000 1.00
t2areayearc_3    -0.07      0.16    -0.40     0.24       1579 1.00

Family Specific Parameters:
      Estimate Est.Error l-95% CI u-95% CI Eff.Sample Rhat
sigma     1.95      0.03     1.90     2.01       2000 1.00

Samples were drawn using sampling(NUTS). For each parameter, Eff.Sample
is a crude measure of effective sample size, and Rhat is the potential
scale reduction factor on split chains (at convergence, Rhat = 1).
```

For models including splines, the output of `summary` gives limited information. First, the credible intervals of the standard deviations of the coefficients forming the splines (under `'Smooth Terms'`) are sufficiently far away from zero to indicate non-linearity in the combined effect of `area` and `yearc`. Second, even after controlling for these predictors, rent per square meter will still vary considerably by district, as visible under `'Group-Level Effects'` in the output. To further understand the effect of the predictor, we apply graphical methods:

```
marginal_effects(fit_rent1, surface = TRUE)
```

In Figure 2, the marginal effect of each predictor is displayed with the other predictor fixed at its mean. In Figure 3, the combined effect demonstrates the interaction between the variables. In particular, housing rents appear to be highest for small and relatively new apartments.

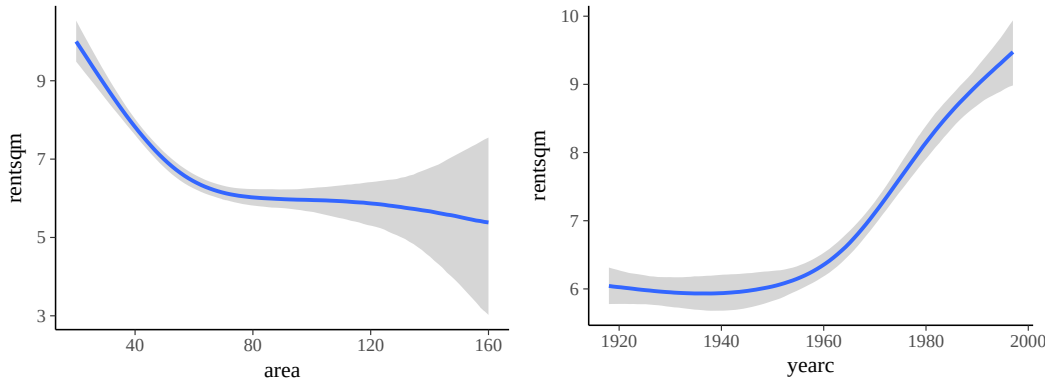


Figure 2: Marginal effects plots of the `fit_rent1` model for single predictors.

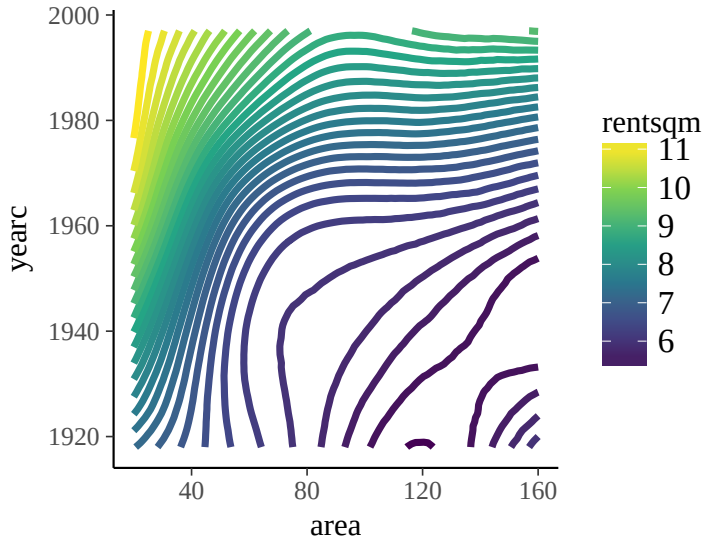


Figure 3: Surface plot of the `fit_rent1` model for the combined effect of `area` and `yearc`.

In the above example, we only consider the mean of the response distribution to vary by `area` and `yearc`, but this may not be a reasonable assumption since the variance might vary with these variables as well. Accordingly, we fit splines and the effects of district for both the location and scale parameter, which is called `sigma` in Gaussian models.

```
bform <- bf(rentsqm ~ t2(area, yearc) + (1|ID1|district),
            sigma ~ t2(area, yearc) + (1|ID1|district))
fit_rent2 <- brm(bform, data = rent99, chains = 2, cores = 2)
```

If not otherwise specified, `sigma` is predicted on the log-scale to ensure it is positive no matter how the predictor term looks. Instead of `(1|district)` as in the previous model, we now use `(1|ID1|district)` in both formulas, which results in modeling the varying intercepts of both model parts as correlated (see the description of the ID-syntax above). The group-level part of the `summary` output looks like the following:

```
Group-Level Effects:
~district (Number of levels: 336)
                            Estimate Est.Error l-95% CI u-95% CI Eff.Sample Rhat
sd(Intercept)                   0.60      0.06     0.49     0.73        744 1.00
sd(sigma_Intercept)             0.11      0.02     0.06     0.15        751 1.00
cor(Intercept,sigma_Intercept)  0.72      0.17     0.35     0.98        648 1.00
```

As visible from the positive correlation of the intercepts, districts with overall higher rent per square meter also have higher variance in the outcome.

Lastly, we want to turn our attention to the splines. While `marginal_effects` is used to visualize effects of predictors on the expected response, `marginal_smooths` is used to show just the spline parts of the model:

```
marginal_smooths(fit_rent2)
```

The plot on the left-hand side of Figure 4 resembles the one in Figure 3, but the scale is different since only the spline is plotted. The right-hand side of 4 shows the spline for `sigma`. Since we apply the log-link on `sigma` by default, the spline is on the log-scale as well. As visible in the plot, the variation in the rent per square meter is highest for relatively small and old apartments, while the variation is smallest for medium to large apartments build around the 1960s.

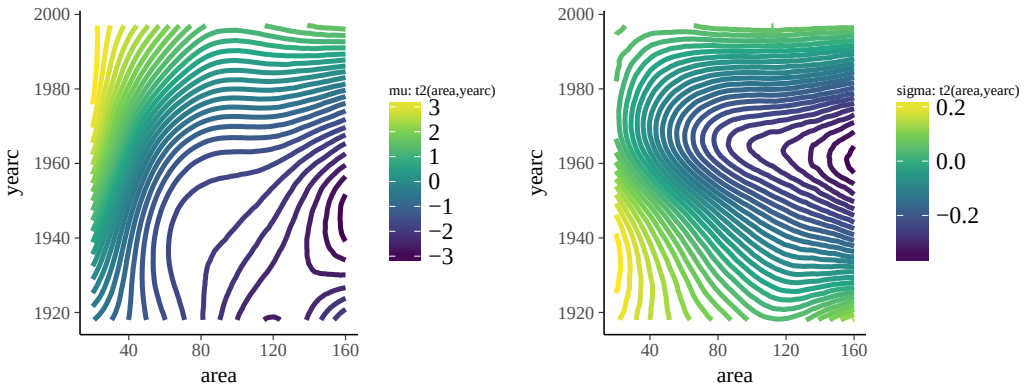


Figure 4: Plots showing the smooth terms of the `fit_rent2` model.

Example 3: Insurance loss payments

On his blog, Markus Gesmann predicts the growth of cumulative insurance loss payments over time, originated from different origin years (see http://www.magesblog.com/2015/11/loss-developments-via-growth-curves-and.html). We will use a slightly simplified version of his model for demonstration purposes:

$$cum_{AY,dev} \sim N(\mu_{AY,dev}, \sigma)$$

$$\mu_{AY,dev} = ult_{AY}\left(1 - \exp\left(-\left(\frac{dev}{\theta}\right)^{\omega}\right)\right)$$

The cumulative insurance payments *cum* will grow over time, and we model this dependency using the variable *dev*. Furthermore, ult_{AY} is the (to be estimated) ultimate loss of accident each year. It constitutes a non-linear parameter in our framework along with the parameters θ and ω, which are responsible for the growth of the cumulative loss and are for now assumed to be the same across years. We load the data

```
url <- paste0("https://raw.githubusercontent.com/mages/",
              "diesunddas/master/Data/ClarkTriangle.csv")
loss <- read.csv(url)
head(loss)
```

```
    AY dev      cum
1 1991   6  357.848
2 1991  18 1124.788
3 1991  30 1735.330
4 1991  42 2182.708
5 1991  54 2745.596
6 1991  66 3319.994
```

and translate the proposed model into a non-linear **brms** model.

```
nlform <- bf(cum ~ ult * (1 - exp(-(dev / theta)^omega)),
             ult ~ 1 + (1|AY), omega ~ 1, theta ~ 1, nl = TRUE)

nlprior <- c(prior(normal(5000, 1000), nlpar = "ult"),
             prior(normal(1, 2), nlpar = "omega"),
             prior(normal(45, 10), nlpar = "theta"))

fit_loss1 <- brm(formula = nlform, data = loss, family = gaussian(),
                 prior = nlprior, control = list(adapt_delta = 0.9))
```

In the above functions calls, quite a few things happen. The formulas are wrapped in `bf` to combine them into one object. The first formula specifies the non-linear model and we set argument `nl = TRUE` so that **brms** takes this formula literally, instead of using standard R formula parsing. We specify one additional formula per non-linear parameter to: (a) clarify which variables are covariates and which are parameters; and (b) specify the predictor term for the parameters. We estimate a group-level effect for accident year (variable `AY`) and ultimate loss `ult`. This specification nicely shows how a non-linear parameter is actually a placeholder for a linear predictor, which in the case of `ult`, contains only a varying intercept for year. Both `omega` and `theta` are assumed to be constant across observations so we just fit a population-level intercept.

Priors on population-level effects are required and are mandatory to ensure identifiability for the present model. Otherwise, we may observe that different Markov chains converge to different parameter regions as multiple posterior distribution are equally plausible. Setting prior distributions is a difficult task especially in non-linear models because it requires some experience and knowledge both about the model that is being fitted and about the data at hand. Additionally, there is more to keep in mind to optimize the sampler's performance: First, using non- or weakly-informative priors in non-linear models often leads to problems even if the model is generally identified. For instance, if a zero-centered and reasonably wide prior such as `normal(0,10000)` is set for `ult`, there is little information about `theta` and `omega` for samples of `ult` being close to zero, which may lead to convergence problems. Second, Stan works best when parameters are roughly on the same order of magnitude (Stan Development Team, 2017b). In the present example, `ult` is three orders larger than `omega`. Although the sampler seems to work quite well in this case, it may not work well for other models. One solution is to rescale parameters before model fitting. For the present example, one could downscale `ult` by replacing it with `ult * 1000` and correspondingly the `normal(5000,1000)` prior with `normal(5,1)`.

In the `control` argument we increase `adapt_delta` to get rid of a few divergent transitions (cf. Stan Development Team, 2017b; Bürkner, 2017). Again the model is summarized via

```
summary(fit_loss1)
```

```
 Family: gaussian (identity)
Formula: cum ~ ult * (1 - exp(-(dev / theta)^omega))
         ult ~ 1 + (1 | AY)
         omega ~ 1
         theta ~ 1
   Data: loss (Number of observations: 55)
Samples: 4 chains, each with iter = 2000; warmup = 1000; thin = 1;
         total post-warmup samples = 4000
   WAIC: Not computed

Group-Level Effects:
~AY (Number of levels: 10)
                  Estimate Est.Error l-95% CI u-95% CI Eff.Sample Rhat
sd(ult_Intercept)   745.74    231.31   421.05  1306.04        916    1

Population-Level Effects:
                Estimate Est.Error l-95% CI u-95% CI Eff.Sample Rhat
ult_Intercept    5273.70    292.34  4707.11  5852.28        798    1
omega_Intercept     1.34      0.05     1.24     1.43       2167    1
theta_Intercept    46.07      2.09    42.38    50.57       1896    1

Family Specific Parameters:
      Estimate Est.Error l-95% CI u-95% CI Eff.Sample Rhat
sigma   139.93     15.52    113.6   175.33       2358    1

Samples were drawn using sampling(NUTS). For each parameter, Eff.Sample
is a crude measure of effective sample size, and Rhat is the potential
scale reduction factor on split chains (at convergence, Rhat = 1).
```

as well as

```
marginal_effects(fit_loss1)
```

which yields Figure 5.

We can also visualize the cumulative insurance loss over time separately for each year, as in Figure 6.

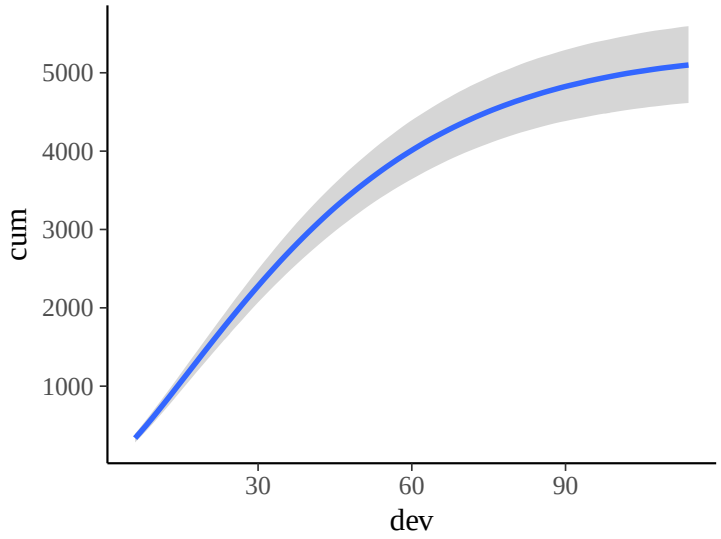


Figure 5: Marginal effects plots of the `fit_loss1` model.

```
conditions <- data.frame(AY = unique(loss$AY))
rownames(conditions) <- unique(loss$AY)
me_year <- marginal_effects(fit_loss1, conditions = conditions,
                            re_formula = NULL, method = "predict")
plot(me_year, ncol = 5, points = TRUE)
```

It is evident that there is some variation in cumulative loss across accident years, for instance due to natural disasters happening only in certain years. Further, we see that the uncertainty in the predicted cumulative loss is larger for later years in which there are fewer available data points.

In the above model, we considered `omega` and `delta` to be constant across years, which may not necessarily be true. We can easily investigate this by fitting varying intercepts for all three non-linear parameters also estimating group-level correlation using the `ID` syntax:

```
nlform2 <- bf(cum ~ ult * (1 - exp(-(dev / theta)^omega)),
              ult ~ 1 + (1|ID1|AY), omega ~ 1 + (1|ID1|AY),
              theta ~ 1 + (1|ID1|AY), nl = TRUE)

fit_loss2 <- update(fit_loss1, formula = nlform2,
                    control = list(adapt_delta = 0.90))
```

We could have also specified all predictor terms more conveniently with one formula as

```
ult + omega + theta ~ 1 + (1|ID1|AY)
```

because the structure of the predictor terms is identical.

To compare model fit, we perform leave-one-out cross-validation.

```
loo(fit_loss1, fit_loss2)
```

```
                      LOOIC    SE
fit_loss1             715.44 19.24
fit_loss2             720.60 19.85
fit_loss1 - fit_loss2  -5.15  5.34
```

Since smaller values indicate better expected out-of-sample predictions and thus a better model fit, the simpler model that has only a varying intercept over parameter `ult` is preferred. This result may not be overly surprising, given that modeling three varying intercepts and three group-level correlations is probably overkill for data containing only 55 observations. Nevertheless, it nicely demonstrates how to apply the `ID` syntax in practice. More examples of non-linear models can be found in `vignette("brms_nonlinear")`.

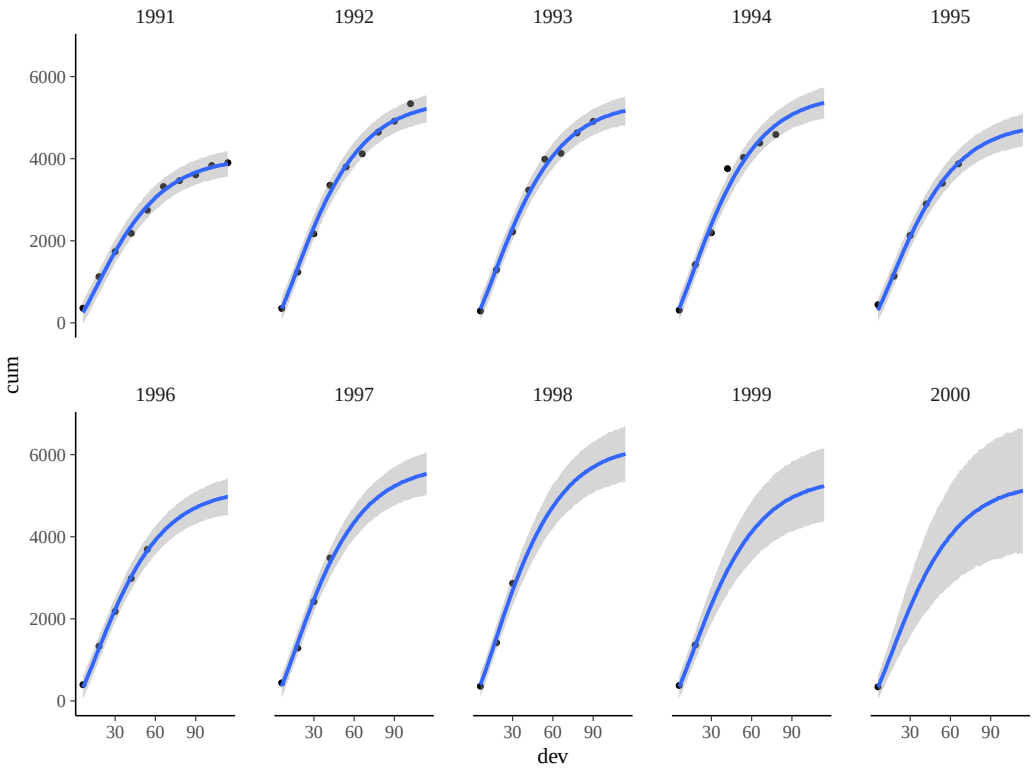


Figure 6: Marginal effects plots of the `fit_loss1` model separately for each accident year.

Example 4: Performance of school children

Suppose that we want to predict the performance of students in the final exams at the end of the year. There are many variables to consider, but one important factor will clearly be school membership. Schools might differ in the ratio of teachers and students, the general quality of teaching, in the cognitive ability of the students they draw, or other unobserved factors that may induce dependency among students in the same school. Thus, it is advisable to apply multilevel modeling techniques by including school membership as a group-level term. We should account for class membership and other levels of the educational hierarchy as well, but for the purposes of the present example, we will focus on schools only.

Usually, accounting for school membership is straight-forward; we simply add a varying intercept to the formula for each school with `(1 | school)`. However, a non-negligible number of students might change schools during the year, which would result in a situation where one student is a member of multiple schools, necessitating a multi-membership model. Specifying such a model not only requires information on the different schools students attend during the year, but also the amount of time spent at each school. The time spent can be used to weight the influence each school has on its students, since more time attending a school will likely result in greater influence. For now, let us assume that students change schools maximally once a year and spend equal time at each school. We will later see how to relax these assumptions.

Real educational data are usually relatively large and complex so that we simulate our own data for the purpose of this example. We simulate 1000 students partitioned into 10 schools of 100 students each. We model 10% of students as changing schools.

```
data_mm <- sim_multi_mem(nschools = 10, nstudents = 1000, change = 0.1)
head(data_mm)
```

```
  s1 s2  w1  w2        y
1  8  9 0.5 0.5 16.27422
2 10  9 0.5 0.5 18.71387
3  5  3 0.5 0.5 23.65319
4  3  5 0.5 0.5 22.35204
5  5  3 0.5 0.5 16.38019
6 10  6 0.5 0.5 17.63494
```

The code of function `sim_multi_mem` can be found in the online supplement for this paper. To better illustrate the multimembership condition, students changing schools appear in the first rows of the data. Data of students being only at a single school looks as follows:

```
data_mm[101:106, ]
```

```
    s1 s2  w1  w2         y
101  2  2 0.5 0.5 27.247851
102  9  9 0.5 0.5 24.041427
103  4  4 0.5 0.5 12.575001
104  2  2 0.5 0.5 21.203644
105  4  4 0.5 0.5 12.856166
106  4  4 0.5 0.5  9.740174
```

Although school variables `s1` and `s2` are identical, we still have to specify both in order to pass the data appropriately. Incorporating no other predictors for simplicity, a multi-membership model is specified as

```
fit_mm <- brm(y ~ 1 + (1 | mm(s1, s2)), data = data_mm)
```

The only new syntax element is that multiple grouping factors `s1` and `s2` are wrapped in `mm`. Everything else remains exactly the same. Note that we did not specify the relative weights of schools for each student and thus equal weights are assumed by default.

```
summary(fit_mm)
```

```
 Family: gaussian (identity)
Formula: y ~ 1 + (1 | mm(s1, s2))
   Data: data_mm (Number of observations: 1000)
Samples: 4 chains, each with iter = 2000; warmup = 1000; thin = 1;
         total post-warmup samples = 4000
   WAIC: Not computed

Group-Level Effects:
~mms1s2 (Number of levels: 10)
              Estimate Est.Error l-95% CI u-95% CI Eff.Sample Rhat
sd(Intercept)     2.76      0.82     1.69     4.74        682 1.01

Population-Level Effects:
          Estimate Est.Error l-95% CI u-95% CI Eff.Sample Rhat
Intercept       19      0.93    17.06     20.8        610    1

Family Specific Parameters:
      Estimate Est.Error l-95% CI u-95% CI Eff.Sample Rhat
sigma     3.58      0.08     3.43     3.75       2117    1

Samples were drawn using sampling(NUTS). For each parameter, Eff.Sample
is a crude measure of effective sample size, and Rhat is the potential
scale reduction factor on split chains (at convergence, Rhat = 1).
```

With regard to the assumptions made in the above example, it is unlikely that all children who change schools stay in both schools equally long. To relax this assumption, we have to specify weights. First, we amend the simulated data to contain non-equal weights `w1` and `w2` for students changing schools:

```
data_mm[1:100, "w1"] <- runif(100, 0, 1)
data_mm[1:100, "w2"] <- 1 - data_mm[1:100, "w1"]
head(data_mm)
```

```
  s1 s2        w1         w2        y
1  8  9 0.3403258 0.65967423 16.27422
2 10  9 0.1771435 0.82285652 18.71387
3  5  3 0.9059811 0.09401892 23.65319
4  3  5 0.4432007 0.55679930 22.35204
5  5  3 0.8052026 0.19479738 16.38019
6 10  6 0.5610243 0.43897567 17.63494
```

Incorporating these weights into the model is straightforward:

```
fit_mm2 <- brm(y ~ 1 + (1 | mm(s1, s2, weights = cbind(w1, w2))),
               data = data_mm)
```

The summary output is similar to the previous, so we omit it for parsimony. The second assumption that students change schools only once a year, may also easily be relaxed by providing more than two grouping factors in `mm` (i.e., `mm(s1,s2,s3)`).

Comparison between packages

Over the years, many R packages have offered implementations for MLMs, each being more or less general in their supported models. In Bürkner (2017), I compared **brms** with **lme4** (Bates et al., 2015), **MCMCglmm** (Hadfield, 2010), **rstanarm** (Stan Development Team, 2017a), and **rethinking** (McElreath, 2017). Since then, quite a few new features have been added to **brms** and **rstanarm**. Accordingly, in the present paper, I will update these comparisons, focusing on **brms**, **rstanarm**, and **MCMCglmm**. While **brms** and **rstanarm** are both based on the Stan probabilistic programming language, **MCMCglmm** implements its own custom MCMC algorithm. Modeling options and other important information for these packages are summarized in Table 1 and will be discussed in detail below.

Regarding general model classes, all three packages support common types, including linear, count, and certain survival models. Currently, **brms** and **MCMCglmm** provide more flexibility when modeling categorical and ordinal data. Additionally, they offer support for zero-inflated and hurdle models to account for zeros in the data (see Example 1 above). For survival or time-to-event models, **rstanarm** offers great flexibility via the `stan_jm` function, which allows for complex association structures between time-to-event data and one or more models of longitudinal covariates (for details see `https://cran.r-project.org/web/packages/rstanarm/vignettes/jm.html`). Model classes currently specific to **brms** are robust linear models using Student's *t*-distribution (family `student`), response time models via the exponentially modified Gaussian distribution (family `exgaussian`), and the Wiener diffusion model (family `wiener`). The latter simultaneously models dichotomous outcomes and their corresponding response times (for a detailed example see `http://singmann.org/wiener-model-analysis-with-brms-part-i/`).

All three packages offer many additional modeling options, with **brms** currently having the greatest flexibility (see Table 1 for a summary). Moreover, the packages differ in the general framework in which they are implemented: **brms** and **MCMCglmm** each come with a single model fitting function (`brm` and `MCMCglmm` respectively) through which all of their models can be specified. Further, their framework allows seamless combination of most modeling options in the same model. In contrast, the approach of **rstanarm** is to emulate existing functions of other packages, which has the advantage of an easier transition between classical and Bayesian models. Although the syntax used to specify models is most similar to that of a classical model, `rstanarm` comes with the main disadvantage that many modeling options cannot be used in combination in the same model.

Information criteria are available in all three packages. The advantages of WAIC and LOO implemented in **brms** and **rstanarm** are their less restrictive assumptions, and that their standard errors can be easily estimated to get a better sense of the uncertainty in the criteria. Comparing the options for prior distributions, **brms** offers a little more flexibility than **MCMCglmm** and **rstanarm**, as virtually any prior distribution can be applied on population-level effects as well as on the standard deviations of group-level effects. In addition, we believe that the way priors are specified in **brms** is more intuitive as it is directly evident which prior has been applied. In **brms**, Bayes factors are available via both Savage-Dickey ratios (Wagenmakers et al., 2010) and bridge-sampling (Gronau and Singmann, 2017), while **rstanarm** only allows for the latter option. For a detailed comparison with respect to sampling efficiency, see Bürkner (2017).

Conclusion

The present paper is meant to introduce R users and developers to the extended **lme4** formula syntax applied in **brms**. Four examples are presented to illustrate various features in the **brms** framework. For some of the more basic models that **brms** can fit, see Bürkner (2017). Many more examples can be found in the growing number of vignettes accompanying the package (see `vignette(package = "brms")` for an overview).

To date, **brms** is already one of the most flexible R packages when it comes to regression modeling and there are more features in the works (see `https://github.com/paul-buerkner/brms/issues` for the current list of issues). In addition to smaller, incremental updates, major planned updates include: (1) latent variables estimated via confirmatory factor analysis; and (2) extended multilevel and residual

	brms	**rstanarm**	**MCMCglmm**
Model classes			
Linear models	yes	yes	yes
Robust linear models	yes	no	no
Count data models	yes	yes	yes
Survival models	yes	yes[1]	yes
Response times models	yes	no	no
Beta models	yes	yes	no
Categorical models	yes	yes[2]	yes
Multinomial models	no	no	yes
Ordinal models	various	cumulative[2]	cumulative
Zero-inflated and hurdle models	yes	no	yes
Modeling options			
Variable link functions	various	various	no
Multilevel structures	yes	yes	yes
Multi-membership	yes	no	yes
Multivariate responses	yes	yes[3]	yes
Non-linear predictors	yes	limited[4]	no
Distributional regression	yes	no	no
Finite mixtures	yes	no	no
Splines (additive models)	yes	yes	yes
Gaussian Processes	yes	no	no
Temporal autocorrelation	yes	yes[2,5]	no
Spatial autocorrelation	yes	yes[2,5]	no
Monotonic effects	yes	no	no
Category specific effects	yes	no	no
Measurement error	yes	no	no
Weights	yes	yes	no
Offset	yes	yes	using priors
Censored data	yes	yes[1]	yes
Truncated data	yes	no	no
Customized covariances	yes	no	yes
Missing value imputation	yes	no	no
Bayesian specifics			
Population-level priors	flexible	flexible	normal
Group-level priors	normal	normal	normal
Covariance priors	flexible	restricted[6]	restricted[7]
Bayes factors	yes	yes[8]	no
Parallelization	yes	yes	no
Other			
Estimator	HMC, NUTS	HMC, NUTS	MH, Gibbs[9]
Information criterion	WAIC, LOO	WAIC, LOO	DIC
C++ compiler required	yes	no	no

Table 1: Comparison of the capabilities of the **brms**, **rstanarm**, and **MCMCglmm** packages. Notes: (1) Advanced functionality available via `stan_jm`. (2) No group-level terms and related functionality allowed. (3) Cannot be combined with other modeling options such as splines. (4) Functionality limited to linear Gaussian models and certain pre-specified non-linear functions. (5) Functionality available only on GitHub branches (https://github.com/stan-dev/rstanarm). (6) For details, see Hadfield (2010). (7) For details, see https://github.com/stan-dev/rstanarm/wiki/Prior-distributions. (8) Available via the **bridgesampling** package (Gronau and Singmann, 2017). (9) Estimator consists of a combination of both algorithms.

correlation structures. The **brms** package is under continuous develepment thanks to constructive user input. Suggestions for future features may be submitted to https://github.com/paul-buerkner/brms.

Acknowledgments

First of all, I would like to thank the Stan Development Team for creating the probabilistic programming language Stan, which is an incredibly powerful and flexible tool for performing full Bayesian inference. Without it, **brms** could not fit a single model. Furthermore, I want to thank Heinz Holling, Donald Williams, and Ruben Arslan for valuable comments on earlier versions of the paper. I would also like to thank the many users who reported bugs or had ideas for new features.

Bibliography

D. Bates, M. Mächler, B. Bolker, and S. Walker. Fitting linear mixed-effects models using **lme4**. *Journal of Statistical Software*, 67(1):1–48, 2015. [p395, 396, 408]

M. Betancourt. A conceptual introduction to hamiltonian monte carlo. *arXiv preprint*, 2017. URL https://arxiv.org/pdf/1701.02434.pdf. [p395]

M. Betancourt, S. Byrne, S. Livingstone, and M. Girolami. The geometric foundations of hamiltonian monte carlo. *arXiv preprint arXiv:1410.5110*, 2014. [p395]

H. Brown and R. Prescott. *Applied Mixed Models in Medicine*. John Wiley & Sons, 2015. [p395]

P.-C. Bürkner. **brms**: An R package for bayesian multilevel models using stan. *Journal of Statistical Software*, 2017. [p395, 396, 398, 404, 408]

B. Carpenter, A. Gelman, M. Hoffman, D. Lee, B. Goodrich, M. Betancourt, M. A. Brubaker, J. Guo, P. Li, and A. Ridell. Stan: A probabilistic programming language. *Journal of Statistical Software*, 2017. [p395]

E. Demidenko. *Mixed Models: Theory and Applications with R*. John Wiley & Sons, 2013. [p395]

S. Duane, A. D. Kennedy, B. J. Pendleton, and D. Roweth. Hybrid monte carlo. *Physics Letters B*, 195(2): 216–222, 1987. [p395]

L. Fahrmeir, T. Kneib, S. Lang, and B. Marx. *Regression: Models, Methods and Applications*. Springer-Verlag, 2013. [p400]

A. Gelman and J. Hill. *Data Analysis Using Regression and Multilevel/Hierarchical Models*. Cambridge University Press, 2006. [p395, 396]

A. Gelman, J. B. Carlin, H. S. Stern, and D. B. Rubin. *Bayesian Data Analysis*, volume 2. Taylor & Francis, 2014. [p395]

Q. F. Gronau and H. Singmann. *Bridgesampling: Bridge Sampling for Marginal Likelihoods and Bayes Factors*, 2017. URL https://CRAN.R-project.org/package=bridgesampling. R package version 0.4-0. [p408, 409]

J. D. Hadfield. Mcmc methods for multi-response generalized linear mixed models: The **MCMCglmm** R package. *Journal of Statistical Software*, 33(2):1–22, 2010. [p396, 397, 408, 409]

T. J. Hastie and R. J. Tibshirani. *Generalized Additive Models*, volume 43. CRC Press, 1990. [p395]

M. D. Hoffman and A. Gelman. The no-u-turn sampler: Adaptively setting path lengths in hamiltonian monte carlo. *The Journal of Machine Learning Research*, 15(1):1593–1623, 2014. [p395]

M. J. Lindstrom and D. M. Bates. Nonlinear mixed effects models for repeated measures data. *Biometrics*, pages 673–687, 1990. [p395]

R. McElreath. *Rethinking: Statistical Rethinking Course and Book Package*, 2017. URL https://github.com/rmcelreath/rethinking. R package version 1.59. [p408]

R. M. Neal. *Handbook of Markov Chain Monte Carlo*, volume 2, chapter MCMC Using Hamiltonian Dynamics. CRC Press, 2011. [p395]

J. Pinheiro and D. Bates. *Mixed-Effects Models in S and S-PLUS*. Springer-Verlag, 2006. [p395]

J. Pinheiro, D. Bates, S. DebRoy, D. Sarkar, and R Core Team. **nlme**: *Linear and Nonlinear Mixed Effects Models*, 2016. URL http://CRAN.R-project.org/package=nlme. R package version 3.1-124. [p396]

C. E. Rasmussen and C. K. I. Williams. *Gaussian Processes for Machine Learning*. Massachusetts Institute of Technology, 2006. [p395]

R. A. Rigby and D. M. Stasinopoulos. Generalized additive models for location, scale and shape. *Journal of the Royal Statistical Society C*, 54(3):507–554, 2005. [p395]

H. Singmann, B. Bolker, and J. Westfall. **afex**: *Analysis of Factorial Experiments*, 2015. URL https://CRAN.R-project.org/package=afex. R package version 0.15-2. [p397]

Stan Development Team. *Rstanarm: Bayesian Applied Regression Modeling via Stan.*, 2017a. URL http://mc-stan.org/. R package version 2.17.2. [p396, 408]

Stan Development Team. *Stan: A C++ Library for Probability and Sampling, Version 2.17.0*, 2017b. URL http://mc-stan.org/. [p404]

Stan Development Team. *Stan Modeling Language: User's Guide and Reference Manual*, 2017c. URL http://mc-stan.org/manual.html. [p395]

M. Stasinopoulos and B. Rigby. *Gamlss.data: GAMLSS Data*, 2016. URL https://CRAN.R-project.org/package=gamlss.data. R package version 5.0-0. [p400]

A. Vehtari, A. Gelman, and J. Gabry. **loo**: *Efficient Leave-One-Out Cross-Validation and WAIC for Bayesian Models.*, 2016a. URL https://github.com/stan-dev/loo. R package version 1.0.0. [p400]

A. Vehtari, A. Gelman, and J. Gabry. Practical bayesian model evaluation using leave-one-out cross-validation and waic. *Statistics and Computing*, pages 1–20, 2016b. [p400]

E.-J. Wagenmakers, T. Lodewyckx, H. Kuriyal, and R. Grasman. Bayesian hypothesis testing for psychologists: A tutorial on the savage–dickey method. *Cognitive psychology*, 60(3):158–189, 2010. [p408]

J. Westfall and T. Yarkoni. Statistically controlling for confounding constructs is harder than you think. *PloS one*, 11(3):e0152719, 2016. [p398]

C. K. Williams and C. E. Rasmussen. Gaussian processes for regression. In *Advances in Neural Information Processing Systems*, pages 514–520, 1996. [p396]

S. N. Wood. Stable and efficient multiple smoothing parameter estimation for generalized additive models. *Journal of the American Statistical Association*, 99(467):673–686, 2004. [p395]

S. N. Wood. Fast stable restricted maximum likelihood and marginal likelihood estimation of semi-parametric generalized linear models. *Journal of the Royal Statistical Society B*, 73(1):3–36, 2011. [p396, 397]

S. N. Wood, F. Scheipl, and J. J. Faraway. Straightforward intermediate rank tensor product smoothing in mixed models. *Statistics and Computing*, pages 1–20, 2013. [p401]

Paul-Christian Bürkner
Faculty of Psychology, University of Münster
Fliednerstr. 21, 48151 Münster
Germany
paul.buerkner@gmail.com

Support Vector Machines for Survival Analysis with R

by Césaire J. K. Fouodo, Inke R. König, Claus Weihs, Andreas Ziegler and Marvin N. Wright

Abstract This article introduces the R package **survivalsvm**, implementing support vector machines for survival analysis. Three approaches are available in the package: The regression approach takes censoring into account when formulating the inequality constraints of the support vector problem. In the ranking approach, the inequality constraints set the objective to maximize the concordance index for comparable pairs of observations. The hybrid approach combines the regression and ranking constraints in a single model. We describe survival support vector machines and their implementation, provide examples and compare the prediction performance with the Cox proportional hazards model, random survival forests and gradient boosting using several real datasets. On these datasets, survival support vector machines perform on par with the reference methods.

Introduction

Survival analysis considers time to an event as the dependent variable. For example, in the veteran's administration study (Kalbfleisch and Prentice, 2002), a clinical trial of lung cancer treatments, the dependent variable is time to death. The particularity of such a survival outcome is censoring, indicating that no event occurred during the study. For example, in the veteran's lung cancer study, some patients stayed alive until the end of the study such that time to death is censored. Because the time to event is unknown for censored observations, standard regression techniques cannot be used. The censoring indicator δ is set to be 0 or 1 for censored and not censored individuals, respectively. The main interest is to analyze the time $\boldsymbol{T} \in [0, \infty]$ until an event occurs, given covariates $\boldsymbol{X} \in \mathbb{R}^d$.

The most popular statistical approach for survival analysis is the Cox proportional hazards (PH) model, which is described in detail in standard textbooks (Kleinbaum and Klein, 2012; Lee and Wang, 2003). The most important advantage of the PH model is that it does not assume a particular statistical distribution of the survival time. However, the crucial assumption is that the effect of covariates on the survival variable is time independent. The hazards of two individuals are thus assumed to be proportional over time (proportional hazards assumption). The general form of the PH model is given by

$$h(t|\boldsymbol{X}) = h_0(t) \exp(\beta' \boldsymbol{X}) \,, \tag{1}$$

where $\beta' = (\beta_1, \beta_2, \ldots, \beta_d) \in \mathbb{R}^d$ are parameters to be estimated and $h_0(t)$ is the baseline hazard function, which is independent of the covariates. It does not need to be pre-specified or estimated in the model.

The proportional hazards assumption can easily be checked in one dimension ($\boldsymbol{X} \in \mathbb{R}$), but is difficult to verify when working in higher dimensions. Another difficulty when fitting a PH model occurs when the number of covariates d exceeds the number of individuals n, since it uses the partial likelihood for parameter estimation.

One alternative approach to the PH model is to use support vector machines (*SVMs*). SVMs were first proposed by Vapnik (1995) as a learning method for dichotomous outcomes. They are known for their good theoretic foundations and high classification accuracy on high dimensional classification problems (Cervantes et al., 2008; Hsu et al., 2003). The formulation of an SVM supposes a target variable $\boldsymbol{Y} \in \{-1, 1\}$ and covariates $\boldsymbol{X} \in \mathbb{R}^d$. Assuming that the two target classes are linearly separable, there exists a linear function $f(\boldsymbol{x}) = \psi x + b$ such that $\boldsymbol{y} f(\boldsymbol{x}) > 0$. The SVM task is to find the separating hyperplane $H(\psi, b) = \{\boldsymbol{x} \mid \langle \boldsymbol{x}, \psi \rangle + b = 0\}$ for the two classes with the maximal margin. The margin is defined as the smallest distance between any data point and the separating hyperplane. Data points lying at this distance from the separating hyperplane are called support vectors. They determine the margin, and must verify either $f(\boldsymbol{x}) = 1$ or $f(\boldsymbol{x}) = -1$. If the two classes are not linearly separable, misclassifications can be allowed. This is done by introducing slack variables $\xi_i \geq 0$, allowing but penalizing misclassifications. The slack variable for an individual i is defined as $\xi_i = |\boldsymbol{y}_i - f(\boldsymbol{x}_i)|$. Hence, it is $\xi_i = 0$ if i is correctly classified, $\xi_i < 1$ if it is inside the margin and on the correct side of the hyperplane, $\xi_i > 1$ if the data point lies on the wrong side of the hyperplane, and $\xi_i = 1$ if observation i lies exactly on the hyperplane. Figure 1 presents an illustration of the SVM approach for a two dimensional data set. Data points are grouped into the classes $y = 1$ (circles) and $y = -1$ (triangles). The filled and darkened data points are the support vectors defining the margin. The two red data points lying inside the margin are misclassified. They are penalized depending on their locations on the wrong or the correct side of the hyperplane, represented by the dashed line. All the other data points are correctly classified and therefore not penalized.

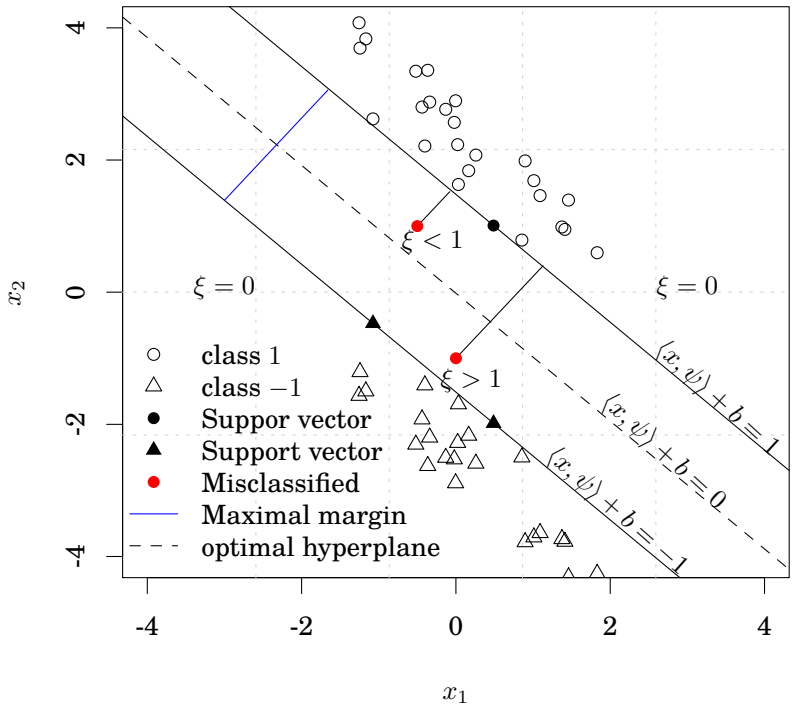


Figure 1: Illustration of the SVM approach with a two dimensional data set, split into two classes $y = 1$ (circles) and $y = -1$ (triangles). The filled data points are the support vectors defining the margin. Data points lying outside of the margin are correctly classified and not penalized ($\xi = 0$), contrary to the two misclassified and penalized red points lying inside the margin. These data points are penalized with the slack variables $\xi < 1$ or $\xi > 1$, depending on whether they lie on the correct or on the wrong side of the separating hyperplane (dashed line). This illustration is inspired by Rhode (2012) and the SVM problem was solved with the **kernlab** package (Karatzoglou et al., 2004).

To achieve the goal of the SVM approach described above, the following optimization problem is posed in primal space:

$$\begin{aligned} \min_{\psi,b,\xi} \quad & \tfrac{1}{2}\|\psi\|^2 + \gamma \sum_{i=1}^{n} \xi_i \\ \text{subject to} \quad & -\left(y_i(\langle x_i, \psi\rangle + b) + \xi_i - 1\right) \leq 0 \\ \text{and} \quad & \xi_i \geq 0,\ i = 1, \ldots, n\,, \end{aligned} \tag{2}$$

where ψ, b and the slack variables ξ_i are unknown, n is the number of individuals, and $\gamma > 0$ is a regularization parameter controlling the maximal margin and misclassification penalties. Instead of solving the optimization problem (2) in the primal space, the optimization problem is transformed to the dual problem, and the Lagrange function is optimized in the dual space. Details can be found in Bishop (2007). Eitrich and Lang (2006) recommends tuning the model to find the best parameter for regularization. In the SVM problem posed in (2), it is assumed that the two classes are linearly separable, but this is usually not the case. Russell and Norvig (2010) demonstrates that a set of n data points is always separable in an $n - 1$ dimensional space, which guarantees the existence of a higher dimensional space in which the two classes are linearly separable. This concept is described in more detail in the next section.

SVMs were extended to support vector regression, a variant for continuous outcomes by Vapnik (1998). More recently, several extensions to survival analysis were proposed. Van Belle et al. (2007) and Evers and Messow (2008) extended the formulation of the SVM problem with the aim to maximize the concordance index for comparable pairs of observations. This approach, also known as the ranking approach, was modified in Van Belle et al. (2008) to improve computational performance. Shivaswamy

et al. (2007) introduced a regression approach to survival analysis, based on the idea of support vector regression (Vapnik, 1998). Van Belle et al. (2011) combined the ranking and regression approaches in a single model to build the hybrid approach of SVMs for survival outcomes. The ranking approach, as proposed by Evers and Messow (2008), is implemented in the R package **survpack** available on the authors' website (Evers, 2009). The approaches of Van Belle et al. (2008) and Van Belle et al. (2011) are available in a Matlab toolbox (Yang and Pelckmans, 2014).

In the next section, we describe the three approaches for survival SVMs in detail. After that, we present the implementation of these methods in the R package **survivalsvm**. Finally, an application of survival SVMs on real data sets compares their prediction performance and runtime with established reference methods and other available implementations.

Survival support vector machines

Three approaches have been proposed to solve survival problems using SVMs: the regression (Shivaswamy et al., 2007), the ranking (Van Belle et al., 2007; Evers and Messow, 2008; Van Belle et al., 2008) and the hybrid approach (Van Belle et al., 2011). The regression approach is based on the support vector regression (SVR) (Vapnik, 1998) idea and aims at finding a function that estimates observed survival times as continuous outcome values $\boldsymbol{y}_i$ using covariates $\boldsymbol{x}_i$. A naïve variant of this approach is to ignore all censored observations and just solve the resulting SVR problem. Unfortunately, such a formulation implies a loss of information, since it does not take the particularity of survival data into account. Shivaswamy et al. (2007) improved this formulation by including the censoring in the SVM problem. For censored observations, the time to event after censoring is unknown and thus predictions greater than the censoring time do not have to be penalized. However, all survival predictions lower than the censoring time are penalized as usual. For not censored data, the exact survival times are known and, as in standard SVR, all survival predictions lower or greater than the observed survival time are penalized. Alternatively, survival predictions from censored and not censored data can be penalized differently, as proposed by Khan and Zubek (2008). However, this implies more parameters of regularization when formulating the survival SVR problem, increasing computation times (Van Belle et al., 2011). In the present work, we implement the survival SVR as proposed by Shivaswamy et al. (2007) and formulate the problem as follows:

$$\begin{aligned} \min_{\psi,b,\xi,\xi*} \quad & \tfrac{1}{2}\|\psi\|^2 + \gamma \sum_{i=1}^{n} (\xi_i + \xi_i^*) \\ \text{subject to} \quad & \boldsymbol{y}_i - \langle \psi, F(\boldsymbol{x}_i)\rangle - b \leq \xi_i \,, \\ & \delta_i(\langle \psi, F(\boldsymbol{x}_i)\rangle + b - \boldsymbol{y}_i) \leq \xi_i^* \\ \text{and} \quad & \xi_i, \xi_i^* \geq 0 \,, \end{aligned} \tag{3}$$

where $i = 1, \ldots, n$. δ is the censoring indicator and F is a function that maps observed covariates to the feature space. The feature space, compared with the original space in which data is observed, is a higher dimensional space in which the training data points are projected to be separated more easily by a hyperplane. More details about this concept can be found in Vapnik (1995). Since the feature space implies a higher dimensional space, the inner product $\langle \psi, F(\boldsymbol{x}_i)\rangle$ is calculated using a kernel function to reduce runtime (Vapnik, 1995). In the case of no censoring, i.e., $\delta_i = 1$, the inequality constraints in (3) are the same as in the classical SVR problem. However, if censoring occurs, the second constraint is reduced to $\xi_i^* \geq 0$.

The ranking approach considers survival analysis based on SVMs as a classification problem with an ordinal target variable (Herbrich et al., 1999). It aims at predicting risk ranks between individuals instead of estimating survival times (Van Belle et al., 2007; Evers and Messow, 2008). Suppose two individuals i and j at time t, where an event occurs for j but not necessarily for i. Their predictions based on the classical SVM idea are $\langle \psi, F(\boldsymbol{x}_i)\rangle + b$ and $\langle \psi, F(\boldsymbol{x}_j)\rangle + b$, respectively. The ranking approach aims at predicting the correct ranking of $\boldsymbol{y}_i$ and $\boldsymbol{y}_j$, meaning $(\boldsymbol{y}_i - \boldsymbol{y}_j)(\langle \psi, F(\boldsymbol{x}_i)\rangle - \langle \psi, F(\boldsymbol{x}_j)\rangle) > 0$. If we further suppose $\boldsymbol{y}_j < \boldsymbol{y}_i$, predictions for i and j have to verify $\langle \psi, F(\boldsymbol{x}_i)\rangle - \langle \psi, F(\boldsymbol{x}_j)\rangle > 0$. Therefore, the ranking approach, as proposed by Van Belle et al. (2007), is equivalent to solving the optimization problem

$$\begin{aligned} \min_{\psi,\xi} \quad & \tfrac{1}{2}\|\psi\|^2 + \gamma \sum_{\substack{j<i \\ \delta_i=1}} v_{ij}\xi_{ij} \\ \text{subject to} \quad & \langle \psi, F(\boldsymbol{x}_i)\rangle - \langle \psi, F(\boldsymbol{x}_j)\rangle \geq 1 - \xi_{ij} \\ \text{and} \quad & \xi_{ij} \geq 0,\ i, j = 1, ..., n \,, \end{aligned} \tag{4}$$

where

$$v_{ij} = \begin{cases} 1 & \text{if } i \text{ and } j \text{ are comparable} \\ 0 & \text{else} \end{cases}$$

is the comparison function for observations i and j. Assuming that individuals are sorted increasingly by their survival times, two observations i and j ($i < j$) are comparable if their survival times t_i and t_j verify $t_i < t_j$ and $\delta_i = 1$, i.e., the shorter observed time is not censored. If we further suppose that the observation with the shortest survival time is not censored, each observation $i = 2, \ldots, n$ will be comparable to at least one other individual. The problem posed in (4) is equivalent to maximizing the concordance index (*C-index*) defined by Van Belle et al. (2007) over comparable pairs for a given prediction function u as

$$CI_n(u) = \frac{1}{n(n-1)} \sum_{v_{ij}=1} I\left[\Big(u(x_i) - u(x_j)\Big)\Big(t_i - t_j\Big)\right],$$

where $I(a) = 1$ if $a > 0$, and $I(a) = 0$, otherwise. This definition is inspired from those proposed by Harrell et al. (1984). Given an observation i, we only consider one comparable observation $\bar{j}(i) = i - 1$. We denote the first variant of the ranking approach presented in (4) as *vanbelle1*. In an alternative formulation, Van Belle et al. (2011) restrict the comparability effect to the inequality constraints as follows:

$$\begin{aligned} \min_{\psi,\xi} \quad & \tfrac{1}{2}\|\psi\|^2 + \gamma \sum_{i=1}^{n} \xi_i \\ \text{subject to} \quad & \langle \psi, F(\boldsymbol{x}_i)\rangle - \langle \psi, F(x_{\bar{j}(i)})\rangle \geq y_i - y_{\bar{j}(i)} - \xi_i \\ \text{and} \quad & \xi_i \geq 0\,, \end{aligned} \tag{5}$$

where $i = 1, \ldots, n$. Thereby, the value 1 on the right-hand side of the first constraint in *vanbelle1* (4) is replaced by the difference $y_i - y_{\bar{j}(i)}$ between the survival times y_i and $y_{\bar{j}(i)}$ of data points i and its nearest comparable neighbor $\bar{j}(i)$. We denote this second variant of the ranking approach as *vanbelle2*. Another variant of the ranking approach was developed by Evers and Messow (2008) and implemented in the R package **survpack**, offering the possibility to fit survival SVM models using either linear or radial basis kernels. In this variant, for a data point i, all comparable pairs are considered, instead of only the nearest neighbor. The denotation *evers* is used to refer to this variant.

The hybrid approach (Van Belle et al., 2011) combines the regression and ranking approaches in the survival SVMs problem. Thus, the constraints of (3) and (5) are included in the optimization problem

$$\begin{aligned} \min_{\psi,b,\varepsilon,\xi,\xi^*} \quad & \tfrac{1}{2}\|\psi\|^2 + \gamma \sum_{i=1}^{n} \varepsilon_i + \mu \sum_{i=1}^{n} (\xi_i + \xi_i^*) \\ \text{subject to} \quad & \langle \psi, F(\boldsymbol{x}_i)\rangle - \langle \psi, F(\boldsymbol{x}_{\bar{j}(i)})\rangle \geq \boldsymbol{y}_i - \boldsymbol{y}_{\bar{j}(i)} - \varepsilon_i\,, \\ & \boldsymbol{y}_i - \langle \psi, F(\boldsymbol{x}_i)\rangle - b \leq \xi_i\,, \\ & \delta_i(\langle \psi, F(\boldsymbol{x}_i)\rangle + b - \boldsymbol{y}_i) \leq \xi_i^*\,, \\ \text{and} \quad & \varepsilon_i, \xi_i, \xi_i^* \geq 0\,, \end{aligned} \tag{6}$$

where $i = 1, \ldots, n$. To solve the optimization problems (3), (4), (5) and (6) we consider the corresponding Lagrange function in dual space and solve the quadratic optimization problem. The next section presents our implementation of these models in R.

Implementation in R

We implemented the four models presented in (3), (4), (5) and (6) in the **survivalsvm** package. The function `survivalsvm` fits a new survival model, and risk ranks are predicted using the generic R function `predict`. Common to the four models is that a quadratic optimization problem is solved when moving to the dual space. In the function `survivalsvm`, we solve this optimization problem using two quadratic programming solvers: `ipop` from the package **kernlab** (Karatzoglou et al., 2004) and `pracma` from the package **pracma** (Borchers, 2016). The function `pracma` wraps `quadprog` from the package **quadprog** (Berwin A., 2013), which is implemented in **Fortran** to solve quadratic problems as described by Goldfarb and Idnani (1982). Thereby, the kernel matrix is assumed to be positive semi-definite. If this is not the case, the function `nearPD` from the **Matrix** package (Bates and Maechler, 2016) is used to adjust the kernel matrix to the nearest positive semi-definite matrix. In contrast to

quadprog, ipop is written in pure R. Hence, the runtime of ipop is expected to be greater than that of quadprog for solving the same optimization problem. However, an advantage of ipop is that the kernel matrix does not need to be modified when solving the optimization problem. The user of survivalsvm can choose which solver is used for solving the quadratic optimization problem.

As in the ranking formulations (4) and (5), the hybrid formulation (6) calculates differences between comparable data points. Three options to define comparable pairs are available in survivalsvm: makediff1 removes the assumption that the first data point is not censored, makediff2 computes differences over not censored data points only and makediff3 uses the definition described above.

The R package **survivalsvm** allows the user to choose one of the four kernels linear, additive (Daemen and De Moor, 2009), radial basis, and polynomial, labeled lin_kernel, add_kernel, rbf_kernel and poly_kernel, respectively. They can be passed to the survivalsvm function using the kernel parameter.

Example of usage

To exemplify the usage, the implementation is applied to the data set *veteran* available in the package **survival** (Therneau, 2015). The function Surv from the package **survival** serves to construct a survival target object.

```
R> library(survivalsvm)
R> library(survival)
R> data(veteran, package = "survival")
```

First, we split the data into a training and a test data set

```
R> set.seed(123)
R> n <- nrow(veteran)
R> train.index <- sample(1:n, 0.7 * n, replace = FALSE)
R> test.index <- setdiff(1:n, train.index)
```

and next fit a survival support vector regression model

```
R> survsvm.reg <- survivalsvm(Surv(diagtime, status) ~ .,
+        subset = train.index, data = veteran,
+        type = "regression", gamma.mu = 1,
+        opt.meth = "quadprog", kernel = "add_kernel")
```

The regularization parameter is passed using the argument gamma.mu. For each of the models (3), (4) and (5), only one value is required, while two values are needed when fitting a hybrid model. Calling the print function on the output gives

```
R> print(survsvm.reg)
survivalsvm result
Call:
survivalsvm(Surv(diagtime, status) ~ ., subset = train.index, data = veteran,
 type = "regression",  gamma.mu = 1, opt.meth = "quadprog", kernel = "add_kernel")

Survival svm approach                 : regression
Type of Kernel                        : add_kernel
Optimization solver used              : quadprog
Number of support vectors retained    : 39
survivalsvm version                   : 0.0.4
```

We can now make the predictions for the observations given by test.index:

```
R> pred.survsvm.reg <- predict(object = survsvm.reg,
+        newdata = veteran, subset = test.index)
```

and print the prediction object:

```
R> print(pred.survsvm.reg)
survivalsvm prediction

Type of survivalsvm                    : regression
Type of kernel                         : add_kernel
Optimization solver used in model      : quadprog
Predicted risk ranks                   : 13.89 14.95 11.12 15.6 10.7 ...
survivalsvm version                    : 0.0.4
```

Real data application and comparison to other survival models

To evaluate the survival SVM models and our implementation, four publicly available survival data sets were used. The first is the data set *veteran* from the Veteran's lung cancer trial study (Kalbfleisch and Prentice, 2002), available in the package **survival**. It includes 137 individuals and 5 covariates. Second, we utilized data from the *Interpretation of a Trial Stopped Early* study (Ermerson and Banks, 1994) in which 130 individuals participated. Two survival outcomes were of interest in this study, complete remission and death. The corresponding data sets are labeled *leuk_cr* and *leuk_death*. For both data sets, 10 covariates are provided. Third, we used the Germany Breast Cancer Study Group 2 (*GBSG2*) data (Schumacher et al., 1994) which consists in 686 samples and 8 covariates. Finally, we considered the *Mayo Clinic Lung Cancer* (*MCLC*) study (Loprinzi et al., 1994) comprising 168 individuals and 8 covariates. Table 1 provides a brief summary of the data sets used.

Data set	Sample size	#Covariates	Status	Survival time
veteran	137	5	status	time
leuk_cr	130	10	complete_rem	data_cr
leuk_death	130	10	status_last_fol_up	data_last_fol_up
GBSG2	686	8	cens	time
MCLC	168	8	status	time

Table 1: Data sets used to compare prediction performance and runtime. The data sets *leuk_cr* and *leuk_death* differ only in the event considered. In *leuk_cr*, the event is a complete remission while in *leuk_death* the event of interest is death. The two last columns give the names of the censoring status and survival time variables.

For each data set, we fitted the four survival SVM models (3), (4), (5) and (6) using linear, additive and radial basis function (*RBF*) kernels. The ranking approach of survival SVMs implemented in the R package **survpack** is applied using linear and RBF kernels, the two kernels offered by the package. The Cox PH model, random survival forest (*RSF*) (Ishwaran et al., 2008) and gradient boosting (*Gboost*) for survival analysis Ridgeway (1999) served as reference models.

For RSF, the package **randomForestSRC** (Ishwaran and Kogalur, 2016) was used. The number of random variables for splitting and the minimal number of events in the terminal nodes were tuned when building survival trees. **randomForestSRC** refers to these parameters as `mtry` and `nodesize`, respectively. For gradient boosting models implemented in **mboost** (Hothorn et al., 2016), we fitted a PH model as base learner. The number of boosting iterations and the regression coefficient were tuned. In **mboost**, these parameters are named `mstop` and `nu`, respectively.

Tuning was conducted using 5×10-fold nested cross validation: Data sets were randomly divided into 5 almost equally sized subsamples, and in each iteration, one of the 5 groups was used as test data set. On the remaining groups, the models were trained after tuning the model parameters with a 10-fold cross validation. The best models were chosen based on the *C*-index. The **mlr** package (Bischl et al., 2016a,b) was employed for parameter tuning.

Experiments were run on a high performance computing platform using 6 cores of Intel Xeon 3.33 GHz CPUs and an available memory of 183 GB. Table 2 summarizes the empirical mean runtime required by each survival SVM model to run a single resampling operation. As depicted, the runtimes are influenced by the kernel function used. Compared with linear and additive kernel functions, which required approximatively equal runtimes, the runtimes of the radial basis function (RBF) are higher. The reason is that the RBF requires one additional parameter, which has to be optimized via tuning. The effect of the number of parameters is also observed on the hybrid survival SVM approach, which required more runtime than the other approaches. Finally, the runtime of the ranking approaches *vanbelle1* and *vanbelle2*, implemented in **survivalsvm**, was considerably lower than that of the *evers* approach, implemented in the **survpack** package.

Table 3 and Figure 2 present the performance estimates of the compared models, based on the *C*-index. Of the 17 models, the best model received the rank 1, the worst model the rank 17. In case of ties, all tied models were assigned the mean rank value. On the *veteran* data set, the hybrid approach with the additive kernel performed best of all survival SVM models, with only slight differences to the PH model and *SSVR* approach. The models based on the ranking approach (*vanbelle1* and *vanbelle2*)

Data set	Kernel	Mean runtime in minutes				
		vanbelle1	vanbelle2	SSVR	hybrid	evers
veteran	linear	0.45	0.44	1.33	16.12	6.57
	additive	0.58	0.60	1.41	15.97	
	RBF	4.79	5.06	14.02	144.03	43.33
leuk_cr	linear	0.87	0.90	1.38	26.24	3.56
	additive	0.98	0.99	1.54	30.75	
	RBF	2.98	3.21	8.57	238.97	21.13
leuk_death	linear	0.29	0.30	0.96	28.08	4.82
	additive	0.33	0.36	0.96	30.84	
	RBF	2.99	3.10	8.52	269.54	20.04
GBSG2	linear	2.90	3.01	41.89	1064.53	1005.39
	additive	3.65	4.21	65.57	374.68	
	RBF	30.91	35.23	597.77	NA†	NA†
MCLC	linear	0.47	0.46	1.83	48.07	14.92
	additive	0.64	0.62	2.10	17.18	
	RBF	4.86	5.11	17.98	585.94	81.34

Table 2: Mean runtime for each survival SVM model to run a single resampling operation. This operation includes tuning of the parameters of regularization for the ranking and regression based models. Since the RBF, in comparison to linear and additive kernel functions, requires one additional parameter to be computed, its runtimes were higher. Furthermore, because the hybrid approach uses two parameters of regularization, it also needs more time to find the best parameters of regularization. The implementation of the *evers* approach in **survpack** does not include the additive kernel.
† Interrupted after 10 days of computation.

performed worse than the other models. The differences between the reference models were small. For the *leuk_cr* data set, the *evers* approach with the RBF kernel was the best SVM approach, followed by the hybrid approach with the additive kernel. The best reference models were the PH model and GBoost. For *leuk_death*, the hybrid approach with the additive kernel performed on par with RSF. The RSF model also performed better than the other reference models on the *GBSG2* data set, while *evers* performed worse. The hybrid model was still the best survival SVM model, with almost the same results for the linear and additive kernel. No results could be obtained for the RBF kernel in 10 days of computation. For the *MCLC* data set, the survival SVM with the hybrid approach and RBF kernel performed best, while the PH model was the best reference model. The differences were small, except for *evers*, which performed worse. In summary, there are only slight differences between the best survival SVM models and the best reference models. However, the differences between the SVM approaches and kernels were substantial.

Conclusion

We presented the R package **survivalsvm** for fitting survival SVMs. Three approaches are available in the package. First, in the regression approach, the classical SVR was extended to censored survival outcomes. Second, in the ranking approach, the ranks between model predictions and the observed survival times are maximized based on the *C*-index. The third approach, called the hybrid approach, combines the two first approaches into a single model. We implemented these three approaches in the **survivalsvm** package and used 5 data sets to compare the prediction performance and runtime of our implementation with the Cox PH model, RSF and gradient boosting. Furthermore, we included an implementation of a variant of the ranking approach (Evers and Messow, 2008) in the comparison. Of the survival SVM models, the hybrid approach generally performed best in terms of prediction performance but was slowest to compute. We observed only small differences between the best SVM models and the best reference models and could not determine a clear winner.

Comparing the ranking and regression based models, the *evers* approach always required more runtime than the approaches implemented in **survivalsvm**. Although the hybrid approach performed better than the others survival SVM approaches, its runtime was considerably increased. This was due to the fact that the formulation of the hybrid approach needs two parameters of regularization, while the ranking and regression approaches require only one parameter.

Method	Kernel	veteran C-index (SD)	Rank	leuk_cr C-index (SD)	Rank	leuk_death C-index (SD)	Rank	GBSG2 C-index (SD)	Rank	MCLC C-index (SD)	Rank
vanbelle1	linear	0.68 (0.05)	11	0.61 (0.05)	13	0.62 (0.06)	15	0.60 (0.05)	9	0.61 (0.05)	9.5
	additive	0.57 (0.07)	17	0.63 (0.06)	12	0.66 (0.08)	11	0.59 (0.03)	11	0.62 (0.05)	5.5
	RBF	0.60 (0.10)	14	0.53 (0.19)	17	0.59 (0.12)	17	0.60 (0.07)	10	0.61 (0.03)	7
vanbelle2	linear	0.59 (0.15)	16	0.59 (0.06)	14.5	0.61 (0.08)	16	0.59 (0.06)	12	0.61 (0.06)	12
	additive	0.59 (0.07)	15	0.57 (0.03)	16	0.64 (0.10)	13	0.58 (0.04)	13	0.62 (0.05)	5.5
	RBF	0.64 (0.05)	12	0.59 (0.06)	14.5	0.63 (0.11)	14	0.58 (0.07)	14	0.61 (0.05)	9.5
SSVR	linear	0.69 (0.03)	8.5	0.66 (0.05)	10	0.68 (0.08)	10	0.67 (0.04)	6.5	0.63 (0.09)	3
	additive	0.71 (0.05)	3	0.70 (0.03)	7	0.71 (0.09)	3	0.67 (0.04)	6.5	0.64 (0.08)	2
	RBF	0.70 (0.04)	4	0.68 (0.07)	9	0.70 (0.09)	4	0.67 (0.06)	8	0.61 (0.08)	14
hybrid	linear	0.69 (0.04)	10	0.71 (0.05)	5	0.68 (0.05)	9	0.68 (0.04)	4	0.62 (0.03)	4
	additive	0.71 (0.02)	1	0.72 (0.02)	4	0.72 (0.09)	1.5	0.68 (0.05)	5	0.61 (0.05)	9.5
	RBF	0.69 (0.03)	8.5	0.71 (0.06)	6	0.69 (0.11)	8	NA†		0.64 (0.05)	1
evers	linear	0.62 (0.08)	13	0.64 (0.11)	11	0.65 (0.09)	12	0.56 (0.03)	15	0.58 (0.13)	16
	RBF	0.70 (0.08)	6	0.73 (0.06)	3	0.70 (0.12)	6	NA†		0.56 (0.06)	17
PH Model		0.71 (0.04)	2	0.73 (0.03)	1.5	0.70 (0.10)	5	0.68 (0.03)	2.5	0.61 (0.05)	9.5
RSF		0.70 (0.05)	5	0.70 (0.06)	8	0.72 (0.09)	1.5	0.69 (0.03)	1	0.60 (0.02)	15
GBoost		0.70 (0.09)	7	0.73 (0.03)	1.5	0.69 (0.09)	7	0.68 (0.03)	2.5	0.61 (0.06)	13

Table 3: Prediction performance of the five survival SVMs (*vanbelle1*, *vanbelle2*, *SSVR*, *hybrid* and *evers*) and three reference methods (PH, RSF and GBoost) on 5 data sets. The *C*-index was computed for each method using 5×10-fold nested cross validation. Mean values and standard deviations (*SD*) are shown. The *C*-index and a rank is presented for each model. Rank 1 was assigned to the best performing model, rank 17 to the worst performing one. In case of ties, all tied models were assigned the mean rank value. The `quadprog` optimizer was used in the package **survivalsvm**. The best SVM models and the best reference models are marked in light and dark gray colors, respectively. The implementation of the *evers* approach in **survpack** does not include the additive kernel.
† Interrupted after 10 days of computation.

ISSN 2073-4859

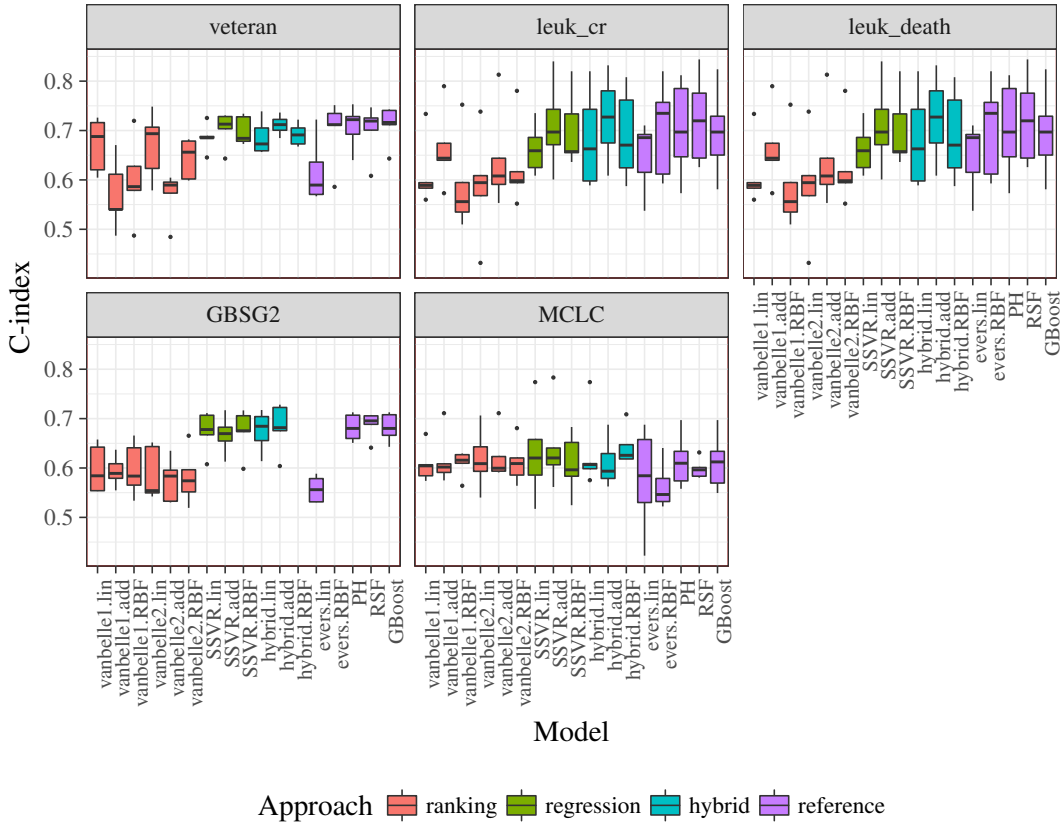


Figure 2: Prediction performance for the five survival support vector models (*vanbelle1*, *vanbelle2*, *SSVR*, *hybrid* and *evers*) and three reference methods (PH, RSF and GBoost) on 5 data sets. The *C*-index was computed for each method using nested 5×10 cross validation. The `quadprog` optimizer was used in the package **survivalsvm**. Plots were generated using the **ggplot2** (Wickham, 2009) and **tikzDevice** (Sharpsteen et al., 2016) packages.

The best performing kernel functions depended on the data set and the chosen SVM model. For the ranking approaches, the differences were larger than the regression and hybrid approaches. For the hybrid approach, the additive and RBF kernels achieved the best results. However, the runtimes for the RBF kernel were substantially larger. Again, this was due to the tuning of an additional parameter.

Our implementation utilizes quadratic programming and an interior point optimizer to solve the quadratic optimization problem derived from the primal support vector optimization problem. When the quadratic programming is used for a non positive semi-definite kernel matrix, this matrix is slightly modified to the nearest positive semi-definite matrix. Calling the interior point optimizer does not make any modification on the original matrix, but is computationally slower since the software is fully implemented in R. Pölsterl et al. (2015) proposed a fast algorithm to train survival SVMs in primal space. This algorithm is fast in low dimensions, but for high dimensional problems the authors recommended reducing the dimensions before applying an SVM algorithm. However, some special and fast algorithms, such as the sequential minimal optimization (*SMO*) (Platt, 1998), which are available for classical SVM optimization problems, were shown to be more accurate (Horn et al., 2016). The implementation for survival SVMs could possibly be improved by an extension of the *SMO* optimization procedure.

Having restrictions on only the nearest neighbor in the ranking approach, as formulated in *vanbelle1* and *vanbelle2*, can considerably improve the computational performance, but can also reduce prediction performance. In principle, the number of nearest neighbors is not limited to the choices of Evers and Messow (2008) and Van Belle et al. (2008). Since the optimal number of neighbors depends on the dataset and the availability of computational resources, it may be included as a further tuning parameter.

In conclusion, we have shown that SVMs are a useful alternative for survival prediction. The package **survivalsvm** provides a fast and easy-to-use implementation of the available approaches of survival SVMs. Our results show that the choice of the SVM model and the kernel function is crucial. In addition to prediction performance, runtime is an important aspect for large data sets. We

recommend to conduct benchmark experiments using several approaches and available kernels before analyzing a data set.

Availability

The package **survivalsvm** is available on CRAN and a development version at https://github.com/imbs-hl/survivalsvm. The presented examples and R code to reproduce all results in this paper are available at https://github.com/imbs-hl/survivalsvm-paper.

Bibliography

D. Bates and M. Maechler. *Matrix: Sparse and Dense Matrix Classes and Methods*, 2016. URL https://CRAN.R-project.org/package=Matrix. R package version 1.2-6. [p415]

T. Berwin A. *Quadprog: Functions to Solve Quadratic Programming Problems.*, 2013. URL https://CRAN.R-project.org/package=quadprog. R package version 1.5-5. [p415]

B. Bischl, M. Lang, L. Kotthoff, J. Schiffner, J. Richter, Z. Jones, and G. Casalicchio. *Mlr: Machine Learning in R*, 2016a. URL https://CRAN.R-project.org/package=mlr. R package version 2.9. [p417]

B. Bischl, M. Lang, L. Kotthoff, J. Schiffner, J. Richter, E. Studerus, G. Casalicchio, and Z. M. Jones. mlr: Machine learning in R. *Journal of Machine Learning Research*, 17(170):1–5, 2016b. URL http://jmlr.org/papers/v17/15-066.html. [p417]

C. Bishop. *Pattern Recognition and Machine Learning*. Springer-Verlag, 2nd edition, 2007. ISBN 978-0-387-31073-2. [p413]

H. W. Borchers. *Pracma: Practical Numerical Math Functions*, 2016. URL https://CRAN.R-project.org/package=pracma. R package version 1.9.3. [p415]

J. Cervantes, X. Li, W. Yu, and K. Li. Support vector machine classification for large data sets via minimum enclosing ball clustering. *Neurocomputing*, 71(4–6):611–619, 2008. URL https://dx.doi.org/10.1016/j.neucom.2007.07.028. [p412]

A. Daemen and B. De Moor. Development of a kernel function for clinical data. In *2009 Annual International Conference of the IEEE Engineering in Medicine and Biology Society*, pages 5913–5917, 2009. [p416]

T. Eitrich and B. Lang. Efficient optimization of support vector machine learning parameters for unbalanced datasets. *Journal of Computational and Applied Mathematics*, 196(2):425–436, 2006. URL http://dx.doi.org/10.1016/j.cam.2005.09.009. [p413]

S. Ermerson and P. Banks. Interpretation of a leukemia trial stopped early. In N. Lange, editor, *Case Studies in Biometry*, chapter 14, pages 275 – 99. John Wiley & Sons, 1994. [p417]

L. Evers. *survpack: Methods for Fitting High-Dimensional Survival Models*, 2009. URL http://www.stats.gla.ac.uk/~levers/software.html. R package version 0.1-4. [p414]

L. Evers and C.-M. Messow. Sparse kernel methods for high-dimensional survival data. *Bioinformatics*, 24(14):1632–1638, 2008. URL https://dx.doi.org/10.1093/bioinformatics/btn253. [p413, 414, 415, 418, 420]

D. Goldfarb and A. Idnani. Dual and primal-dual methods for solving strictly convex quadratic programs. In J. P. Hennart, editor, *Numerical Analysis*, number 909 in Lecture Notes in Mathematics, pages 226–239. Springer-Verlag, 1982. URL https://dx.doi.org/10.1007/BFb0092976. [p415]

F. E. Harrell, K. L. Lee, R. M. Califf, D. B. Pryor, and R. A. Rosati. Regression modelling strategies for improved prognostic prediction. *Statistics in Medicine*, 3(2):143–152, 1984. URL https://dx.doi.org/10.1002/sim.4780030207. [p415]

R. Herbrich, T. Graepel, and K. Obermayer. Support vector learning for ordinal regression. In *9th International Conference on Artificial Neural Networks (ICANN99)*, pages 97–102, 1999. URL https://dx.doi.org/10.1049/cp:19991091. [p414]

D. Horn, A. Demircioğlu, B. Bischl, T. Glasmachers, and C. Weihs. A comparative study on large scale kernelized support vector machines. *Advances in Data Analysis and Classification*, pages 1–17, 2016. URL https://dx.doi.org/10.1007/s11634-016-0265-7. [p420]

T. Hothorn, P. Buehlmann, T. Kneib, M. Schmid, and B. Hofner. *mboost: Model-Based Boosting*, 2016. URL http://CRAN.R-project.org/package=mboost. R package version 2.6-0. [p417]

C.-W. Hsu, C.-C. Chang, and C.-J. Lin. A practical guide to support vector classification. Technical report, Department of Computer Science, National Taiwan University, 2003. URL http://www.csie.ntu.edu.tw/~cjlin/papers.html. Date of access: April 4, 2017. [p412]

H. Ishwaran and U. B. Kogalur. *Random Forests for Survival, Regression and Classification (RF-SRC)*, 2016. URL https://cran.r-project.org/package=randomForestSRC. R package version 2.3.0. [p417]

H. Ishwaran, U. B. Kogalur, E. H. Blackstone, and M. S. Lauer. Random survival forests. *The Annals of Applied Statistics*, 2(3):841–860, 2008. URL https://dx.doi.org/10.1214/08-AOAS169. [p417]

J. D. Kalbfleisch and R. L. Prentice. *The Statistical Analysis of Failure Time Data*. John Wiley & Sons, 2nd edition, 2002. ISBN 978-0-471-36357-6. [p412, 417]

A. Karatzoglou, A. Smola, K. Hornik, and A. Zeileis. Kernlab – an S4 package for kernel methods in R. *Journal of Statistical Software*, 11(9):1–20, 2004. URL https://dx.doi.org/10.18637/jss.v011.i09. [p413, 415]

F. M. Khan and V. B. Zubek. Support vector regression for censored data (SVRc): A novel tool for survival analysis. In *Eighth IEEE International Conference on Data Mining (ICDM08)*, pages 863–868, 2008. URL https://dx.doi.org/10.1109/ICDM.2008.50. [p414]

D. G. Kleinbaum and M. Klein. *Survival Analysis, a Self-Learning Text*. Springer-Verlag, 3rd edition, 2012. ISBN 978-1-4419-6646-9. [p412]

E. T. Lee and J. W. Wang. *Statistical Methods for Survival Data Analysis*. John Wiley & Sons, 2003. ISBN 978-0-471-45854-8. [p412]

C. L. Loprinzi, J. A. Laurie, H. S. Wieand, J. E. Krook, P. J. Novotny, J. W. Kugler, J. Bartel, M. Law, M. Bateman, and N. E. Klatt. Prospective evaluation of prognostic variables from patient-completed questionnaires. North Central Cancer Treatment Group. *Journal of Clinical Oncology: Official Journal of the American Society of Clinical Oncology*, 12(3):601–607, 1994. URL https://dx.doi.org/10.1200/JCO.1994.12.3.601. [p417]

J. Platt. Sequential minimal optimization: A fast algorithm for training support vector machines. Technical report, Microsoft Research, 1998. URL https://www.microsoft.com/en-us/research/publication/sequential-minimal-optimization-a-fast-algorithm-for-training-support-vector-machines. Date of access: April 4, 2017. [p420]

S. Pölsterl, N. Navab, and A. Katouzian. Fast training of support vector machines for survival analysis. In A. Appice, P. P. Rodrigues, V. Santos Costa, J. Gama, A. Jorge, and C. Soares, editors, *Machine Learning and Knowledge Discovery in Databases, ECML PKDD 2015*, pages 243–259. Springer-Verlag, 2015. URL https://dx.doi.org/10.1007/978-3-319-23525-7_15. [p420]

A. Rhode. Learning Kernels SVM, 2012. URL https://quantsignals.wordpress.com/2012/09/25/kernel-learning-svm. Accessed on Jan 03, 2018. [p413]

G. Ridgeway. The state of boosting. *Computing Science and Statistics*, 31:172–181, 1999. [p417]

S. J. Russell and P. Norvig. *Artificial Intelligence: A Modern Approach*. Pearson, Boston, 3rd edition, 2010. ISBN 978-0-13-207148-2. [p413]

M. Schumacher, B. G., H. Bojar, K. Huebner, M. Olschewski, W. Sauerbrei, C. Schmoor, C. Beyerle, R. L. A. Neumann, and H. F. Rauschecker. Randomized 2×2 trial evaluating hormonal treatment and the duration of chemotherapy in node-positive breast cancer patients. *Journal of Clinical Oncology*, 12(10):2086 – 2093, 1994. URL https://dx.doi.org/10.1200/JCO.1994.12.10.2086. [p417]

C. Sharpsteen, C. Bracken, K. Müller, and Y. Xie. *tikzDevice: R Graphics Output in LaTeX Format*, 2016. URL https://CRAN.R-project.org/package=tikzDevice. R package version 0.10-1. [p420]

P. K. Shivaswamy, W. Chu, and M. Jansche. A support vector approach to censored targets. In *Seventh IEEE International Conference on Data Mining (ICDM 2007)*, pages 655–660, 2007. URL https://dx.doi.org/10.1109/ICDM.2007.93. [p413, 414]

T. M. Therneau. *A Package for Survival Analysis in S*, 2015. URL https://CRAN.R-project.org/package=survival. version 2.38. [p416]

V. Van Belle, K. Pelckmans, J. Suykens, and S. Van Huffel. Support vector machines for survival analysis. In *Proceedings of the Third International Conference on Computational Intelligence in Medicine and Healthcare (CIMED2007)*, pages 1–8, 2007. [p413, 414, 415]

V. Van Belle, K. Pelckmans, J. Suykens, and S. Van Huffel. Survival SVM : a practical scalable algorithm. In *European Symposium on Artificial Neural Networks (ESANN)*, pages 89–94, 2008. [p413, 414, 420]

V. Van Belle, K. Pelckmans, S. Van Huffel, and J. A. K. Suykens. Support vector methods for survival analysis: a comparison between ranking and regression approaches. *Artificial Intelligence in Medicine*, 53(2):107–118, 2011. URL https://dx.doi.org/10.1016/j.artmed.2011.06.006. [p414, 415]

V. N. Vapnik. *The Nature of Statistical Learning Theory*. Springer-Verlag, 1995. ISBN 978-0-387-94559-0. [p412, 414]

V. N. Vapnik. *Statistical Learning Theory*. John Wiley & Sons, New York, 1998. ISBN 978-0-471-03003-4. [p413, 414]

H. Wickham. *ggplot2: Elegant Graphics for Data Analysis*. Springer-Verlag, 2009. ISBN 978-0-387-98140-6. URL http://ggplot2.org. [p420]

L. Yang and K. Pelckmans. *Survlab: A Survival Analysis Toolbox*, 2014. URL http://user.it.uu.se/~kripe367/survlab/index.html. version 2.0. [p414]

Césaire J. K. Fouodo
Institut für Medizinische Biometrie und Statistik
Universität zu Lübeck
Universitätsklinikum Schleswig-Holstein, Campus Lübeck
Lübeck, Germany
fouodo@imbs.uni-luebeck.de

Inke R. König
Institut für Medizinische Biometrie und Statistik
Universität zu Lübeck
Universitätsklinikum Schleswig-Holstein, Campus Lübeck
Lübeck, Germany
inke.koenig@imbs.uni-luebeck.de

Claus Weihs
Fakultät Statistik
Technische Universität Dortmund
Dortmund, Germany
weihs@statistik.uni-dortmund.de

Andreas Ziegler
StatSol
Lübeck, Germany
ziegler@statsol.de

Marvin N. Wright
Institut für Medizinische Biometrie und Statistik
Universität zu Lübeck, Universitätsklinikum Schleswig-Holstein, Campus Lübeck
Lübeck, Germany
Leibniz Institute for Prevention Research and Epidemiology – BIPS
Bremen, Germany
wright@leibniz-bips.de

Nonparametric Independence Tests and k-sample Tests for Large Sample Sizes Using Package HHG

by Barak Brill, Yair Heller, and Ruth Heller

Abstract Nonparametric tests of independence and k-sample tests are ubiquitous in modern applications, but they are typically computationally expensive. We present a family of nonparametric tests that are computationally efficient and powerful for detecting any type of dependence between a pair of univariate random variables. The computational complexity of the suggested tests is sub-quadratic in sample size, allowing calculation of test statistics for millions of observations. We survey both algorithms and the **HHG** package in which they are implemented, with usage examples showing the implementation of the proposed tests for both the independence case and the k-sample problem. The tests are compared to existing nonparametric tests via several simulation studies comparing both runtime and power. Special focus is given to the design of data structures used in implementation of the tests. These data structures can be useful for developers of nonparametric distribution-free tests.

Introduction

A common question that arises in the analysis of data is whether two random variables, X and Y, are independent. The null hypothesis is

$$H_0 : F_{XY}(x, y) = F_X(x)F_Y(y) \quad \forall \quad x, y \tag{1}$$

where F_X and F_Y are the marginal cumulative distribution functions of X and Y, and F_{XY} is the joint cumulative distribution function. The case where Y is categorical and X is continuous is the k-sample problem. An omnibus consistent test will reject the null hypothesis in (1) for any dependence between X and Y, with probability increasing to one as the sample size tends to infinity.

In recent years, there has been great interest in developing tests of independence that are able to identify complex dependencies based on N independent observations from F_{XY}. For univariate random variables, the first omnibus consistent test was based on summation of a score over all N 2×2 partitions of the sample space where every data point serves as a partition point (Hoeffding, 1948). This test is available via the function `hoeffd` from package **Hmisc** (Harrell Jr et al., 2018). Another classic approach is based on the measure of mutual information following partitioning of the data into a 2-dimensional grid (Paninski, 2003). This approach is taken in the R packages **minet** (Meyer et al., 2008, 2017), **infotheo** (Meyer, 2014), and **entropy** (Hausser and Strimmer, 2009, 2014) with various extensions to the partitioning schemes used.

Recently, several nonparametric omnibus consistent tests have been suggested that have computational complexity at least quadratic in sample size. Reshef et al. (2011), with CRAN package **minerva** (Albanese et al., 2013; Filosi et al., 2017), suggested MIC, which is based on the maximum of penalized estimated mutual information partitions. Gretton et al. (2008), with CRAN package **dHSIC** (Pfister et al., 2018; Pfister and Peters, 2017), suggested HSIC, which is a kernel test based on the empirical estimate of the Hilbert Schmidt norm of the cross-covariance operator. Székely et al. (2007, 2009), with CRAN package **energy** (Rizzo and Szekely, 2017), suggested dCov, which is based on the joint empirical characteristic function. Both kernel and characteristic function methods may be implemented in a scenario where X or Y are multivariate. Heller et al. (2016), with CRAN package **HHG**, suggested tests which aggregate by maximization or by summation the likelihood ratio test (LRT) scores over all possible partitions of data, with m partition points for each variable. The suggested tests aggregate over all partitions of the data of size $m \times m$ (i.e., having $m \times m$ cells) for a range of m values.

For the k-sample problem, Székely and Rizzo (2004) suggested a test based on joint empirical characteristic function, which they implemented in package **energy** as well. Gretton et al. (2012a) suggested a family of consistent two sample tests based on kernels. The function `kmmd` from package **kernlab** (Zeileis et al., 2004) implements this family of tests for several kernel choices. Jiang et al. (2015), with CRAN package **dslice**, suggested the dynamic slicing test statistic, which aggregates by maximization the penalized LRT score with a penalty for fine partitions. Heller et al. (2016), with CRAN package **HHG**, suggested tests which aggregate by maximization or by summation the LRT scores over all possible partitions of data.

The potential advantage in power as sample size increases in the state-of-the-art tests listed above

is hindered by the computational cost. In practice, many of the state-of-the-art tests cannot be applied when sample sizes are in the thousands. The computational problem is compounded when multiple tests are to be carried out in the same analysis, e.g., when the aim is to detect all pairwise associations among the variables in a dataset.

Modifications to some of the tests listed above can be used for large sample sizes. For the HSIC test statistic, it was suggested to compute the quadratic time HSIC test statistic for subsets of the data, and then aggregate the HSIC test statistics towards the test statistic for the null hypothesis in (1) (Gretton et al., 2012a,b). Two approximate HSIC statistics, the Nyström HSIC test statistic and the random fourier feature HSIC, have been shown to have reduced computational complexity while enjoying power comparable to the original HSIC statistic (Zhang et al., 2018). Other computationally efficient ways to compute HSIC were suggested in Jitkrittum et al. (2016b,a). The three computationally efficient adaptations of the HSIC test (Nyström, RFF, FSIC) have an intrinsic trade-off between power and computational complexity given by a resolution parameter of the method. The user is able to 'pay in runtime' for more power. A computationally efficient algorithm for computing the univariate dCov test statistic in $O(Nlog(N))$ time was developed in Huo and Székely (2016). For the suggested test in Gretton et al. (2012a), computationally efficient modifications have been considered in Zhao and Meng (2015) and Chwialkowski et al. (2015). A computationally efficient algorithm for computing the univariate energy test statistic of Székely and Rizzo (2004) in $O(Nlog(N))$ time was developed in Huang and Huo (2017). Jiang et al. (2015) suggested considering only a subset of partition locations for large sample sizes for their dynamic slicing test. In our present work, we suggest a similar modification for the tests in Heller et al. (2016).

This paper describes two main contributions. The first is to provide a method and software for discovering dependence that has reasonable computational time for any sample size. Specifically, we modify the algorithms for the tests of Heller et al. (2016) , which were shown to have good power in complex settings, to aggregate only a representative subset of all partitions of the data, thus achieving a computational complexity which is sub-quadratic in sample size. The suggested tests have power competitive with state-of-the-art methods, but can be computed at a fraction of the time. Second, we extend the algorithms for the tests of Heller et al. (2016) to allow partitions with a different number of partition points on each axis. LRT scores of all partitions of size $m \times l$ of the sample space are aggregated where m and l are the number of partition points of the X and Y variables, respectively. This generalization does not increase the computational complexity, yet it can result in better power for alternatives where the optimal number of partition points in each axis is different.

The paper is organized as follows. We introduce the atom based *MinP* statistic and detail our novel contributions in the following two sections. Then, in "Usage examples", we present the work flow of the package (function calls and outputs). In "k-sample tests" we present the computationally efficient tests for the K-Sample problem and their work flow. In "Simulation" we compare the novel tests to other state-of-the-art tests in terms of power and runtime. Finally, in "Discussion" we provide some final remarks. Other tests available in **HHG** are detailed in Appendix B.

The atom based test statistics

Suppose we have N sample points, where each sample is a pair (x, y). We split the plane into m parts along the X axis and into l parts along the Y axis. (Note that for now, m and l are fixed; later we will show how we choose the best m and l automatically so that the user will not need to fix them.) We consider the set of all $m \times l$ partitions of the data, where a split point is possible only once every A ordered observations in each variable. The indivisible blocks of observations are called atoms. We assume for simplicity that the number of atoms N_A is an integer multiple of A, $N_A = N/A$. Figure 1 shows an example partition of the sample space based on atoms.

A cell c is defined by four integers, marking its left, right, top and bottom boundaries on an equidistant grid of $N_A \times N_A$ points. A cell with all four boundaries in the interior of the grid will be considered a 'center' cell (cell type 1). A cell with either its top or bottom boundary at the edge of the grid will be considered a 'top' or 'bottom' cell, respectively (cell type 2). A cell with either its left or right boundary at the edge of the grid will be considered a 'left' or 'right' cell, respectively (cell type 3). A set which has two boundaries at the edge of the grid (e.g., 'left' and 'top') will be called a 'corner' cell (cell type 4). Let O_c denote the number of samples observed in a cell, and E_c the expected number of samples when H_0 is true. For a cell of width w atoms and height h atoms, $E_c = \frac{w}{N_A}\frac{h}{N_A}N = whA^2/N$. Let $\mathcal{C}$ be the set of all cells and $\mathcal{C}(w, h, t)$ the set of all cells of size $w \times h$ atoms and type t, where $t \in \{1, 2, 3, 4\}$. We define the function $n(t, w, h, m, l)$, returning the number of

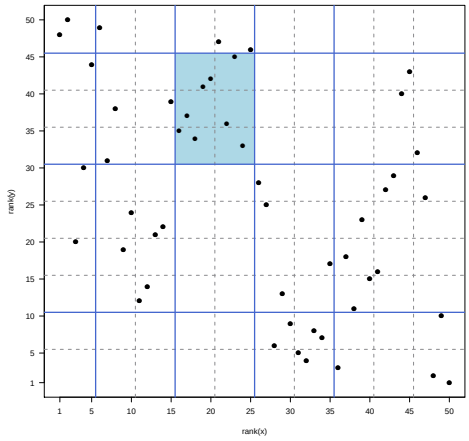


Figure 1: A visualization of a single partition of the atoms plane. The atom size is $A = 5$, so only partitions on the dashed black grid lines are allowed at boundary values $5.5, 10.5, 15.5, \ldots$. For $m = 5$ and $l = 4$, a single 5×4 partition is depicted with blue line boundaries. For $N = 50$ sample points (black dots), we have $N_A = 10$ atoms dividing each axis. The cell coloured in blue, which has width $w = 2$ atoms and height $h = 3$ atoms, has $O_c = 8$ sample points, whereas only $E_c = 2 \times 3 \times 5^2/50 = 3$ are expected under H_0.

$m \times l$ partitions a cell of type t and size w, h participates in:

$$n(t,w,h,m,l) = \begin{cases} \binom{N_A-w-2}{m-3} \cdot \binom{N_A-h-2}{l-3} & \text{t=1 (center cell)} \\ \binom{N_A-w-2}{m-3} \cdot \binom{N_A-h-1}{l-2} & \text{t=2 (top/bottom cell)} \\ \binom{N_A-w-1}{m-2} \cdot \binom{N_A-h-2}{l-3} & \text{t=3 (left/right cell)} \\ \binom{N_A-w-2}{m-2} \cdot \binom{N_A-h-1}{l-2} & \text{t=4 (corner cell)} \end{cases} \tag{2}$$

Let $\tilde{\Gamma}^{m\times l}$ be the set of all $m \times l$ partitions of the plane, where a split point is possible only between atoms. For a given $m \times l$ partition size, the aggregated by sum test statistic is

$$\begin{aligned} S^{m\times l} &= \sum_{\Gamma\in\tilde{\Gamma}^{m\times l}} \sum_{c\in\Gamma} O_c \log\left(O_c/E_c\right) = \sum_{c\in\mathcal{C}} \sum_{\Gamma\in\tilde{\Gamma}^{m\times l}} I\left(c\in\Gamma\right) O_c \log\left(O_c/E_c\right) \\ &= \sum_{t=1}^{4} \sum_{w=1}^{(N_A+1-m)} \sum_{h=1}^{(N_A+1-l)} \sum_{c\in\mathcal{C}(w,h,t)} O_c \log\left(O_c/E_c\right) n\left(t,w,h,m,l\right), \end{aligned} \tag{3}$$

where $I(\cdot)$ is the indicator function. The last equality in (3) demonstrates that for computing $S^{m\times l}$, we can iterate over cells instead of partitions, thus achieving a computational complexity of $\mathcal{O}\left({N_A}^4 + NlogN\right)$, even though the number of possible partitions is $|\tilde{\Gamma}^{m\times l}| = \binom{N_A-1}{m-1} \cdot \binom{N_A-1}{l-1}$.

Since the optimal partition size $m \times l$ is unknown, we propose taking the minimum p-value over the plausible range of partition sizes:

$$MinP = \min_{2\le m,l\le m.max} p_{m\times l}, \tag{4}$$

where $p_{m\times l}$ is the p-value of the test statistic $S^{m\times l}$.

In Appendix A we present the full pseudo-code for the algorithm, including the case when N is not a multiple of A. The pseudo-code also shows how $\{S^{m\times l} : m = 2, \ldots, m.max, l = 2, \ldots, m.max\}$ is computed at the same computational complexity as a single $S^{m\times l}$. The atom based test in (4) is consistent as long as $N_A \to \infty$ and $m.max^2/N \to 0$ (for a proof see Appendix C in Brill, 2016).

The null distribution of $MinP$ and $p_{m\times l}$

In this section, we show how to tabulate the null distribution of our proposed statistic $MinP$. This tabulation requires tabulating the null distribution of $S^{m\times l}$. Fortunately, the test statistics in (3)-(4) are based on the ranked observations and therefore are distribution free. Consequently, the null distributions of $\{S_{m\times l} : 2 \le m, l \le m.max\}$ can be tabulated off-line (prior to seeing the data) in order

to evaluate the p-value of any test statistic that combines $\{p_{m\times l} : 2 \leq m, l \leq m.max\}$.

The tabulation of the null distribution of $S^{m\times l}$ and the $MinP$ test statistics is described in the schematic diagram in Figure 2. The structure of generated null distributions and stored tabulations of null distributions differs. While one generates the vector of $S^{m\times l}$ statistics from a single sample, the package data structure for a null table is constructed by sorted arrays of the marginal distributions. Given a null table of size B repetitions, one can compute all marginal p-values for $S^{2\times 2}, S^{2\times 3}, S^{3\times 2}, \ldots, S^{m.max\times m.max}$ in $O(m.max^2 log(B))$ time once each of the $m \times l$ statistics is computed from data. Then the $MinP$ statistic is simply the minimum of all these p-values. An additional $O(log(B))$ search is required for computing the true p-value of the achieved $MinP$ test statistic. Given the null table, calculating the $MinP$ statistic takes altogether $O(N \log N + N_A^4 + m.max^2 \log B + \log B)$, and since $m.max \leq N_A$, this is at most $O(N \log N + N_A^4 + N_A^2 \log B)$. Since typically $\log B < N_A^2$, the complexity is typically $O(N \log N + N_A^4)$.

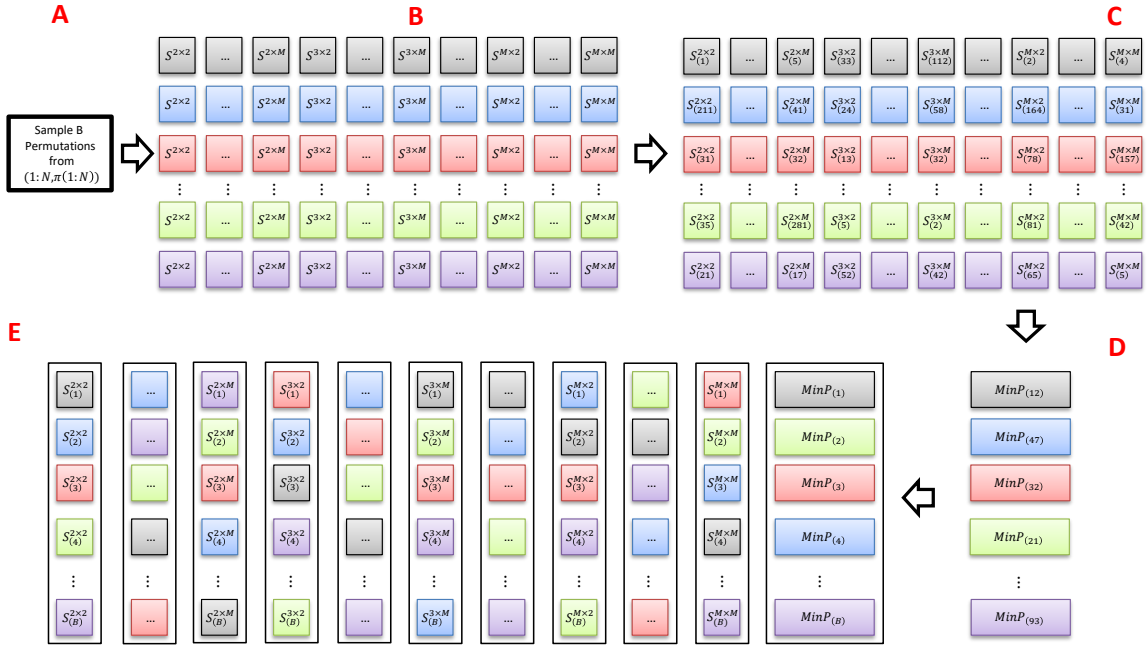


Figure 2: Schematic for the computation of the p-value for the $MinP$ statistic. In step A, sample N pairs without replacement from $\{1,\ldots,N\} \times \{1,\ldots,N\}$, B times. In step B, compute all test statistics for each sample of N pairs, color coded so that we have B rows and $(M-1)^2$ columns of test statistics. In step C, compute the within column rank for each test statistic: the rank is in the subscript for each test statistic, and its p-value is solely determined by the rank and B. In step D, compute $MinP$ for each sample (row) and its rank in the subscript. In step E, each sorted column along with the sorted $MinP$ column is stored individually for fast access and computation of p-values.

In practice, one does not need to maintain all marginal null distributions at a fixed resolution. Only the lower p-values (high $S^{m\times l}$ scores) are used for rejections. Thus, when one simulates a large null table such as $B = 10^6$, marginal ECDFs can be maintained at 0.001 increments of the cumulative probability distribution function for p-values bigger than some parameter for compression α' (e.g., $\alpha' = 0.05$) and at maximum resolution for lower p-values. Using the above parameters as the two different resolutions and α', a null table of 10^6 values is compressed to just over $5 \cdot 10^4$ values. This data structure makes p-value computation via the null table simple and efficient, with null tables sizes being maintainable even for large values of B.

In addition, this data structure is utilized as a combination method for statistics with a nominal false positive rate of α. Importantly, one can utilize this efficient data structure with any set of statistics and a general combination score which takes into account only marginal p-values. For example, Heller et al. (2016) propose another type of combination score for their tests, of the form $-\sum_{m=2}^{m.max} log\,(p_{m\times m})$. The combination score makes use of this data structure as well.

Usage examples

Function calls and input arguments

The test procedure utilizes two function calls. The first function call carries out the tabulation of the null distributions for both the $S^{m\times l}$ statistics and the *MinP* combination statistic. The second function call computes the *MinP* test statistic and its p-value, given the look-up tables for the distributions of $S^{m\times l}$ and *MinP* (see Table 1).

The arguments for the null distribution tabulation are the sample size, the maximal partition size considered, and the number of atoms. These are denoted by the parameters `n`, `mmax`, and `nr.atoms`, respectively. The default parameters for the test are `mmax = min(10,n)` and `nr.atoms = min(40,n)`. In Section "Simulations", we discuss these defaults in terms of power, runtime, and the trade-off between the two. Other parameters include the type of combination score used (*MinP* or Fisher) and the type of data partition used ($m \times m$ or $m \times l$).

The arguments for computing the MinP test statistic via `Fast.independence.test` are two vectors of size `n` for the joint observations of (X, Y), and a null table object produced by the function `Fast.independence.test.nulltable`. All test parameters are kept in the null table object.

Table 1: The novel atom-based tests in the HHG package.

Function Name	Description
`Fast.independence.test.nulltable`	Function creates null table objects for the atoms based omnibus distribution-free test of independence between two univariate random variables. Input arguments are the number of atoms, $m.max$, and the type of partitioning (all $m \times l$ or all $m \times m$ partitions).
`Fast.independence.test`	Performs the atoms based distribution-free test of independence of two univariate random variables, which is computationally efficient for large data sets. Input arguments are two numeric vectors of data, and optionally a null table object (if a null table has not been precomputed, one is generated on the fly).

Output description

The output of the function `Fast.independence.test.nulltable` is a null table data structure. This data structure contains the sorted marginal distributions of $S^{m\times l}$ statistics, the sorted distribution of the *MinP* test score, and the test parameters as in Figure 2.

The output of `Fast.independence.test` is an object of class `"UnivariateStatistic"` containing the results of both the marginal $S^{m\times l}$ tests performed on the data, along with the results of the *MinP* test. The fields `m.stats` and `pvalues.of.single.m` contain the test statistics for the tests of fixed partitions size and their respective P-values, given by (3). The fields `MinP` and `MinP.pvalue` contain the data adaptive test statistic and its P-value, given by (4).

Example code

We begin by computing the a look-up tables of the null distributions. This is done by calling the first function, with the sample size as a parameter:

```
# compute null table, using default m.max,
# number of atoms and number of permutations under the null.

nt = Fast.independence.test.nulltable(n)
```

The number of atoms for the procedure and the maximal partition size are set to their default values. The object `nt` now holds a look-up table which is specific for the selected test parameters and sample size. We will use this object to carry out the test using `Fast.independence.test`:

```
# carry out test, parameters set by null table passed.
res = Fast.independence.test(x,y,nt)

# print results and P-value. P-value given under entry 'MinP.pvalue'.
res
```

The last line prints the output for the test. The output begins by describing the test parameters: test statistic chosen, partition sizes considered, and number of atoms:

```
HHG univariate combined independence statistic
Statistics type combined:
sum of ADP-EQP-ML on Likelihood Ratio scores.

Single m statistics are the sum of scores over All Derived Partitions (ADP) of the data.
Statistics are normalized by the number of possible partitions and sample size.

Minimum partition size: 2  Maximum partition size: 10
Sample size: 1200
Equipartition nr.atoms: 40
```

The output printed shows the $S^{m\times l}$ test statistics along with their marginal p-values:

```
Single m (partition size) statistics:
       m=2       m=3      m=4      m=5      m=6      m=7      m=8      m=9      m=10
l=2   0.000548  0.00117  0.00180  0.00244  0.00308  0.00369  0.00429  0.00486  0.00541
l=3   0.001165  0.00244  0.00374  0.00505  0.00633  0.00759  0.00881  0.00998  0.01111
...     ...       ...          ...      ...      ...      ...      ...      ...      ...
l=10  0.004584  0.00950  0.01453  0.01962  0.02470  0.02973  0.03468  0.03955  0.04434

Single m (partition size) pvalues:
       m=2     m=3     m=4     m=5     m=6      m=7      m=8      m=9      m=10
l=2   0.199  0.1294  0.0995  0.0597  0.05473  0.04478  0.04478  0.03483  0.03483
l=3   0.134  0.0647  0.0398  0.0299  0.02488  0.01493  0.01493  0.01493  0.01493
...    ...    ...     ...     ...      ...      ...      ...      ...      ...
l=10  0.174  0.0846  0.0448  0.0249  0.01493  0.00995  0.00995  0.00995  0.00995
```

Finally, the output shows the MinP test statistic score along with it's p-value, which is the p-value for the test. The selected partition size attaining the smallest p-value is also shown:

```
MinP Statistic - Test statistic is minimum of above single m pvalues:
0.01
Partition size with minimum p-value: 6X6
p-value for MinP test:0.01
```

As stated above, the proposed method is distribution free. As such, the look-up table generated can be used with any data of the same size, as we show in the next example.

```
# generate data of size n:
x.2 = rnorm(n)
y.2 = x.2 + rnorm(n)

# carry out test using exactly the same null table as before.
res2 = Fast.independence.test(x.2,y.2,nt)
```

The main advantage of distribution free tests is that while standard permutation based tests performed on M null hypotheses with B permutations (in each test) require $M \times B$ independent calculations of the test statistics, distribution free tests require only M test statistics and B permutations, $M + B$ in total. This condition makes them ideal for scenarios where a large number of hypotheses are examined simultaneously.

For large values of N_A and table size B, the computation of the look-up table can still be cumbersome. The package vignette shows how this process can be parallelized.

k-sample tests

A special case of the independence problem is the k-sample problem: independence testing for continuous X and categorical Y, where Y has K different categories, $1, \ldots, K$. The null hypothesis can

be formulated as:

$$H_0 : F_X(x|y = 1) = F_X(x|y = 2) = \ldots = F_X(x|y = k), \forall x \tag{5}$$

We modify the *MinP* test statistic in Heller et al. (2016) to consider only partitions at the atom level. For example, instead of considering all possible partitions, we consider only partition points that are found on an equidistant grid of N_A points. Let p_m be the p-value of the test statistic that aggregates by summation or maximization all the LRT scores with partitions of size m of the real line, where a split point is possible only every N_A points. The atom based test statistic is

$$MinP = \min_{2 \leq m \leq m.max} (p_m) . \tag{6}$$

The code snippets below show how to analyze an example. The two vectors `X` and `Y` are of size `1000`. The entries in `Y` are the group labels: 500 zeros and 500 ones.

```
# generate null table
nt = hhg.univariate.ks.nulltable(c(500,500), #group structure
                        variant = 'KSample-Equipartition', #computationally
                        # efficient variant of the K-sample test.
                        mmax = 10, #m.max parameter
                        nr.atoms = 30, #number of atoms
                        nr.replicates = 10^4) #number of replicates

# run test
res = hhg.univariate.ks.combined.test(X,Y,nt)

# Shows Average LRT scores: AVG LRT score of
# m = 2,...,10 cell tables
res
```

As with independence tests in which many k-sample tests with the same sample sizes are carried out, the same null table can be used. We demonstrate this in the next example:

```
# generate sample with the same group structure -
# normal deviates with a shift between groups.
X2 = rnorm(length(Y),0,1)+1*Y

# perform test using the same null table.
res2 = hhg.univariate.ks.combined.test(X2,Y,nt)

# view results
res2
```

Simulations

We used simulations to assess the performance of the atom based tests in terms of both run time and power. Full source code for reproducing the simulation results, graphs and usage examples is found in the supplementary material, and at the GitHub repository (https://barakbri.github.io/HHG_large_sample_framework/).

All run times were measured using the CRAN package **rbenchmark** (Kusnierczyk, 2012). Run times were measured separately from power estimation, as simulation for power estimation has been parallelized via the **doRNG** package (Gaujoux, 2017). All run time experiments were done serially, without parallelization.

The test of independence

In order to assess the power of the *MinP* test statistics along with the actual run time required, we present four different scenarios of dependence between univariate random variables. Figure 3 shows the bivariate relationship along with a representative noisy sample: two monotone relationships (left panels), and two non-monotone relationships (right panels).

The presented methods were run with $N = 2500$, $N_A = 5, 10, 15, 30, 45, 60$, and with summation over all $m \times l$ or $m \times m$ partitions. This allows one to assess the affect of N_A on power and run time, along with the possible affect of summation over tables with a different number of partition points on

the two axes. Reducing the number of atoms N_A, allows one to estimate the breakdown of the method in terms of power.

The parameter $m.max$ was set to the package default ($m.max = 10$), except for $N_A = 5$ where $m.max$ was constrained by the number of atoms to 5. Heller et al. (2016) and Brill (2016) show by various simulation studies that the power of the test is quite robust to the selection of $m.max$ since it selects the optimal m, l adaptively; the test considers all partitions of sizes 2×2 to $m.max \times m.max$ and selects the best partition in the sense that its p-value is minimal.

Methods compared to in the simulation study are dCOV from CRAN package **energy**, MIC from CRAN package **minerva**, and HSIC from CRAN package **dHSIC**. We note that typically faster competitive methods are somewhat degenerate variations on these methods and would have much lower power than the originals.

Figure 4 shows the trade-off between runtime and power. We see that as the number of atoms increase, the run time increases but power is almost the same for 30 atoms or more. We set the default value for the number of atoms in the functions given in Table 1 to be the minimum between sample size and 40, promising the user a result in reasonable time regardless of sample size. Even for a small number of atoms such as $N_A = 10$, the MinP test power is similar to the MinP test with a high number of atoms. For the smallest number of atoms considered, $N_A = 5$, power may drop for complex signals which require fine partitions of the data. For the monotone settings considered, the method maintains competitive power also for $N_A = 5$.

Figure 5 shows the power as a function of sample size. The power of the test for the monotone settings is highest for dCov with the atoms based test a close second, similar to the best one achieved by the competitors. For the Circles setting, power for the $m \times m$ and $m \times l$ variant is similar. The setting is symmetric in practice, and the $m \times m$ variant enjoys higher power since it needs to account for fewer possible selections of partition size under the $MinP$ test statistic. Nevertheless, the loss of power is small when considering all $m \times l$ partitions of the data. For the Sine setting, power differs greatly between $m \times l$ and $m \times m$ variants. The setting is not symmetric in X and Y. One can see that the optimal partition of the plane to capture the dependence requires few partitions of the Y axis but many partitions of the X axis. See Brill (2016) for thorough simulations of different types of bivariate relationships: in the non-symmetric settings, considering $m \times l$ partitions of the data leads to substantial power gain; in symmetric settings, considering $m \times l$ partitions of the data leads to little power loss over considering only $m \times m$ partitions.

This simulation study demonstrated that the presented tests have a power advantage over competitors in settings where the underlying dependence is complex, i.e., settings where in multiple regions the joint density and the product of the marginal densities differ. For those settings, a fine partition of the sample space is optimal. Competing tests have tuning parameters, and the choice of tuning parameter can affect the power of the test. Specifically, tuning parameters in competitor methods include the degree of the L_p norm in dCOV, the kernel bandwidth in dHSIC, and the maximum partition size considered in MIC. The low power achieved by alternative methods in the 'Circles' and 'Sine' settings could be partially attributed to the use of the package default setting for the tuning parameters. The MinP procedure does not have a tuning parameter that can materially affect its performance, since the single best partition size is chosen in a data adaptive manner. For many other scenarios demonstrating this, see Heller et al. (2016) and Brill (2016).

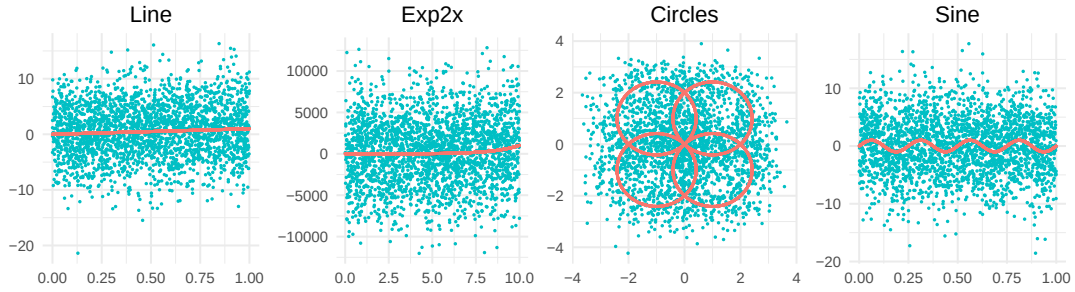


Figure 3: Four bivariate relationships (in red), along with a sample of $N = 2500$ observations in blue.

Figure 6 shows the run-time as a function of sample size and atom size. In the left panel, the total time to carry out a test with 1000 permutations is computed for 4 methods: $MinP$ (including null table construction) with $S^{m \times l}$ test statistic ($N_A = 45$), dCov, HSIC, and MIC (including null table construction). We see that the $O\left(N_A^4\right)$ portion of the computation time of $MinP$ takes the largest portion of the computation, being greater than the $O\left(Nlog\left(N\right)\right)$ part. As such, computation time is constant for all N values considered. For other tests considered, computation time might be shorter for smaller samples, but the computational complexity is quadratic (or above) and hence more

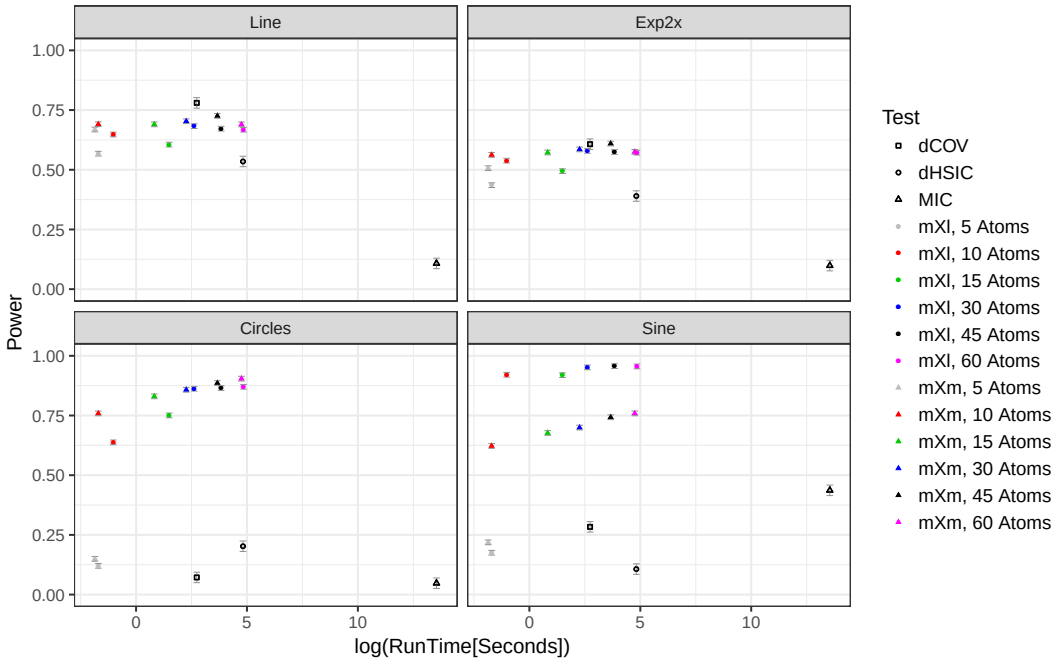


Figure 4: Power and logarithm of run time for $N = 2500$. Run times for each method were computed for a full procedure, including tabulation of the null hypothesis and parameter selection. Specifically, for **dCov** and **HSIC**, the number of permutations was a single run of the test function with 1000 permutations (as computation of kernel distance matrices is done only once for the original data); for **MinP**, runtime consists of 1000 computations for the null table and the test statistic (computation of statistics under the null hypothesis, efficient tabulation of marginal distributions, and $MinP$, computation of $MinP$ for the tested dataset, which was then used for computing the P-value); for **MIC**, run times consist of 1001 computations of the test statistic for the dataset tested and distribution under the null. For a pre-computed null table of size $N = 2500$, running times for MIC and $MinP$ would be one thousandth of the time shown above, since these two tests are distribution free. Power computation is based on 10000 datasets for the MinP procedure and 2000 datasets for alternative tests. Errors bars show 95% confidence intervals.

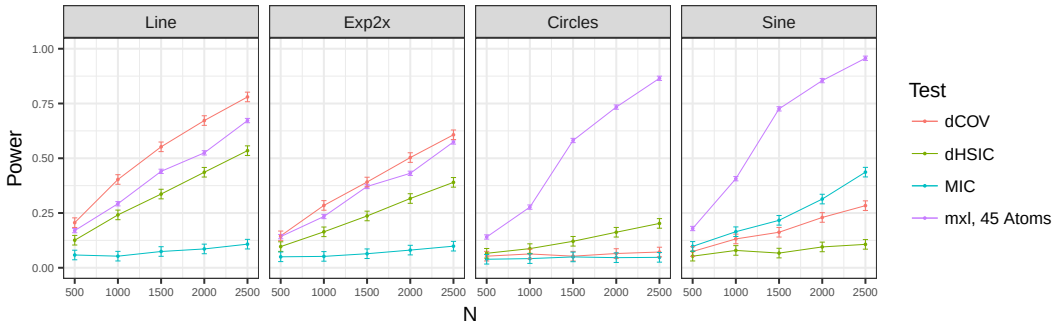


Figure 5: Power versus sample size, in each of the scenarios of Figure 3. Based on 10000 datasets for the MinP procedure, and 2000 datasets for alternative tests. Error bars show 95% confidence intervals.

computationally demanding than *MinP* for larger sample sizes. The right panel presents the run time versus number of atoms for $N = 300$ and $m.max = 10, 15$. The relationship is clearly linear on the log scale. The linear fit shows that the computational complexity is indeed to the fourth order in terms of N_A. Overall, the algorithm has a computational complexity which is $\mathcal{O}\left(N \log N + N_A^4\right)$.

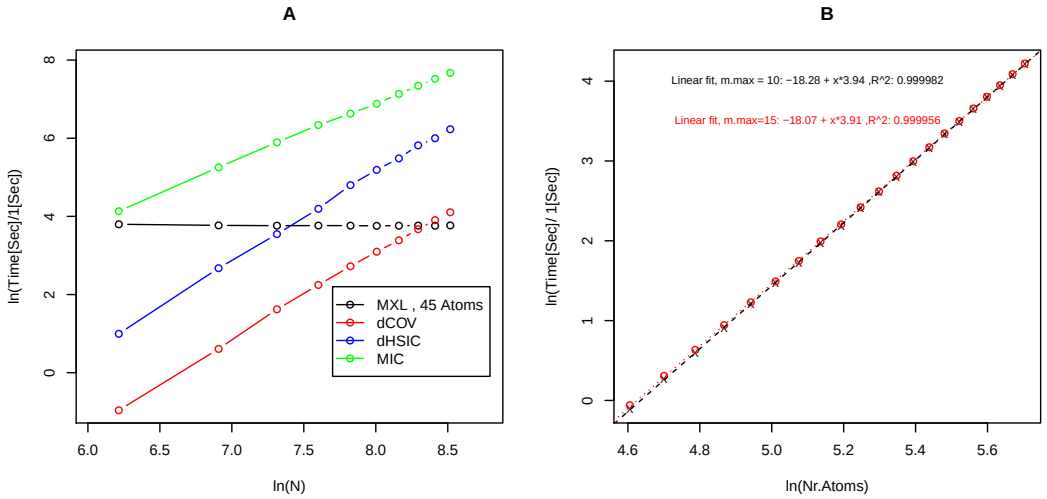


Figure 6: A (Left): Runtime analysis on log scale for different methods for sample sizes $N = 500, 1000, \ldots, 5000$. B (Right): Runtime comparison on log scale, compared with number of atoms. Number of atoms varied between 100 and 300. Computations performed were $S^{m\times l}$ with $mmax = 10$ (black) and $mmax = 15$ (red). Linear fit verifies known computational complexity, $\mathcal{O}\left(N \log N + N_A^4\right)$. Time measurements were performed with 100 and 50 repetitions for left and right panels, respectively.

The k-sample test

In order to assess the power of the *MinP* test statistics, we present a simulation study. Methods compared were the energy two sample test, given by `eqdist.etest` from package **energy**, and the kernel MMD test, given by `kmmd` from **kernlab**.

Let $\sum_{i=1}^{K} p_i N(\mu_i, \sigma_i^2)$ denote the mixture distribution of K Gaussian distributions with means μ_i and variances σ_i^2. We examine the following three settings: the shifted normal, sampling from $N\left(0, 1^2\right)$ and from $N\left(0.075, 1^2\right)$; the 2 component normal mixture, sampling from $0.7N\left(0, 1^2\right) + 0.3N\left(0, 8^2\right)$ and $0.7N\left(0.15, 1^2\right) + 0.3N\left(0, 8^2\right)$; and the 3 component normal mixture, sampling from $f\left(X|Y = 1\right) = \frac{1}{3}N\left(-4, 1\right) + \frac{1}{3}N\left(0, 1\right) + \frac{1}{3}N\left(4, 1\right)$, and $f\left(X|Y = 2\right) = \frac{1}{3}N\left(-4, 0.8^2\right) + \frac{1}{3}N\left(0, 0.8^2\right) + \frac{1}{3}N\left(4, 0.8^2\right)$.

Table 2: A power comparison of the *MinP* test with a different number of atoms, the energy test, and the MMD test. For the three data generations, the power is given in rows 1-3, as well as the run time in row 4. Datasets are samples with two equal groups of size 2500 each to a total of $N = 5000$. For each setting, we simulate 10000 datasets for the MinP procedure and 2000 datasets for alternative tests, with 1000 permutations for each test. Run times are measured over 100 repetitions.

	10 Atoms	25 Atoms	50 Atoms	ENERGY	KMMD
Normal, Shift	0.65	0.70	0.70	0.72	0.53
Mixture, 2 Components	0.72	0.75	0.74	0.21	0.36
Mixture, 3 Components	0.86	0.94	0.94	0.09	0.07
Run time (seconds)	2.96	3.09	3.23	5.22	146.51

Table 2 provides the power result for competitor tests, along with the *MinP* test statistic for $N_A = 10, 25, 50$. The *MinP* test has excellent power in comparison with energy and MMD, and the run time is lower. For additional simulations that include many additional settings, see Brill (2016). In general, the proposed k-sample test performs best when the difference between the distributions is complex and specifically when the density plots of the two distributions intersect multiple times, see Heller et al. (2016) for details.

Discussion

We presented computationally fast and powerful novel tests for detecting complex dependencies among univariate random variables. These tests can be used in the framework suggested in Heller and Heller (2016) in order to construct computationally-fast, distribution-free multivariate tests for powerful identification of complex multivariate relationships. Briefly, one may reduce tests of independence between multivariate X and Y to univariate ones by several methods, such as choosing a reference point in X and Y and testing whether distances between observations and reference points are associated in X and Y, or by choosing a direction and projecting X and Y on it. Heller and Heller (2016) discuss methods of aggregation over several reference points. If the univariate tests utilized are computationally efficient for large sample sizes and consistent, the resulting multivariate tests will also be consistent and computationally efficient.

Using our approach for obtaining a look-up table for $MinP$, the null distribution of any test statistic that combines individual p-values from distribution-free tests can be efficiently tabulated. Steps A to E depicted in Figure 2 can be performed with $S^{m\times l}$ columns replaced by other rank based test statistics $T_1, \ldots, T_M$, and the $MinP$ can be replaced by any p-value combining function $f(p_1, \ldots, p_M)$. For example, package `coin` contains various rank based tests to detect shift or scale differences across distributions. The null distribution tabulation we presented can be useful for constructing a look-up table for a distribution-free combined test for shift and scale.

Appendix A

Let N be the number of observations, and N_A be the number of atoms such that the size of an atom is given by an integer $A = N/N_A$. The locations of splits between atoms are thus given by $T_i = i \cdot A$ for $i \in \{0, 1, 2, \ldots, N_A\}$. If N is not a complete multiple of N_A, we define the locations of splits between atoms to be $T_i = \lfloor i \cdot N/N_A \rfloor$.

Step I (compute ECDF): The empirical CDF can be computed in $O(N_A^2 + N)$ time for all possible split locations: $\hat{F}(T_i, T_j), 0 \le i, j, \leqq N_A$. First, execute Algorithm 1 to compute the matrix $A_{r,s}$.

Algorithm 2 Compute ECDF at all inter-atoms split points

procedure COMPUTEECDF
  Initialize $A_{i,j} \leftarrow 0$, for $0 \le i, j, \le N_A$
  Initialize $B_{i,j} \leftarrow 0$, for $0 \le i, j, \le N_A$
  for $i = 1$ to N **do**
    $AtomRank_x \leftarrow \lceil N_A/N \cdot rank(x_i) \rceil$
    $AtomRank_y \leftarrow \lceil N_A/N \cdot rank(y_i) \rceil$
    $B_{AtomRank_x, AtomRank_y} \leftarrow B_{AtomRank_x, AtomRank_y} + 1$
  for $s = 0$ to N_A **do**
    for $r = 0$ to N_A **do**
      $A_{r,s} \leftarrow B_{r,s-1} + B_{r-1,s} - B_{r-1,s-1} + B_{r,s}$
      $B_{r,s} \leftarrow A_{r,s}$

The empirical cumulative distribution function at the split point given by ranks (T_i, T_j) is given by $\hat{F}(T_i, T_j) = A_{i,j}/N$. For a cell of borders $[T_{i1}, T_{i2}] \times [T_{j1}, T_{j2}]$, the expected number of observations is given by:

$$E(T_{i1}, T_{i2}, T_{j1}, T_{j2}) = (T_{i2} - T_{i1}) \cdot (T_{j2} - T_{j1})/N \tag{7}$$

Once the empirical CDF has been tabulated, the observed counts of the cell can be calculated in constant time:

$$O(T_{i1}, T_{i2}, T_{j1}, T_{j2}) = A_{i2,j2} - A_{i1,j2} - A_{i2,j1} + A_{i1,j1} \tag{8}$$

Step II (Aggregate LRT scores of all cells of given size): A cell of the sample space $[T_{i1}, T_{i2}] \times [T_{j1}, T_{j2}]$ is given by its boundaries $i1, i2, j1, j2$. We define a cell to be 'top' or 'bottom' cell if $i2 = N$ or $i1 = 0$. A cell is considered 'right' or 'left' if $j2 = N$ or $j1 = 0$. If a cell is in both categories, it is called 'corner'. If a cell is not 'top','bottom','left', or 'right', it is called a 'center' cell. Let $Type(i1, i2, j1, j2)$ be a function taking the borders of a cell and returning 1 for 'center' cells, 2 for 'top' or 'bottom', 3 for 'left' or 'right', and 4 for 'corner' cells.

We define the width of the cell to be $w(i1, i2) = i2 - i1$ and the height of the cell to be $h(j1, j2) = j2 - j1$. For each size and type, we sum up the scores of that size and type.

Algorithm 2 computes the array $S_{t,w,h}$, which stores the sum of scores of the form $O \cdot ln\left(\frac{O}{E}\right)$ for all cells of type t, width w, and height h.

Algorithm 3 Aggregate contribution to statistic by cell type and size

```
1:  procedure AGGREGATECELLSCORES
2:      Initialize S_{t,w,h} ← 0, ∀t ∈ [1,4], w,h ∈ [1, N_A − 1]
3:      for i1 = 0 to N_A − 1 do
4:          for i2 = i1 + 1 to N_A do
5:              w_0 ← w(i2, i1)
6:              for j1 = 0 to N_A − 1 do
7:                  for j2 = j1 + 1 to N_A do
8:                      t ← Type(i1, i2, j1, j2)
9:                      h_0 ← h(j2, j1)
10:                     o ← O(i1, i2, j1, j2)
11:                     e ← E(i1, i2, j1, j2)
12:                     S_{t,w_0,h_0} ← S_{t,w_0,h_0} + o · ln(o/e)
```

Step III (Computing $S^{m\times l}$): Next, we aggregate over S_{t,w_0,h_0}, over all cell sizes and types, to compute $S^{m\times l}$ with $m,l \in [2, m.max]$.

The number of partitions in which a cell is found is given by the number of possible ways to select additional partitioning locations around that cell, denoted by $n(t,w,h,m,l)$ (2). For example, a 'center' cell requires $m-3$ choices of additional X partition points and $l-3$ additional Y partition points to fully define a $m \times l$ partition. Hence a 'center' cell of size w,h is found in $\binom{N_A-w-2}{m-3} \cdot \binom{N_A-h-2}{l-3}$ partitions. Note that the number of tables in which a cell is found does not depend on its edge locations but only on its size and type. Thus, the contribution for $S^{m\times l}$ of all center cells of size $w \times h$ atoms is given by the sum of their $O \cdot ln\left(\frac{O}{E}\right)$ scores, multiplied by $\binom{N_A-w-2}{m-3} \cdot \binom{N_A-h-2}{l-3}$. The differentiation by cell type is needed since a cell bordering the edge of the sample space requires only one partition point, thus allowing the selection of an extra partition point.

The final step is given in algorithm 3. Algorithm 3 computes $S^{m\times l}$ by summing over all values of the array $S_{t,w,h}$ with weights $n(t,w,h,m,l)$.

Algorithm 4 Compute $S^{m\times l}$ for all $m,l \in [2, m.max]$

```
1:  procedure COMPUTESTATISTIC
2:      Initialize S^{m×l} ← 0, ∀m,l ∈ [2, m.max]
3:      for m = 2 to m.max do
4:          for l = 2 to m.max do
5:              for t = 1 to 4 do
6:                  for w = 1 to N_A − 1 do
7:                      for h = 1 to N_A − 1 do
8:                          S^{m×l} ← S^{m×l} + n(t,w,h,m,l) · S_{t,w,h}
```

Appendix B

In Table 3 we list the additional available tests in the HHG package. Specifically, the multivariate tests of Heller et al. (2013) and the univariate test of Heller et al. (2016).

Bibliography

D. Albanese, M. Filosi, R. Visintainer, S. Riccadonna, G. Jurman, and C. Furlanello. minerva and minepy: a c engine for the mine suite and its r, python and matlab wrappers. *Bioinformatics*, 29(3): 407–408, 2013. URL https://doi.org/10.1093/bioinformatics/bts707. [p424]

B. Brill. Scalable non-parametric tests of independence. Master's thesis, Tel Aviv University, 2016. URL http://primage.tau.ac.il/libraries/theses/exeng/free/2899741.pdf. [p426, 431, 433]

Table 3: Nonparametric tests previously available in the **HHG** package for small to moderate sample sizes

Function Name	Description
`hhg.test`	Implements the multivariate test of independence described in Heller et al. (2013), testing for the independence of two random vectors $\vec{X}$ and $\vec{Y}$,of dimensionality p and q, respectively.
`hhg.test.2.sample` `hhg.test.k.sample`	Adaptation of the above procedure to the k-sample problem (equality of distributions). Given a multivariate measurement $\vec{X}$ and a group factor variable $\vec{Y}$, test whether the distributions $\vec{X}\|Y = 1, \vec{X}\|Y = 2, \ldots, \vec{X}\|Y = k$ are equal. This is the nonparametric extension of the MANOVA problem (sensitive not only to shifts in means).
`hhg.univariate.ind.combined.test`	Univariate test of independence for problem (1), as described in Heller et al. (2016).
`hhg.univariate.ks.combined.test`	Test for univariate equality of distributions (i.e., $X\|Y = 1\ X\|Y = 2\ \ldots\ X\|Y = k$), as described in Heller et al. (2016). This is the nonparametric extensions of the ANOVA problem, which is sensitive to any difference in distribution (not only to shifts in means).

K. P. Chwialkowski, A. Ramdas, D. Sejdinovic, and A. Gretton. Fast two-sample testing with analytic representations of probability measures. In *Advances in Neural Information Processing Systems*, pages 1981–1989, 2015. [p425]

M. Filosi, R. Visintainer, and D. Albanese. *minerva: Maximal Information-Based Nonparametric Exploration for Variable Analysis*, 2017. URL https://CRAN.R-project.org/package=minerva. R package version 1.4.7. [p424]

R. Gaujoux. *doRNG: Generic Reproducible Parallel Backend for 'foreach' Loops*, 2017. URL https://CRAN.R-project.org/package=doRNG. R package version 1.6.6. [p430]

A. Gretton, K. Fukumizu, C. H. Teo, L. Song, B. Schölkopf, and A. J. Smola. A kernel statistical test of independence. In *Advances in neural information processing systems*, pages 585–592, 2008. [p424]

A. Gretton, K. M. Borgwardt, M. J. Rasch, B. Schölkopf, and A. Smola. A kernel two-sample test. *Journal of Machine Learning Research*, 13(Mar):723–773, 2012a. [p424, 425]

A. Gretton, D. Sejdinovic, H. Strathmann, S. Balakrishnan, M. Pontil, K. Fukumizu, and B. K. Sriperumbudur. Optimal kernel choice for large-scale two-sample tests. In F. Pereira, C. J. C. Burges, L. Bottou, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 25*, pages 1205–1213. Curran Associates, Inc., 2012b. URL http://papers.nips.cc/paper/4727-optimal-kernel-choice-for-large-scale-two-sample-tests.pdf. [p425]

F. E. Harrell Jr, with contributions from Charles Dupont, and many others. *Hmisc: Harrell Miscellaneous*, 2018. URL https://CRAN.R-project.org/package=Hmisc. R package version 4.1-1. [p424]

J. Hausser and K. Strimmer. Entropy inference and the james-stein estimator, with application to nonlinear gene association networks. *Journal of Machine Learning Research*, 10(Jul):1469–1484, 2009. [p424]

J. Hausser and K. Strimmer. *entropy: Estimation of Entropy, Mutual Information and Related Quantities*, 2014. URL https://CRAN.R-project.org/package=entropy. R package version 1.2.1. [p424]

R. Heller and Y. Heller. Multivariate tests of association based on univariate tests. In D. D. Lee, M. Sugiyama, U. V. Luxburg, I. Guyon, and R. Garnett, editors, *Advances in Neural Information*

Processing Systems 29, pages 208–216. Curran Associates, Inc., 2016. URL http://papers.nips.cc/paper/6220-multivariate-tests-of-association-based-on-univariate-tests.pdf. [p434]

R. Heller, Y. Heller, and M. Gorfine. A consistent multivariate test of association based on ranks of distances. *Biometrika*, 100(2):503–510, 2013. URL https://doi.org/10.1093/biomet/ass070. [p435, 436]

R. Heller, Y. Heller, S. Kaufman, B. Brill, and M. Gorfine. Consistent distribution-free k-sample and independence tests for univariate random variables. *Journal of Machine Learning Research*, 17(29): 1–54, 2016. [p424, 425, 427, 430, 431, 433, 435, 436]

W. Hoeffding. A non-parametric test of independence. *The annals of mathematical statistics*, pages 546–557, 1948. URL https://doi.org/10.1214/aoms/1177730150. [p424]

C. Huang and X. Huo. An efficient and distribution-free two-sample test based on energy statistics and random projections. *arXiv preprint arXiv:1707.04602*, 2017. [p425]

X. Huo and G. J. Székely. Fast computing for distance covariance. *Technometrics*, 58(4):435–447, 2016. URL https://doi.org/10.1080/00401706.2015.1054435. [p425]

B. Jiang, C. Ye, and J. S. Liu. Non-parametric *K*-sample tests via dynamic slicing. *Journal of the American Statistical Association*, 110(510):642–653, 2015. URL https://doi.org/10.1080/01621459.2014.920257. [p424, 425]

W. Jitkrittum, Z. Szabó, K. Chwialkowski, and A. Gretton. Interpretable distribution features with maximum testing power. In D. D. Lee, M. Sugiyama, U. V. Luxburg, I. Guyon, and R. Garnett, editors, *Advances in Neural Information Processing Systems 29*, pages 181–189. Curran Associates, Inc., 2016a. URL http://papers.nips.cc/paper/6148-interpretable-distribution-features-with-maximum-testing-power.pdf. [p425]

W. Jitkrittum, Z. Szabó, and A. Gretton. An adaptive test of independence with analytic kernel embeddings. *arXiv preprint arXiv:1610.04782*, 2016b. [p425]

A. Karatzoglou, A. Smola, and K. Hornik. *kernlab: Kernel-Based Machine Learning Lab*, 2016. URL https://CRAN.R-project.org/package=kernlab. R package version 0.9-25. [p]

B. B. . S. Kaufman, based in part on an earlier implementation by Ruth Heller, and Y. Heller. *HHG: Heller-Heller-Gorfine Tests of Independence and Equality of Distributions*, 2017. URL https://CRAN.R-project.org/package=HHG. R package version 2.2. [p]

W. Kusnierczyk. *rbenchmark: Benchmarking routine for R*, 2012. URL https://CRAN.R-project.org/package=rbenchmark. R package version 1.0.0. [p430]

D. Lopez-Paz, P. Hennig, and B. Schölkopf. The randomized dependence coefficient. In *Advances in neural information processing systems*, pages 1–9, 2013. [p]

P. E. Meyer. *infotheo: Information-Theoretic Measures*, 2014. URL https://CRAN.R-project.org/package=infotheo. R package version 1.2.0. [p424]

P. E. Meyer, F. Lafitte, and G. Bontempi. minet: Ar/bioconductor package for inferring large transcriptional networks using mutual information. *BMC bioinformatics*, 9(1):461, 2008. URL https://doi.org/10.1186/1471-2105-9-461. [p424]

P. E. Meyer, F. Lafitte, and G. Bontempi. *minet: Mutual Information NETworks*, 2017. URL http://minet.meyerp.com. R package version 3.36.0. [p424]

Microsoft and S. Weston. *foreach: Provides Foreach Looping Construct for R*, 2017. URL https://CRAN.R-project.org/package=foreach. R package version 1.4.4. [p]

L. Paninski. Estimation of entropy and mutual information. *Neural computation*, 15(6):1191–1253, 2003. URL https://doi.org/10.1162/089976603321780272. [p424]

N. Pfister and J. Peters. *dHSIC: Independence Testing via Hilbert Schmidt Independence Criterion*, 2017. URL https://CRAN.R-project.org/package=dHSIC. R package version 2.0. [p424]

N. Pfister, P. Bühlmann, B. Schölkopf, and J. Peters. Kernel-based tests for joint independence. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 80(1):5–31, 2018. URL https://doi.org/10.1111/rssb.12235. [p424]

D. N. Reshef, Y. A. Reshef, H. K. Finucane, S. R. Grossman, G. McVean, P. J. Turnbaugh, E. S. Lander, M. Mitzenmacher, and P. C. Sabeti. Detecting novel associations in large data sets. *science*, 334(6062): 1518–1524, 2011. URL https://doi.org/10.1126/science.1205438. [p424]

M. L. Rizzo and G. J. Szekely. *energy: E-Statistics: Multivariate Inference via the Energy of Data*, 2017. URL https://CRAN.R-project.org/package=energy. R package version 1.7-2. [p424]

C. Spearman. The proof and measurement of association between two things. *The American journal of psychology*, 15(1):72–101, 1904. URL https://doi.org/10.2307/1412159. [p]

G. J. Székely and M. L. Rizzo. Testing for equal distributions in high dimension. *InterStat*, 5(16.10), 2004. [p424, 425]

G. J. Székely, M. L. Rizzo, N. K. Bakirov, et al. Measuring and testing dependence by correlation of distances. *The Annals of Statistics*, 35(6):2769–2794, 2007. URL https://doi.org/10.1214/009053607000000505. [p424]

G. J. Székely, M. L. Rizzo, et al. Brownian distance covariance. *The annals of applied statistics*, 3(4): 1236–1265, 2009. URL https://doi.org/10.1214/09-aoas312. [p424]

H. Wickham and W. Chang. *ggplot2: Create Elegant Data Visualisations Using the Grammar of Graphics*, 2016. URL https://CRAN.R-project.org/package=ggplot2. R package version 2.2.1. [p]

C. Ye, B. Jiang, X. Zhang, and J. S. Liu. *dslice*: an R package for nonparametric testing of associations with application in qtl and gene set analysis. *Bioinformatics*, 31(11):1842–1844, 2015. URL https://doi.org/10.1093/bioinformatics/btv021. [p]

A. Zeileis, K. Hornik, A. Smola, and A. Karatzoglou. kernlab-an s4 package for kernel methods in r. *Journal of statistical software*, 11(9):1–20, 2004. URL https://doi.org/10.18637/jss.v011.i09. [p424]

Q. Zhang, S. Filippi, A. Gretton, and D. Sejdinovic. Large-scale kernel methods for independence testing. *Statistics and Computing*, 28(1):113–130, 2018. URL https://doi.org/10.1007/s11222-016-9721-7. [p425]

J. Zhao and D. Meng. Fastmmd: Ensemble of circular discrepancy for efficient two-sample test. *Neural computation*, 27(6):1345–1372, 2015. URL https://doi.org/10.1162/neco_a_00732. [p425]

Barak Brill
Department of Statistics and Operations Research, Tel-Aviv University
Tel-Aviv
Israel
barakbri@mail.tau.ac.il

Yair Heller
Tel-Aviv
Israel
heller.yair@gmail.com

Ruth Heller
Department of Statistics and Operations Research, Tel-Aviv University
Tel-Aviv
Israel
ruheller@gmail.com

Simple Features for R: Standardized Support for Spatial Vector Data

by Edzer Pebesma

Abstract Simple features are a standardized way of encoding spatial vector data (points, lines, polygons) in computers. The **sf** package implements simple features in R, and has roughly the same capacity for spatial vector data as packages **sp**, **rgeos**, and **rgdal**. We describe the need for this package, its place in the R package ecosystem, and its potential to connect R to other computer systems. We illustrate this with examples of its use.

What are simple features?

Features can be thought of as "things" or objects that have a spatial location or extent; they may be physical objects like a building, or social conventions like a political state. *Feature geometry* refers to the spatial properties (location or extent) of a feature, and can be described by a point, a point set, a linestring, a set of linestrings, a polygon, a set of polygons, or a combination of these. The *simple* adjective of *simple features* refers to the property that linestrings and polygons are built from points connected by *straight* line segments. Features typically also have other properties (temporal properties, color, name, measured quantity), which are called *feature attributes*. Not all spatial phenomena are easy to represent by "things or objects:" continuous phenoma such as water temperature or elevation are better represented as *functions* mapping from continuous or sampled space (and time) to values (Scheider et al., 2016), and are often represented by *raster* data rather than vector (points, lines, polygons) data.

Simple feature access (Herring, 2011) is an international standard for representing and encoding spatial data, dominantly represented by point, line, and polygon geometries (ISO, 2004). It is widely used e.g. by spatial databases (Herring, 2010), GeoJSON (Butler et al., 2016), GeoSPARQL (Perry and Herring, 2012), and open source libraries that empower the open source geospatial software landscape including GDAL (Warmerdam, 2008), GEOS (GEOS Development Team, 2017), and liblwgeom (a PostGIS component, Obe and Hsu (2015)).

The need for a new package

The **sf** (Pebesma, 2018) package is an R package for reading, writing, handling, and manipulating simple features in R, reimplementing the vector (points, lines, polygons) data handling functionality of packages **sp** (Pebesma and Bivand, 2005; Bivand et al., 2013), **rgdal** (Bivand et al., 2017) and **rgeos** (Bivand and Rundel, 2017). However, **sp** has some 400 direct reverse dependencies, and a few thousand indirect ones. Why was there a need to write a package with the potential to replace it?

First of all, at the time of writing **sp** (2003) there was no standard for simple features, and the ESRI shapefile was by far the dominant file format for exchanging vector data. The lack of a clear (open) standard for shapefiles, the omnipresence of "bad" or malformed shapefiles, and the many limitations of the ways it can represent spatial data adversely affected **sp**, for instance in the way it represents holes in polygons, and a lack of discipline to register holes with their enclosing outer ring. Such ambiguities could influence plotting of data, or communication with other systems or libraries.

The *simple feature access* standard is now widely adopted, but the **sp** package family has to make assumptions and do conversions to load them into R. This means that you cannot round-trip data, e.g., loading data in R, manipulating them, exporting them and getting the same geometries back. With **sf**, this is no longer a problem.

A second reason was that external libraries heavily used by R packages for reading and writing spatial data (GDAL) and for geometrical operations (GEOS) have developed stronger support for the simple feature standard.

A third reason was that the package cluster now known as the **tidyverse** (Wickham, 2017, 2014), which includes popular packages such as **dplyr** (Wickham et al., 2017) and **ggplot2** (Wickham, 2016), does not work well with the spatial classes of **sp**:

- **tidyverse** packages assume objects not only *behave* like `data.frames` (which **sp** objects do by providing methods), but *are* `data.frames` in the sense of being a list with equally sized column vectors, which **sp** does not do.

- attempts to "tidy" polygon objects for plotting with **ggplot2** ("fortify") by creating `data.frame` objects with records for each polygon node (vertex) were neither robust nor efficient.

A simple (S3) way to store geometries in `data.frame` or similar objects is to put them in a geometry list-column, where each list element contains the geometry object of the corresponding record, or `data.frame` "row"; this works well with the **tidyverse** package family.

Conventions

Classes

The main classes introduced by package **sf** are

`"sf"`: a `data.frame` (or `tbl_df`) with one or more geometry list-columns, and an attribute `sf_column` indicating the *active* geometry list-column of class `sfc`,

`"sfc"`: a list-column with a set of feature geometries

`"sfg"`: element in a geometry list-column, a feature geometry

`"crs"`: a coordinate reference system, stored as attribute of an `"sfc"`

Except for `"sfg"`, all these classes are implemented as lists. Objects of class `"sfg"` are subtyped according to their class, classes have the following storage form:

`POINT`: numeric vector with a single point

`MULTIPOINT`: numeric matrix with zero or more points in rows

`LINESTRING`: numeric matrix with zero or more points in rows

`POLYGON`: list with zero or more numeric matrices (points as rows); polygon outer ring is followed by zero or more inner rings (holes)

`MULTILINESTRING`: list with zero or more numeric matrices, points in rows

`MULTIPOLYGON`: list of lists following the `POLYGON` structures

`GEOMETRYCOLLECTION`: list of zero or more of the (classed) structures above

All geometries have an empty form, indicating the missing (or `NA`) equivalent for a geometry.

Functions and methods

Category	Functions
binary predicates	`st_contains`, `st_contains_properly`, `st_covered_by`, `st_covers`, `st_crosses`, `st_disjoint`, `st_equals`, `st_equals_exact`, `st_intersects`, `st_is_within_distance`, `st_within`, `st_touches`, `st_overlaps`
binary operations	`st_relate`, `st_distance`
unary operations	`st_dimension`, `st_area`, `st_length`, `st_is_longlat`, `st_is_simple`, `st_is_valid`, `st_jitter`, `st_geohash`, `st_geometry_type`
miscellaneous	`st_sample`, `st_line_sample`, `st_join`, `st_interpolate_aw`, `st_make_grid`, `st_graticule`, `sf_extSoftVersion`, `rawToHex`, `st_proj_info`
setters	`st_set_agr`, `st_set_crs`
constructors	`st_sfc`, `st_sf`, `st_as_sf`, `st_as_sfc`, `st_point`, `st_multipoint`, `st_linestring`, `st_multilinestring`, `st_polygon`, `st_multipolygon`, `st_geometrycollection`, `st_combine`, `st_bind_cols`
in- & output	`st_read`, `st_read_db`, `st_write`, `st_write_db`, `read_sf`, `write_sf`, `st_drivers`, `st_layers`
plotting	`st_viewport`, `st_wrap_dateline`, `sf.colors`

Table 1: Functions provided by package **sf**, arranged by functional category.

Functions are listed in Table 1. Some functions or methods operate on both attributes and geometries, e.g. `aggregate` and `summarise` compute grouped statistics and group (union) corresponding

class	methods
sfg	as.matrix, c, coerce, format, head, Ops, plot, print, st_as_binary, st_as_grob, st_as_text, st_transform, st_coordinates, st_geometry, st_boundary, st_buffer, st_centroid, st_convex_hull, st_difference, st_intersection, st_line_merge, st_make_valid, st_node, st_point_on_surface, st_polygonize, st_segmentize, st_simplify, st_split, st_sym_difference, st_triangulate, st_union, st_voronoi, st_cast, st_collection_extract, st_is, st_zm
sfc	[, [<-, as.data.frame, c, coerce, format, Ops, print, rep, st_as_binary, st_as_text, st_bbox, st_coordinates, st_crs, st_crs<-, st_geometry, st_precision, st_set_precision, str, summary, st_transform, st_boundary, st_buffer, st_centroid, st_convex_hull, st_difference, st_intersection, st_line_merge, st_make_valid, st_node, st_point_on_surface, st_polygonize, st_segmentize, st_simplify, st_split, st_sym_difference, st_triangulate, st_union, st_voronoi, st_cast, st_collection_extract, st_is, st_zm, obj_sum, type_sum
sf	[, [[<-, $<-, aggregate, cbind, coerce, merge, plot, print, rbind, st_agr, st_agr<-, st_bbox, st_coordinates, st_crs, st_crs<-, st_geometry, st_geometry<-, st_precision, st_set_precision, st_transform, st_boundary, st_buffer, st_centroid, st_convex_hull, st_difference, st_intersection, st_line_merge, st_make_valid, st_node, st_point_on_surface, st_polygonize, st_segmentize, st_simplify, st_split, st_sym_difference, st_triangulate, st_union, st_voronoi, st_cast, st_collection_extract, st_is, st_zm, anti_join, arrange, distinct, filter, full_join, gather, group_by, inner_join, left_join, nest, mutate, rename, right_join, sample_frac, sample_n, select, semi_join, separate, slice, spread, summarise, transmute, ungroup, unite
crs	$, is.na, Ops, print, st_as_text, st_crs

Table 2: Methods for **sf** classes: colours indicate geometry operation, class manipulation, tidyverse; and Ops refers to (a subset of) arithmetical operations.

geometries, and `st_interpolate_aw` carries out area-weighted interpolation (Do et al., 2015). The function `st_join` joins pairs of tables based on a geometrical predicate such as `st_intersects`.

Generic methods for `sf` objects are listed in Table 2. Many of them are for creation, extraction, and conversion, and many of them are not needed for every-day work. Where possible, methods act either on a geometry (`sfg`), a geometry set (`sfc`), or a geometry set with attributes (`sf`), Methods return an object of identical class. Coordinate reference systems (CRS) carry through all operations, except for `st_transform`, which transforms coordinates from one reference system into another, and hence, the CRS changes.

Serialisations

The simple feature access defines two serialisation standards: well-known-text (WKT) and well-known-binary (WKB). Well-known text is the default print form and `sfc` columns can be read from WKT `character` vectors, using `st_as_sfc`:

```
> library(sf)
Linking to GEOS 3.5.1, GDAL 2.1.2, proj.4 4.9.3
> (pt <- st_point(c(0,1)))
POINT (0 1)
> (pol <- st_polygon(list(rbind(c(0,0), c(1,0), c(1,1), c(0,1), c(0,0)))))
POLYGON ((0 0, 1 0, 1 1, 0 1, 0 0))
> st_as_sfc("POINT(0 1)") # returns sfc:
Geometry set for 1 feature
geometry type:  POINT
dimension:      XY
bbox:           xmin: 0 ymin: 1 xmax: 0 ymax: 1
epsg (SRID):    NA
proj4string:    NA
POINT (0 1)
```

R native simple feature geometries can be written to WKB using `st_as_binary`:

```
> st_as_binary(st_point(c(0,1)))
 [1] 01 01 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 f0 3f
```

```
> st_as_binary(st_polygon(list(rbind(c(0,0), c(1,0), c(1,1), c(0,1), c(0,0)))))
 [1] 01 03 00 00 00 01 00 00 00 05 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
[26] 00 00 00 00 00 00 00 00 00 00 f0 3f 00 00 00 00 00 00 00 00 00 00 00 00 00
[51] 00 f0 3f 00 00 00 00 00 00 f0 3f 00 00 00 00 00 00 00 00 00 00 00 00 00 00
[76] f0 3f 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
```

Similarly, binary encoded geometries can be read back using `st_as_sfc`.

All communication to and from the underlying libraries GDAL, GEOS and liblwgeom, as well as direct reading and writing of geometry BLOBs in spatial databases, uses binary serialisation and deserialisation, written in C++. This makes code not only fast but also robust: for all possible geometry classes, a single interface is used to communicate to a variety of endpoints.

Spherical geometry

The GEOS library provides a large set of operations for data in a two-dimensional space. For unprojected, geographic data the coordinates are longitude and latitude, and describe points on a sphere (or ellipsoid), not on a plane. The **sf** package allows such data to be passed to all geometric operations, but will emit a message if this happens through GEOS, assuming a flat Earth. For the functions `st_area`, `st_length`, `st_distance`, `st_is_within_distance`, and `st_segmentize` specialized spherical functions, taken from **lwgeom** (Pebesma), are used. The advantage of this package e.g. over **geosphere** (Hijmans, 2016a) is that it supports simple features for distance calculations, where geosphere only computes distances between points. Function st_sample has been modified to work for spherical coordinates when sampling points on an area over a sphere.

It would be nice to get a (more) complete set of functions working for spherical geometry. Potential candidate libraries to be used for this include **s2** (Rubak and Ooms, 2017), liblwgeom (part of PostGIS), CGAL (Fabri and Pion, 2009), and boost.Geometry.

Tidy tools

During the development of **sf**, considerable effort was put into making the new data structures work with the tidyverse. This was done by providing methods for **dplyr** verbs (Table 2), and by helping develop a **ggplot2** `geom` function (next section) that plots maps well.

The tidy tools manifesto prescribes four principles, which we will comment on:

1. **Reuse existing data structures.** We use the simplest R structures (numeric vector for point, matrix for point set, list for any other set), and fully support two standardized serializations (WKT, WKB)
2. **Compose simple functions with the pipe.** functions and methods were designed such that they can be used easily in pipe-based workflows; replacement functions like `st_crs<-` were augmented by `st_set_crs` to make this look better.
3. **Embrace functional programming.** Functions were kept type-safe, empty geometries and empty lists are supported, and operation overloading was done creatively e.g. by providing `Ops` for scaling and shifting a polygon:

   ```
   > pol * 2 + pt
   POLYGON ((0 1, 2 1, 2 3, 0 3, 0 1))
   ```

   Functions like `st_join` for a spatial join allow the user to pass a join function that is *compatible* with `st_intersects`, making the spatial predicate applied for the join completely customisable.
4. **Design for humans.** with the experience of having (co-)written and maintained **sp** for a decade, we have tried to keep **sf** simple and lean. Methods were used as much as possible to keep the namespace small. All functions and methods start with `st_` (for "spacetime", following PostGIS convention) to keep them recognizable, and searchable using tab-completion.

Plotting

Figure 1 (left) shows the default plot for an `"sf"` object with more than one attribute: no color keys are given, default colours depend on whether the variable is numeric (top) or a factor (bottom). Figure 1 was obtained by:

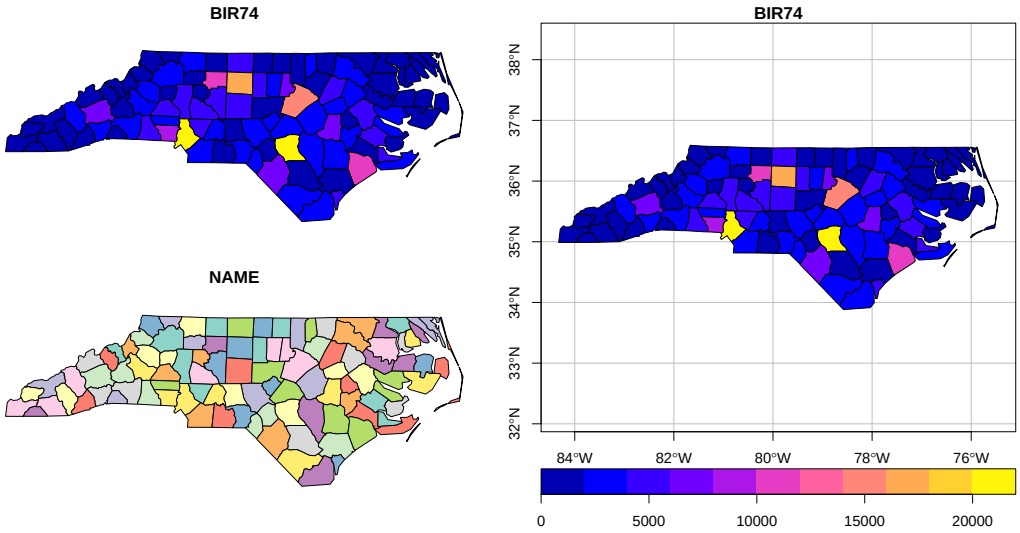


Figure 1: At left: default plot for `sf` object with two attributes; on right: plot for a single attribute with color key, axes and graticule.

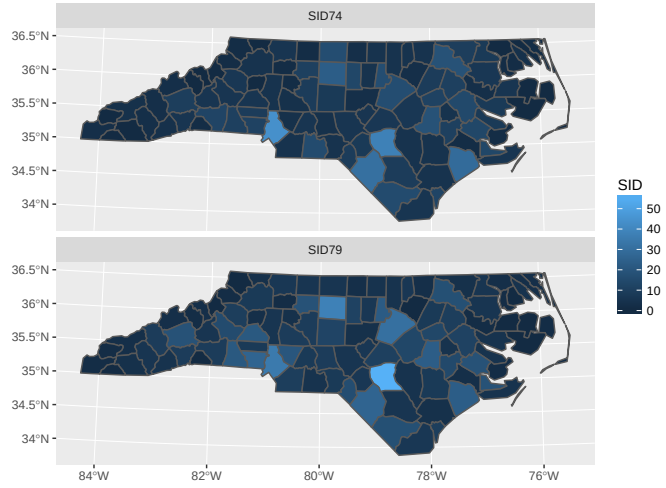


Figure 2: Plot generated with `ggplot2::geom_sf`, the now curved graticules follow constant long/lat lines.

```
> library(sf)
> nc = read_sf(system.file("gpkg/nc.gpkg", package="sf"))
> plot(nc[, c(9,5)])
```

When we plot a single attribute, a color key is default (unless `key.pos=NULL`). The following command

```
> plot(nc[, 9], key.pos = 1, axes = TRUE, graticule = TRUE)
```

adds axes and a graticule (longitude/latitude grid lines) on the right side of Figure 1.

Figure 2 shows a plot generated by `ggplot2` (version 2.2.1 or later):

```
> library(ggplot2)
> library(tidyr)
> library(dplyr)
> nc2 <- nc %>% st_transform(32119) %>% select(SID74, SID79, geom) %>%
+   gather(VAR, SID, -geom)
> ggplot() + geom_sf(data = nc2, aes(fill = SID)) + facet_wrap( ~ VAR, ncol = 1)
```

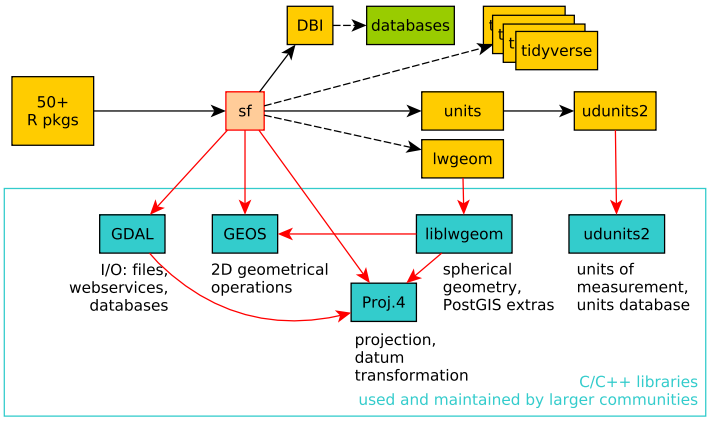


Figure 3: Dependencies of **sf** on other R packages and external system libraries.

Rasters, time series, and units

For some users, starting with **sf** feels like closing an old book (**sp**), and opening a new one. But it is not as if this new book has a similar content, or size. It is unsure when, or even whether at all, the hundreds of packages that use **sp** classes will be modified to use the `sf` classes.

The most heard question is where raster data are in this new book: **sp** provides simple classes for gridded data, **raster** (Hijmans, 2016b) provides heavy duty classes and a massive number of methods to work with them, tightly integrated with the **sp** vector classes. The current version of **raster** accepts **sf** objects in some of its functions by converting them to (the smaller set of) **sp** objects. At the time of writing this, we can only say that this is an area of active discussion, exploration and development, and we will be happy to point interested readers to where the public components of this discussion are taking place.

Besides raster data, time series for spatial features (e.g. for monitoring stations) are hard to map onto `sf` objects: one would either have to put time slices in columns, or add a time column and repeat the feature geometry for each observation. Raster data, spatial time series, and raster time series are the focus of the stars project.

A new aspect of the package is the ability to retrieve spatial measures and to set e.g. distance parameters with explicit measurement units (Pebesma et al., 2016):

```
> st_area(st_transform(nc[1, ], 2264)) # NC state plane, US foot

12244955726 US_survey_foot^2

> st_crs(2264)$units

[1] "us-ft"

> st_area(st_transform(nc[1, ], 2264)) %>% units::set_units(km^2) # convert:

1137.598 km^2
```

which might first confuse, but has the potential to prevent a whole category of scientific errors.

Connections to other computer systems and scalability

In many cases, analysing spatial data with R starts with importing data, or ends with exporting data, from or to a file or database. The ability to do this is primarily given by the well-known text (WKT) and well-known binary (WKB) serialisations that are part of the simple feature standard, and that are supported by **sf**. Communication with the GDAL, GEOS, and liblwgeom libraries uses WKB both ways. GDAL currently has drivers for 93 different spatial vector data connections (file formats, data bases, web services). Figure 3 shows the dependencies of **sf** on other R packages and system libraries. A reason to build upon these libraries is that they are used and maintained by, and hence reflect concensus of, the large community of spatial data experts outside R.

Besides using GDAL, **sf** can directly read and write from and to spatial databases. This currently works with PostGIS using **RPostgreSQL**; making this work with **RPostgres** and in general with spatial databases using **DBI** is under active development. Initial experiments indicate that working with massive, out-of-memory spatial databases in R is possible using the **dbplyr** framework. This not only removes the memory limits of R, but also benefits from the persistent spatial indexes of these databases.

For planar data, **sf** builds its spatial indexes on the fly for spatial binary predicates (`st_intersects`, `st_contains` etc.) and its binary operations (`st_intersection`, `st_difference` etc). A blog post about the spatial indexes in **sf** describes how using indexes makes these operations feasible for larger in-memory datasets. For spherical data, indexes e.g. provided by liblwgeom or by s2 still need to be explored.

Summary and further reading

We present a new package, **sf**, for simple features in R, as a modern alternative for parts of the **sp**-family of packages. It provides new foundational classes to handle spatial vector data in R, and has been received with considerable enthusiasm and uptake. While implementing **sf**, several well-proven concepts have been maintained (separation of geometries and attributes, libraries used), new links have been made (**dplyr**, **ggplot2**, spatial databases), and new concepts have been explored and implemented (units, spatial indexes).

For further reading into the full capabilities of **sf** and its rationale, the reader is refered to the six vignettes that come with the package.

Acknowledgments

Writing **sf** would not have been possible without all the prior work and continuous help of Roger Bivand. Package contributers are Ian Cook, Tim Keitt, Michael Sumner, Robin Lovelace, Hadley Wickham, Jeroen Ooms, and Etienne Racine. All contributors to GitHub issues are also acknowledged. Special thanks go to Dirk Eddelbuettel for developing **Rcpp** (Eddelbuettel et al., 2011; Eddelbuettel, 2013).

Support from the R Consortium has been very important for the development, visibility and fast adoption of **sf**, and is gratefully acknowledged. Anonymous reviewers are acknowledged for helpful comments.

Bibliography

R. Bivand and C. Rundel. *rgeos: Interface to Geometry Engine - Open Source ('GEOS')*, 2017. URL https://CRAN.R-project.org/package=rgeos. R package version 0.3-25. [p439]

R. Bivand, T. Keitt, and B. Rowlingson. *rgdal: Bindings for the 'Geospatial' Data Abstraction Library*, 2017. URL https://CRAN.R-project.org/package=rgdal. R package version 1.2-15. [p439]

R. S. Bivand, E. Pebesma, and V. Gomez-Rubio. *Applied Spatial Data Analysis with R, Second Edition*. Springer-Verlag, 2013. URL http://www.asdar-book.org/. [p439]

H. Butler, M. Daly, A. Doyl, S. Gillies, S. Hagen, and T. Schaub. The GeoJSON format, 2016. ISSN 2070-1721. URL https://tools.ietf.org/html/rfc7946. [p439]

V. H. Do, C. Thomas-Agnan, and A. Vanhems. Accuracy of areal interpolation methods for count data. *Spatial Statistics*, 14:412 – 438, 2015. URL https://doi.org/10.1016/j.spasta.2015.07.005. [p441]

D. Eddelbuettel. *Seamless R and C++ Integration with Rcpp*. Springer-Verlag, New York, 2013. [p445]

D. Eddelbuettel, R. François, J. Allaire, K. Ushey, Q. Kou, N. Russel, J. Chambers, and D. Bates. Rcpp: Seamless R and C++ integration. *Journal of Statistical Software*, 40(8):1–18, 2011. [p445]

A. Fabri and S. Pion. CGAL: The computational geometry algorithms library. In *Proceedings of the 17th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems*, pages 538–539. ACM, 2009. [p442]

GEOS Development Team. *GEOS - Geometry Engine, Open Source*. Open Source Geospatial Foundation, 2017. URL https://trac.osgeo.org/geos/. [p439]

J. R. Herring. OpenGIS implementation standard for geographic information-simple feature access-part 2: SQL option. *Open Geospatial Consortium Inc*, 2010. URL http://portal.opengeospatial.org/files/?artifact_id=25354. [p439]

J. R. Herring. OpenGIS implementation standard for geographic information-simple feature access-part 1: Common architecture. *Open Geospatial Consortium Inc*, page 111, 2011. URL http://portal.opengeospatial.org/files/?artifact_id=25355. [p439]

R. J. Hijmans. *geosphere: Spherical Trigonometry*, 2016a. URL https://CRAN.R-project.org/package=geosphere. R package version 1.5-5. [p442]

R. J. Hijmans. *Raster: Geographic Data Analysis and Modeling*, 2016b. URL https://CRAN.R-project.org/package=raster. R package version 2.5-8. [p444]

ISO. *Geographic Information – Simple Feature Access – Part 1: Common Architecture*, 2004. URL https://www.iso.org/standard/40114.html. ISO 19125-1:2004. [p439]

R. O. Obe and L. S. Hsu. *PostGIS in Action*. Manning Publications Co., 2015. [p439]

E. Pebesma. *lwgeom: Bindings to Selected 'liblwgeom' Functions for Simple Features*. URL https://CRAN.R-project.org/package=lwgeom. R package version 0.1-5. [p442]

E. Pebesma. *sf: Simple Features for R*, 2018. URL https://CRAN.R-project.org/package=sf. R package version 0.6-1. [p439]

E. Pebesma, T. Mailund, and J. Hiebert. Measurement units in R. *The R Journal*, 8(2):486–494, 2016. URL https://journal.r-project.org/archive/2016-2/pebesma-mailund-hiebert.pdf. [p444]

E. J. Pebesma and R. S. Bivand. Classes and methods for spatial data in R. *R News*, 5(2):9–13, 2005. URL https://CRAN.R-project.org/doc/Rnews/. [p439]

M. Perry and J. Herring. OGC GeoSPARQL-a geographic query language for RDF data. *OGC Implementation Standard, ref: OGC*, 2012. [p439]

E. Rubak and J. Ooms. *S2: Google's S2 Library for Geometry on the Sphere*, 2017. URL https://CRAN.R-project.org/package=s2. R package version 0.1-1. [p442]

S. Scheider, B. Gräler, E. Pebesma, and C. Stasch. Modeling spatiotemporal information generation. *International Journal of Geographical Information Science*, 30(10):1980–2008, 2016. URL https://doi.org/10.1080/13658816.2016.1151520. [p439]

F. Warmerdam. The geospatial data abstraction library. In *Open Source Approaches in Spatial Data Handling*, pages 87–104. Springer-Verlag, 2008. [p439]

H. Wickham. Tidy data. *Journal of Statistical Software, Articles*, 59(10):1–23, 2014. ISSN 1548-7660. URL https://doi.org/10.18637/jss.v059.i10. [p439]

H. Wickham. *Ggplot2: Elegant Graphics for Data Analysis*. Springer-Verlag, 2016. [p439, 538]

H. Wickham. *Tidyverse: Easily Install and Load the 'Tidyverse'*, 2017. URL https://CRAN.R-project.org/package=tidyverse. R package version 1.1.1. [p439]

H. Wickham, R. Francois, L. Henry, and K. Müller. *dplyr: A Grammar of Data Manipulation*, 2017. URL https://CRAN.R-project.org/package=dplyr. R package version 0.7.4. [p439]

Edzer Pebesma
Institute for Geoinformatics
Heißenbergstraße 2
Münster, Germany
ORCiD: 0000-0001-8049-7069
edzer.pebesma@uni-muenster.de

Pstat: An R Package to Assess Population Differentiation in Phenotypic Traits

by Stéphane Blondeau Da Silva and Anne Da Silva

Abstract The package **Pstat** calculates P_{ST} values to assess differentiation among populations from a set of quantitative traits and provides bootstrapped distributions and confidence intervals for P_{ST}. Variations of P_{ST} as a function of the parameter c/h^2 are studied as well. The package implements different transformations of the measured phenotypic traits to eliminate variation resulting from allometric growth, including calculation of residuals from linear regression, Reist standardization, and the Aitchison transformation.

Introduction

Understanding the causes governing patterns of morphological variations in the wild represents a fundamental goal of evolutionary biology. In particular, the relative importance of selective and neutral processes behind the observed differentiation remains a crucial question.

Studies comparing differentiation in quantitative traits and neutral markers have significantly increased over the last ten years (Leinonen et al., 2013). Typically, a set of populations is sampled and the degree of genetic differentiation is estimated for a set of molecular markers with the Wright's F_{ST} index (Wright, 1951). For its part, the Q_{ST} index (Spitze, 1993) assesses the degree of phenotypic differentiation over a set of quantitative traits. The logic of F_{ST} and Q_{ST} comparison relies on the assumption that the F_{ST} obtained by the consideration of neutral markers reflects the divergence only induced by genetic drift (Reynolds et al., 1983). Hence, F_{ST} provides a null expectation and allows estimation of the degree of population differentiation that would be reached without selection (Merilä and Crnokrak, 2001).

As a consequence, the comparison between F_{ST} and Q_{ST} leads to three possibilities: (i) $Q_{ST} > F_{ST}$ means that quantitative traits show a higher level of differentiation than what would have been expected under the influence of genetic drift, such that natural selection could induce differentiation between populations by favoring different phenotypes (i.e., heterogeneous selection); (ii) $Q_{ST} < F_{ST}$ could indicate the influence of natural selection, but selecting for same optima among populations (i.e., homogeneous selection); (iii) $Q_{ST} = F_{ST}$ means that no departure from neutral expectations can be detected and that the degree of differentiation in quantitative traits could have been obtained only by genetic drift, even if the contribution of natural selection can neither be excluded nor estimated.

Spitze (1993) introduced and defined the Q_{ST} quantity as follows for diploid species assuming purely additive gene action:

$$Q_{ST} = \frac{\sigma^2_{A_b}}{\sigma^2_{A_b} + 2\sigma^2_{A_w}},$$

where $\sigma^2_{A_b}$ and $\sigma^2_{A_w}$ are the morphological additive genetic variance components between and within populations.

In the wild, the estimation of the additive genetic variance components is challenging as breeding design is impossible. Therefore, Q_{ST} is often approximated by P_{ST} (Leinonen et al., 2006), which is directly calculated from the total phenotypic variance components with no distinction between the relative contribution of genetic and environmental variations:

$$P_{ST} = \frac{c\sigma^2_b}{c\sigma^2_b + 2h^2\sigma^2_w} \quad \text{or} \quad P_{ST} = \frac{\frac{c}{h^2}\sigma^2_b}{\frac{c}{h^2}\sigma^2_b + 2\sigma^2_w}, \tag{1}$$

where σ^2_b and σ^2_w are the respective phenotypic variances between and within populations, c is an estimate of the proportion of the total variance due to additive genetic effects across populations, and h^2 is heritability, the proportion of phenotypic variance due to additive genetic effects (Brommer, 2011). How well P_{ST} approximates Q_{ST} depends on the parameters c and h^2, such that if the values of c and h^2 are known, then the phenotypic divergence quantified by P_{ST} would equal Q_{ST}. This implies caution in the interpretation obtained from P_{ST} (Brommer, 2011).

A large number of studies have assessed the potential for natural selection to affect morphological evolution by comparing phenotypic divergence with neutral genetic divergence via a P_{ST} versus F_{ST} approach (e.g., Mobley et al., 2011; Lima, 2012; He et al., 2013; Shinn et al., 2015). While estimation of F_{ST} values are included in various R packages such as **diveRsity** (Keenan et al., 2013) or **hierfstat**

(Goudet, 2005), no R package exists to deal with the P_{ST} index. In this study, we present the **Pstat** package to handle large datasets of quantitative traits and correct quantitative traits taking into account allometric growth. The package calculates P_{ST} values with their respective bootstrapped confidence intervals, and offers several options to select individuals, traits, or populations. We also provide various plotting tools for the visual evaluation of P_{ST} and F_{ST} values. We will walk through a detailed example to give an overview of the **Pstat** package.

An example to get familiar with the main functions

After loading the package with `library(Pstat)`, load the sample data with `data(test)`. This data frame contains 200 rows, with each row representing an individual in a population of common wetland plants, *Juncus effusus* (see Michalski and Durka 2015 and the Dryad Digital Repository, https://doi.org/10.5061/dryad.bk5hk). The data frame contains the name of the populations (`A`, `B`, `C`, `D`, and `E`) to which each individual belongs and eleven quantitative measures. An excerpt from the sample data are presented in Table 1.

	Populations	QM1	QM2	QM3	QM4	...
1	A	0.18487253	0.4001979	0.1694021	42	...
2	B	0.24023500	0.4718000	0.2178500	46	...
3	C	0.23499676	0.4686213	0.2060222	25	...
4	B	0.20495223	0.3746026	0.1846816	51	...
5	C	0.20739220	0.4866461	0.2131618	19	...
6	C	0.22545341	0.3770903	0.1882165	28	...
7	C	0.18371681	0.4992361	0.2167194	25	...
...	...	...	...	...	...	...

Table 1: Sample from the `test` data frame, containing quantitative measures for individual members of *Juncus effusus* (Michalski and Durka, 2015).

The data preparation

The package can be used to transform data to eliminate variation resulting from allometric growth. Users have the choice between three alternatives:

1. Residuals of a linear regression, with one of the quantitative variables used as the regressor (Kuhry and Marcus, 1977);
2. The allometric transformation described in Reist (1985); or
3. Aitchison's log-ratio transform (Aitchison, 1986).

Among a variety of univariate transformations that aim to separate size and shape variations, Reist (1985) showed that adjustments for size using a regression and residuals (the first option) and allometric adjustments to a standard size (the second option) are preferred since they allow the complete removal of size variations and have minimal impact on the correlation and covariance structure of the data. Unlike the first two options, the third transformation offers the benefit of keeping the same number of variables. We provide examples of each of the three alternatives below.

Simple linear adjustments

The first adjustment method provided by **Pstat** is a simple linear regression. Assuming the existence of linear relationships between the dependent variable and one of the quantitative traits, the `Res` function returns a new data frame with the residuals of the regression. The function's arguments are as follows:

- **data**: the studied data frame to be transformed with as many rows as individuals; the first column must contain the population to which the individual belongs and the other columns may contain quantitative variables.
- **reg**: the name or the rank of the variable chosen as the regressor.
- **Rp**: the names of the populations to be deleted. Default value: `Rp=0`, no population removed.
- **Ri**: the line numbers of individuals to be deleted. Default value: `Ri=0`, no individuals removed.

We present sample output from the `test` data, using one of the quantitative traits as the regressor. A sample of the transformed data output by `Res` is presented in Table 2.

```
## Using the explanatory variable QM3 as the regressor
Res(data=test, reg="QM3")
```

	Populations	QM1	QM2	QM4	...
1	A	0.0339245264	5.621424e-03	6.23817063	...
2	B	0.1001268662	4.497085e-02	8.44522196	...
3	C	0.0922422473	4.966613e-02	-12.11705813	...
4	B	0.0574228940	-3.014565e-02	14.67271191	...
5	C	0.0662351079	6.293798e-02	-18.38127681	...
6	C	0.0787149904	-3.001126e-02	-8.45810788	...
7	C	0.0433557311	7.315960e-02	-12.51293868	...
...	...	...	...	...	...

Table 2: Sample from the adjusted data frame output by `Res`, using `QM3` as the explanatory variable.

Reist transformation

In the second adjustment method provided by **Pstat**, all morphometric measurements are standardized using the transformation proposed by Reist (1985).

Let n be the number of individuals and p the number of quantitative traits such that $\exists\ k \in \{1,\dots,p\}$ and the k^{th} trait is the explanatory variable. Let us denote this variable $(x_i)_{1\leq i\leq n}$ and the other traits as $j \in \{1,\dots,p\}\setminus\{k\}$, $(y_{ij})_{1\leq i\leq n}$. The Reist transformation is

$$\begin{aligned} &\forall i \in \{1,\dots,n\} \quad \text{and} \quad \forall j \in \{1,\dots,p\}\setminus\{k\}\,, \\ &Y_{ij} = log(y_{ij}) - b_j(log(x_i) - log(\overline{x}))\,, \end{aligned}$$

where Y_{ij} is the size adjusted measurement of the j^{th} trait for the i^{th} individual, y_{ij} the original morphometric measurement, $\overline{x}$ the population mean of the explanatory variable, and x_i the value of the explanatory variable for the i^{th} individual. For all $j \in \{1,\dots,p\}\setminus\{k\}$, the parameter b_j is estimated for the quantitative trait y_j (i.e. $(y_{ij})_{1\leq i\leq n}$) and represents the slope of the linear regression of $log(y_j)$ on $log(x)$.

The `ReistTrans` function returns a corrected data frame. Using `QM3` as the explanatory variable, we present a sample of the transformed data frame in Table 3.

```
## Using QM3 as the explanatory variable (identified by column number)
ReistTrans(test, reg=3)
```

	Populations	QM1	QM2	QM4	...
1	A	-0.7445410	-0.3859875	1.631722	...
2	B	-0.6004703	-0.3462059	1.648348	...
3	C	-0.6167708	-0.3421063	1.388608	...
4	B	-0.6893556	-0.4255755	1.708186	...
5	C	-0.6669355	-0.3300087	1.266323	...
6	C	-0.6456670	-0.4250906	1.446049	...
7	C	-0.7175846	-0.3210021	1.384003	...
...	...	...	...	...	...

Table 3: Sample from the Reist adjusted data frame using `QM3` as the explanatory variable.

Aitchison transformation

The third adjustment method provided by **Pstat** performs the Aitchison log-ratio transformation to account for individual size-effects (Aitchison, 1986).

Let n be the number of individuals and p the number of morphological traits. For $j \in \{1,\dots,p\}$, let $(y_{ij})_{1\leq i\leq n}$ represent the quantitative variables. The formula formula for the Aitchison transformation

is as follows:

$$\forall i \in \{1,\ldots,n\} \quad \text{and} \quad \forall j \in \{1,\ldots,p\}\,,$$
$$Y_{ij} = log(y_{ij}) - \frac{1}{p}\sum_{k=1}^{p} log(y_{ik})\,,$$

where Y_{ij} is the transformed measure of the j^{th} trait for the i^{th} individual, and y_{ij} is the original value for the i^{th} individual and the j^{th} trait.

The `AitTrans` function returns a corrected data frame. Sample output are included in Table 4.

```
AitTrans(test)
```

	Populations	QM1	QM2	QM3	QM4	...
1	A	-1.947544	-1.6121417	-1.985498	0.408832854	...
2	B	-1.910214	-1.6170919	-1.952692	0.371908012	...
3	C	-1.834151	-1.5343910	-1.891299	0.192727037	...
4	B	-1.901481	-1.6395625	-1.946710	0.494436831	...
5	C	-1.832709	-1.4622885	-1.820792	0.129251912	...
6	C	-1.801889	-1.5785008	-1.880288	0.292211929	...
7	C	-1.938699	-1.5045418	-1.866950	0.195092172	...
...	...	...	...	...	...	...

Table 4: Sample from the Aitchison adjusted data frame.

Phenotypic differentiation evaluation and confidence intervals

P_{ST} values

We are interested in determining the phenotypic differentiation across the five populations for each of the eleven quantitative traits of the example dataset. The function `Pst` can determine the P_{ST} values of each trait with the associated bootstrapped confidence intervals (Efron and Tibshirani, 1993). The arguments to `Pst` are as follows:

- **data**: the input data frame with as many rows as individuals; the first column must contain the population label and the others quantitative variables.
- **ci**: if `ci=1`, the confidence intervals are added to P_{ST} values. Default value: `ci=0`.
- **csh**: the $\frac{c}{h^2}$ value. Default value: `csh=1`.
- **va**: a vector containing the names or column numbers of the quantitative measures under consideration. If `va=0`, all the variables are selected. Default value: `va=0`.
- **boot**: the number of data frames generated to determine the confidence interval with the bootstrap method. Default value: `boot=1000`.
- **Pw**: the names of the two populations considered to obtain pairwise P_{ST}. Default value: `Pw=0`, no pairwise analysis.
- **Rp**: the names of the populations to be deleted. Default value: `Rp=0`, no populations removed.
- **Ri**: the line numbers of individuals to be deleted. Default value: `Ri=0`, no individuals removed.
- **pe**: the confidence level of the calculated interval. Default value: `pe=0.95`.

Let us apply the `Pst` function to the `test` dataset. The output from `Pst` will be a data frame:

```
## Example 1:  Pairwise Pst values using populations C and D
Pst(test, csh=0.2, Pw=c("C","D"))
[1] "Populations sizes are:"
 C  D
76 32
   Quant_Varia Pst_Values
 1         QM1  0.1749659
 2         QM2  0.7460913
...        ...        ...
```

```
 4        QM10  0.9800028

## Example 2: Pst for the 2nd variable and QM7 with 99% confidence intervals
Pst(test, va=c(2,"QM7"), ci=1, boot=10000, Ri=c(5,117:121), pe=0.99)
[1] "Populations sizes are:"
 A  B  C  D  E
12 76 72 30  4
  Quant_Varia Pst_Values 99 %_LowBoundCI 99 %_UpBoundCI
1         QM2  0.8561307       0.7826177      0.9198395
2         QM7  0.8851413       0.7722856      0.9376501
```

Distribution of P_{ST}

The bootstrapped P_{ST} values output from `BootPst` form a distribution for the selected quantitative trait. In addition to arguments that are shared with `Pst`, the `BootPst` function has the following additional arguments specific to the bootstrap procedure:

- **opt**: if `opt=0`, all the boot values of P_{ST} are returned; if `opt="ci"`, the ordered values and the confidence interval are returned; and if `opt="hist"`, the ordered values and the distribution histogram of P_{ST} are returned. Default value: `opt=0`.
- **va**: the name or column number of the quantitative measure considered.
- **bars**: the maximum number of bars the histogram may have. On the x-axis, the interval $[0, 1]$ is divided into `bars` parts (there may exist unfilled bars). Default value: `bars=20`.

The output from the `BootPst` function is a vector with the bootstrapped values.

Let us apply the `BootPst` function to `test` dataset:

```
## Example 1: Bootstrapped 95% confidence intervals for three populations (B, C, and D).
## Note that populations A and E are dropped
BootPst(test, opt="ci", va="Body_length", Rp=c("A","E"))
[1] "The studied quantitative variable is:"
[1] "Body_length"
[1] "Populations sizes are:"
 B  C  D
76 76 32
[1] "95 % confidence interval determined by 1000 bootstrap values:"
[1] 0.8757057 0.9585423
   [1] 0.7938426 0.8338286 0.8510682 0.8512374 0.8545911 0.8551115 0.8552097
   [8] 0.8637057 0.8641575 0.8644145 0.8659723 0.8671139 0.8671265 0.8676122
   [15] 0.8686147 0.8702277 0.8708352 0.8711419 0.8718030 0.8721783 0.8734932
   ...
   [995] 0.9621794 0.9625852 0.9634700 0.9644283 0.9650500 0.9689611

## Example 2: Histogram for the trait in column 3 (output in Figure 1)
BootPst(test, opt="hist", va=3, bars=50)
[1] "The studied quantitative variable is:"
[1] "QM3"
[1] "Populations sizes are:"
 A  B  C  D  E
12 76 76 32  4
[1] "1000 bootstrap values and Pst distribution:"
   [1] 0.1062747 0.1076470 0.1269888 0.1593121 0.1775196 0.2050347 0.2111617
   [8] 0.2327508 0.2401064 0.2487401 0.2588179 0.2589942 0.2623706 0.2722956
   [15] 0.2827915 0.2860497 0.2935858 0.2947525 0.2954878 0.2995198 0.3003267
   ...
   [995] 0.8211326 0.8253874 0.8293417 0.8318546 0.8420100 0.8635299
```

Variations of P_{ST} values and visual comparison with Wright's F_{ST} index

Brommer (2011) and Lima (2012) offer plots that demonstrate how F_{ST} and P_{ST} depend on the $\frac{c}{h^2}$ ratio. The **Pstat** package provides plotting tools to perform these analyses with the function `TracePst`. Arguments specific to `TracePst` include:

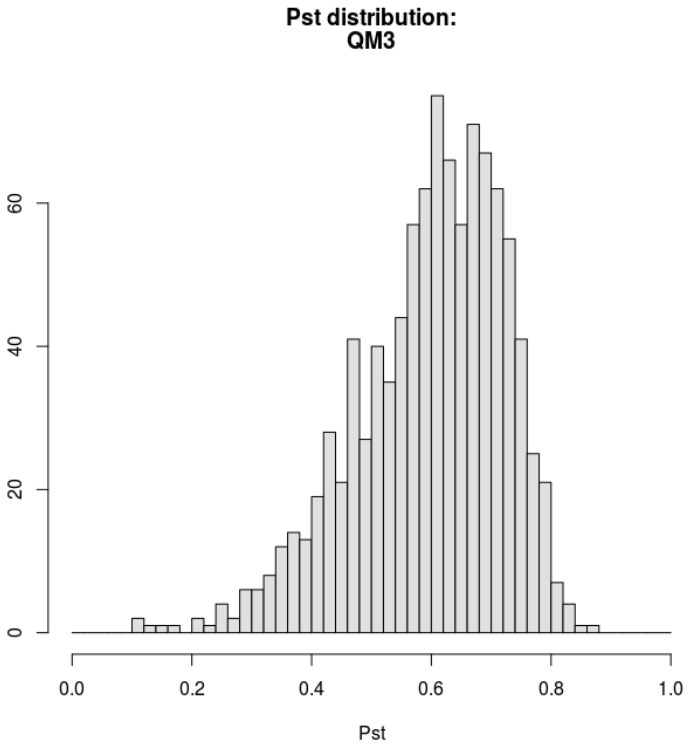


Figure 1: P_{ST} distribution histogram of QM3.

- **va**: a vector containing the selected variables names or numbers (*i.e.* those of the quantitative measures considered). If `va=0`, all the variables are selected. Default value: `va=0`.
- **ci**: if `ci=1`, the confidence interval of P_{ST} is plotted. Default value: `ci=1`.
- **Fst**: the value of Wright's F_{ST}, if available. Default value: `Fst=-1`, value of F_{ST} is unavailable.
- **xm**: x-axis maximum. Default value: `xm=2`.
- **pts**: the number of points used to plot the curves. Default value: `pts=30`.

Let us apply the `TracePst` function to the `test` dataset. The plots output are in Figure 2.

```
# Aitchison adjustment method:
trans_test=AitTrans(test)

# Plots illustrating how comparisons between Fst and Pst depends on c/h^2:
TracePst(trans_test, Fst=0.3, xm=3)
[1] "Populations sizes are:"
 A  B  C  D  E
12 76 76 32  4
```

Conclusion

The use of P_{ST} versus F_{ST} comparison has increased rapidly in the last few years in the field of evolutionary and ecological genetics. The **Pstat** package is the counterpart of existing R packages dealing with *F*-statistics. It calculates P_{ST} values, and also provides bootstrapped confidence intervals, several graphical tools, as well as three ways of transforming data to remove variation resulting from allometric growth.

Bibliography

J. Aitchison. *The Statistical Analysis of Compositional Data*. Chapman and Hall, London, New York, 1986. 416 pp. [p448, 449]

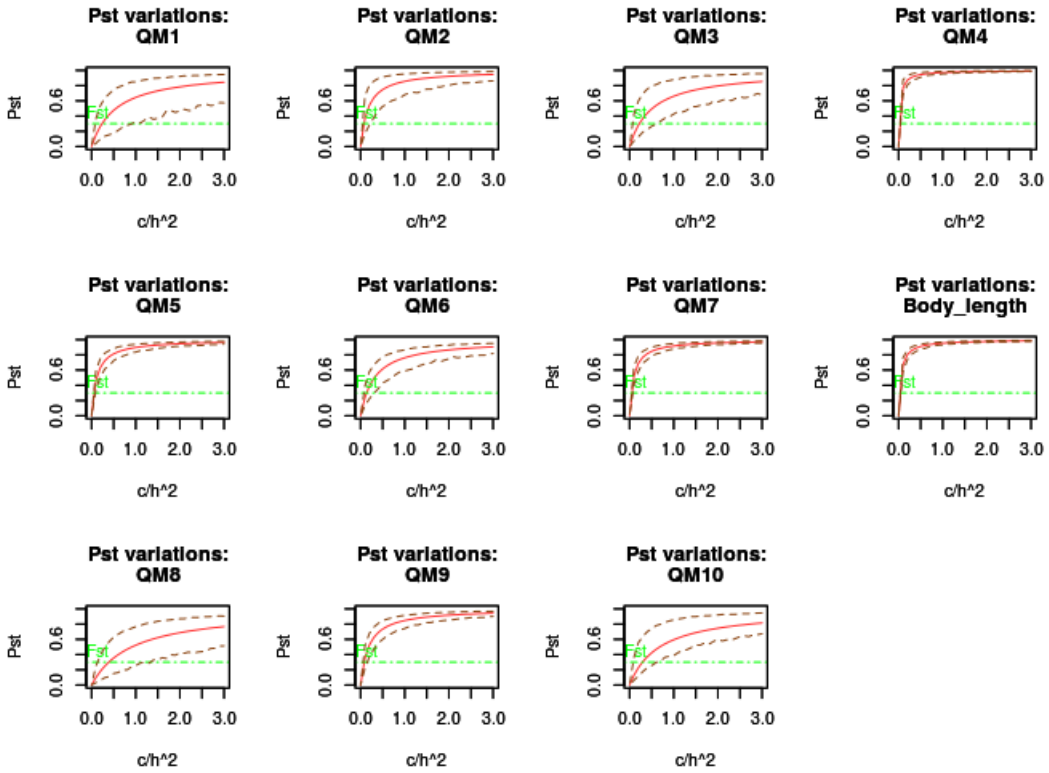


Figure 2: Plots illustrating comparisons between F_{ST} and P_{ST}. The horizontal dotted green line marks the value of F_{ST}. P_{ST} values and associated 95% confidence intervals are plotted in red.

J. E. Brommer. Whither Pst? The approximation of Qst by Pst in evolutionary and conservation biology. *Journal of Evolutionary Biology*, 24:1160–1168, 2011. URL https://doi.org/10.1111/j.1420-9101.2011.02268.x. [p447, 451]

B. Efron and R. Tibshirani. *An Introduction to the Bootstrap*. Chapman and Hall, New York, 1993. [p450]

J. Goudet. HIERSTAT, a package for R to compute and test hierarchical F-statistics. *Molecular Ecology Resources*, 5(1):184–186, 2005. URL https://doi.org/10.1111/j.1471-8286.2004.00828.x. [p448]

Y. He, R. Li, J. Wang, S. Blanchet, and S. Lek. Morphological variation among wild populations of chinese rare minnow (Gobiocypris rarus): Deciphering the role of evolutionary processes. *Zoological Science*, 30:475–483, 2013. URL https://doi.org/10.2108/zsj.30.475. [p447]

K. Keenan, P. McGinnity, T. F. Cross, W. W. Crozier, and P. A. Prodöhl. diveRsity: An R package for the estimation and exploration of population genetics parameters and their associated errors. *Methods in Ecology and Evolution*, 4(8):782–788, 2013. URL https://doi.org/10.1111/2041-210X.12067. [p447]

B. Kuhry and L. F. Marcus. Bivariate linear models in biometry. *Systematic Zoology*, 26(2):201–209, 1977. URL https://doi.org/10.2307/2412842. [p448]

T. Leinonen, J. M. Cano, H. Makinen, and J. Merilä. Contrasting patterns of body shape and neutral genetic divergence in marine and lake populations of threespine sticklebacks. *Journal of Evolutionary Biology*, 19:1803–1812, 2006. URL https://doi.org/10.1111/j.1420-9101.2006.01182.x. [p447]

T. Leinonen, R. J. McCairns, R. B. O'Hara, and J. Merilä. Qst–Fst comparisons: Evolutionary and ecological insights from genomic heterogeneity. *Nature Reviews Genetics*, 14:179–190, 2013. URL https://doi.org/10.1038/nrg3395. [p447]

M. R. Lima. Genetic and morphometric divergence of an invasive bird: The introduced house sparrow (Passer domesticus) in Brazil. *PloS One*, 7(12), 2012. URL https://doi.org/10.1371/journal.pone.0053332. [p447, 451]

J. Merilä and P. Crnokrak. Comparison of genetic differentiation at marker loci and quantitative traits. *Journal of Evolutionary Biology*, 14:892–903, 2001. URL https://doi.org/10.1046/j.1420-9101.2001.00348.x. [p447]

S. G. Michalski and W. Durka. Separation in flowering time contributes to the maintenance of sympatric cryptic plant lineages. *Ecology and Evolution*, 5(11):2172–2184, 2015. URL https://doi.org/10.1002/ece3.1481. [p448]

K. B. Mobley, D. Lussetti, F. Johansson, G. Englund, and F. Bokma. Morphological and genetic divergence in swedish postglacial stickleback (Pungitius pungitius) populations. *BMC Evolutionary Biology*, 2011. URL https://doi.org/10.1186/1471-2148-11-287. [p447]

J. D. Reist. An empirical evaluation of several univariate methods that adjust for size variation in morphometric data. *Canadian Journal Zoology*, 63:1429–1439, 1985. URL https://doi.org/10.1139/z85-213. [p448, 449]

J. Reynolds, B. S. Weir, and C. C. Cockerham. Estimation of the coancestry coefficient: Basis for a short-term genetic distance. *Genetics*, 105:767–779, 1983. [p447]

C. Shinn, S. Blanchet, G. Loot, S. Lek, and G. Grenouillet. Phenotypic variation as an indicator of pesticide stress in gudgeon: Accounting for confounding factors in the wild. *Science of the Total Environment*, 538:733–752, 2015. URL https://doi.org/10.1016/j.scitotenv.2015.08.081. [p447]

K. Spitze. Population structure in Daphnia obtusa: Quantitative genetic and allozymic variation. *Genetics*, 135:367–374, 1993. [p447]

S. Wright. The genetical structure of populations. *Annals of Human Genetics*, 15:323–354, 1951. URL https://doi.org/10.1111/j.1469-1809.1949.tb02451.x. [p447]

Stéphane Blondeau Da Silva, corresponding author
XLIM-Mathis, UMR n°7252 CNRS-Université de Limoges
123, avenue Albert Thomas 87060 Limoges
France
blondeaudasilva@xlim.fr

Anne Da Silva
INRA, UMR n°1061 Génétique Moléculaire Animale
Université de Limoges, UMR n°1061 Génétique Moléculaire Animale
123, avenue Albert Thomas 87060 Limoges
France
anne.blondeau@unilim.fr

Collections in R: Review and Proposal

by Timothy Barry

Abstract R is a powerful tool for data processing, visualization, and modeling. However, R is slower than other languages used for similar purposes, such as Python. One reason for this is that R lacks base support for *collections*, abstract data types that store, manipulate, and return data (e.g., sets, maps, stacks). An exciting recent trend in the R extension ecosystem is the development of *collection packages*, packages that provide classes that implement common collections. At least 12 collection packages are available across the two major R extension repositories, the Comprehensive R Archive Network (CRAN) and Bioconductor. In this article, we compare collection packages in terms of their features, design philosophy, ease of use, and performance on benchmark tests. We demonstrate that, when used well, the data structures provided by collection packages are in many cases significantly faster than the data structures provided by base R. We also highlight current deficiencies among R collection packages and propose avenues of possible improvement. This article provides useful recommendations to R programmers seeking to speed up their programs and aims to inform the development of future collection-oriented software for R.

Introduction

R is one of the most popular languages used for data analysis due to its rich package ecosystem, robust user support community, and powerful modeling and graphical capabilities (Muenchen, 2012). However, R is not a fast language. R has an unusual and costly combination of language features, including lazy function evaluation, extreme dynamism, and pass-by-value to functions (Morandat et al., 2012). Additionally, R lacks base support for common computer science data structures (Wickham, 2014), impeding users from selecting efficient data structures for specific tasks (Cormen, 2009). Some of these limitations are necessary trade-offs of a programming environment that facilitates sophisticated modeling for practitioners across many fields. However, the general trend toward larger datasets makes the performance cost of these limitations more acute.

A promising avenue for addressing these difficulties is the development of R packages that implement *collections*. In computer science, a collection is an interface for storing, manipulating, and retrieving a group of like data items (Dale and Walker, 1996). Common examples of collections include sets, maps, and stacks. Collections are used frequently in programming; the languages Python, Java, and C++ ship with comprehensive collection frameworks (van Rossum and Drake, 2011; Stroustrup, 2013; Schildt, 2014). The addition of collections to R is an exciting prospect. Collections are (in general) backed by mathematically optimal algorithms and thus help accelerate program execution times. In the case of R, collections can be implemented in C++ via **Rcpp** for increased speed (Eddelbuettel et al., 2011). An added benefit of collections is that they make programs more abstract and thus easier to write, read, debug, and revise (Liskov and Zilles, 1974).

In recent years, researchers have developed R packages designed specifically to implement collections. For example, Peng (2006), Brown (2013), and Russell (2017) implemented maps; Meyer and Hornik (2009) implemented sets; O'Neil (2015), Csárdi (2016), Collier (2016), Schmidt (2016), and Poncet (2017) implemented stacks and queues; and Bengtsson (2015), Giuliano (2017), and Pagés et al. (2017) implemented sequences (also known as *lists*). These packages vary in their features, performance, memory footprint, design philosophy, and ease of use. Consequently, selecting the appropriate package for a particular task is challenging. Moreover, these packages have inconsistent application programming interfaces (APIs), and so switching between them or using several in unison is awkward or in some cases impossible.

The purpose of this paper is twofold: first, to describe and compare R packages that implement collections (henceforth called *collection packages*); second, to propose an avenue for improving the collection capabilities of R. The former will aid R users in navigating the existing collection package ecosystem, while the latter will help guide the development of new R collection software. We begin with a brief review of common computer science data structures and clarify the distinction between abstract data types and data structures. We then evaluate R collection packages and compare them in terms of features and performance. We conclude with a discussion of future directions and areas of potential improvement. In particular, we propose the development of a unified, simple, and efficient R collection framework.

Background

We briefly review data structures and abstract data types.

Data structures

A data structure is a way of organizing information in a computer's memory (Black, 1998). There exist myriad data structures, each of which has strengths and weaknesses. In this section we review what we consider to be the three data structures most relevant to R users: arrays, linked lists, and hash tables. We assume the reader is familiar with big O notation [1]. Readers interested in learning more should refer to, for example, Cormen (2009).

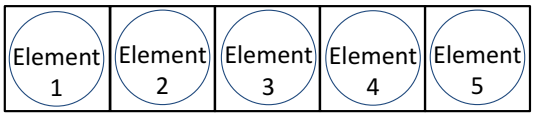


Figure 1: An example of an array. The circles represent data elements, and the squares represent locations of storage in memory.

An array is a linear sequence of elements stored contiguously in memory (Figure 1). Retrieving or modifying an element at a particular index is $\mathcal{O}(1)$. Removing an element from the middle or beginning of an array is $\mathcal{O}(n)$ because elements that follow the element that has been removed must be copied and shifted one position to the left. Checking whether an element is present in an array (called *search*) typically requires iterating over each element and in this case is $\mathcal{O}(n)$. Finally, assuming sufficient space has been allocated so as to not require array resizing, appending an element to the end of an array is $\mathcal{O}(1)$. (If resizing is necessary, appending an element is $\mathcal{O}(n)$.)

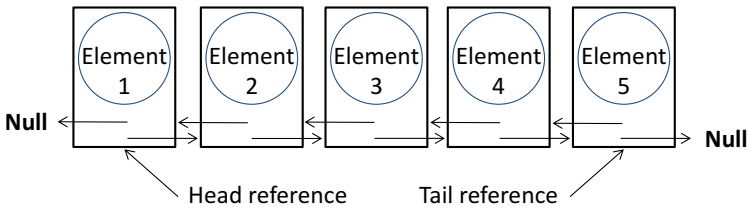


Figure 2: An example of a doubly-linked list. The circles represent data elements, and the rectangles represent nodes. The front and rear nodes have references to `NULL`. The list has both head and tail references.

A linked list is a linear sequence of objects called *nodes* (Figure 2). Each node contains a single data element and a reference to the following node (and, in the case of *doubly-linked lists*, a reference to the previous node as well). We focus here on doubly-linked lists because they are most relevant to the current paper. Insertion or deletion at a node for which a reference exists is $\mathcal{O}(1)$. Because the program must store a reference to the front (or *head*) node to access the list, insertion or removal at the front of a doubly-linked list is $\mathcal{O}(1)$. The program can optionally store a reference to the rear (or *tail*) node, in which case insertion or removal at the rear of the list is $\mathcal{O}(1)$.

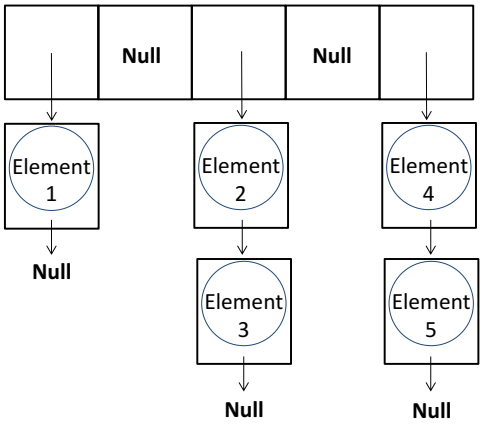


Figure 3: An example of a hash table. The circles represent data elements, the connected squares represent storage locations of an array, and the disconnected rectangles represent linked lists of nodes.

[1] Briefly, for functions $f, g : \mathbb{Z}^{\geq 0} \to \mathbb{R}$, we say f is $\mathcal{O}(g)$ if there exist $c, n_0 > 0$ such that $0 \leq f(n) \leq cg(n)$ for all $n \geq n_0$.

A hash table is a data structure built to support fast insertion, removal, and search (Figure 3). Hash tables can be implemented in a number different ways, but one common implementation is roughly as follows: Allocate an array. Define a function that associates data elements to be stored in the hash table with an index of the array. When adding an element, compute the index with which it is associated and store it in the appropriate array position. If multiple data elements are mapped to the same array position, store the elements in a linked list. Hash tables support amortized $\mathcal{O}(1)$ insertion, removal, and search.

Abstract data types

An abstract data type is a collection of objects whose behavior is defined by a set of operations (Dale and Walker, 1996). Abstract data types are mathematical formalisms rather than concrete objects stored on a computer's hardware. They are useful because they allow a programmer to focus on an object's functionality rather than its implementation details. A *collection* is an abstract data type that stores data elements. Common examples of collections include sets, maps, stacks, queues, and sequences. We briefly describe each of these collections here:

- A *set* is a collection that does not contain duplicate elements. Sets support the operations '`insert`', '`delete`', '`search`', and occasionally '`intersect`' and '`union`'. The operation '`insert`' adds a given element to the collection; '`delete`' removes a given element from the collection; '`search`' checks for the presence of a given element in the collection; and '`intersect`' and '`union`' implement standard set intersection and union procedures.
- A *map* is a set that consists of ordered pairs, called *key-value* pairs. A key x is said to be *associated* with a value y if the ordered pair (x, y) is an element of the collection. Maps impose the condition that each key be associated with a single value. Maps support the operations '`put`', '`get`', and '`remove`'. The operation '`put`' adds a given key-value pair to the collection; '`get`' queries the value associated with a given key; and '`remove`' deletes a given a key-value pair from the collection.
- A *stack* is a collection that supports the following two operations: '`push`' and '`pop`'. The operation '`push`' adds a given element to the collection, and '`pop`' removes the most recently added element from the collection.
- A *queue* is a collection that supports the following two operations: '`enqueue`' and '`dequeue`'. The operation '`enqueue`' adds a given element to the collection, and '`dequeue`' removes the least-recently added element from the collection.
- A *sequence* is an ordered collection of elements. The exact functionality of a sequence depends on its implementation, but typically sequences support the operations '`append`', '`replace`', '`getObjectAtIndex`', '`getIndexOfObject`', '`remove`', and '`search`'. The operation '`append`' adds a given element to the end of the collection; '`replace`' replaces a given element or the element located at a given index with another element; '`getObjectAtIndex`' returns the element located at a given index; '`getIndexOfObject`' returns the index (or indices) at which a given element is located; '`remove`' deletes a given element or the element located at a given index from the collection; and '`search`' checks for the presence of a given element in the collection. Familiar examples of sequences include R vectors and lists. Vectors in R are homogeneous sequences (i.e., can store a single type of object), while lists in R are heterogeneous sequences (i.e., can store different types of objects).

The names we use to refer to the different collection types are standard (Dale and Walker, 1996), with the possible exception of *sequence.* Many programmers prefer the term *list* to *sequence*, but we choose to use the term *sequence* to avoid confusion with R lists.

Abstract data types and data structures are related in that data structures are used to implement abstract data types. To illustrate, one could implement a sequence abstract data type using an array data structure (Figure 1) or a linked list data structure (Figure 2). If using an array, '`append`', for instance, involves inserting an element into the memory location following the memory location of the rightmost element. If using a linked list, on the other hand, '`append`' involves inserting a new node at the tail of the linked list. At the level of hardware, these procedures are clearly distinct. However, at a more abstract level, these procedures both implement the '`append`' operation for different implementations of the sequence abstract data type.

Methods

We searched the Comprehensive R Archive Network (CRAN) and Bioconductor for collection packages, evaluated the API and design of the packages returned by this search, and then ran benchmark tests

to assess the performance of these packages.

Searching CRAN and Bioconductor

We searched CRAN and Bioconductor for packages that implement one or more of the following collection types: map, set, stack, queue, or sequence. Table 1 lists the terms used in the search, the number of packages returned by the search, the number of packages excluded from review after the search, and the names of the packages ultimately selected for review. We searched CRAN and Bioconductor because these platforms are the two most popular repositories for R packages, and because packages must pass a review before publication on these platforms. CRAN packages published through July 3, 2017, and Bioconductor packages published through March 24, 2018, were included in the search.

The vast majority of packages we excluded from review were unrelated to either collections or data structures. For example, the search term "array" returned many packages related to the analysis of microarray data. Some excluded packages, however, did relate to collections or data structures. We note those packages here.

- The packages **hashr** (van der Loo, 2015) and **hashFunction** (Zhan, 2013) implement functions that generate hash codes for R objects. While hashing plays an important role in the construction of hash tables, **hashr** and **hashFunction** do not implement complete map classes.
- The package **filehashSQLite** (Peng, 2012) implements a key-value map using SQLite as the backend. For the sake of simplicity, we neglected **filehashSQLite** to focus exclusively on its more popular companion package, **filehash** (Peng, 2006).
- The package **tictoc** (Izrailev, 2014) implements a complete stack class. The primary purpose of **tictoc** is to benchmark R code; the stack class is tangential. Moreover, the implementation of the **tictoc** stack class is virtually identical to that of a dedicated collection package already selected for review (namely, **liqueueR**).
- The package **DSL** (Bengtsson, 2015) implements a sequence class that can be stored in a distributed manner. While important, distributed computing is outside the scope of the current work.
- The packages **bit64** (Oehlschlägel, 2017) and **bit** (Oehlschlägel, 2014) implement sequence classes that can efficiently store 64-bit integers and 1-bit booleans, respectively. While useful in certain contexts, **bit64** and **bit** are too specialized for inclusion this article.
- The package **Oarray** (Rougier, 2013) implements a sequence class that supports 0-based indexing. This class is too similar to the standard base R vector for inclusion in this review.

Collection	Search terms	N. packages returned	N. packages excluded	Packages kept
Set	'set,' 'hash'	448	447	**sets** (Meyer and Hornik, 2009)
Map	'map,' 'associative array,' 'dictionary,' 'dictionaries,' 'hash,' 'symbol table'	257	254	**filehash** (Peng, 2006), **hash** (Brown, 2013), **hashmap** (Russell, 2017)
Stack	'stack,' 'last-in-first-out,' 'LIFO'	14	9	**rstackdeque** (O'Neil, 2015), **rstack** (Csárdi, 2016) **liqueueR** (Collier, 2016), **dequer** (Schmidt, 2016), **flifo** (Poncet, 2017)
Queue	'queue,' 'first-in-first-out,' 'FIFO'	7	4	**rstackdeque** (O'Neil, 2015), **liqueueR** (Collier, 2016), **dequer** (Schmidt, 2016)
Sequence	'sequence,' 'list,' 'vector,' 'array'	449	446	**listenv** (Bengtsson, 2015), **S4Vectors** (Pagés et al., 2017), **stdvectors** (Giuliano, 2017)

Table 1: Details of the search for collection packages on CRAN and Bioconductor.

We selected a total of 11 packages from CRAN (**filehash**, **hash**, **hashmap**, **sets**, **rstackdeque**, **rstack**, **liqueueR**, **dequer**, **flifo**, **listenv**, and **stdvectors**) and one package from Bioconductor (**S4Vectors**) for review.

Analysis of packages

We analyzed the design and features of the packages selected for review. For a given package (e.g., **rstackdeque**) and collection class implemented by that package (e.g., "`stack`"), we determined the following:

1. API of the class
2. types of R objects that can be stored by the class
3. language in which the class was implemented (e.g., pure R, C++, C, etc.)
4. data structure(s) used to implement the class
5. algorithmic complexity of the operations provided by the class
6. whether the operations provided by the class have side effects or are side effect-free
7. whether the class was implemented using the S3, S4, or R6 object-oriented system
8. any unusual features or limitations of the class.

We conducted this analysis by reading package vignettes and manuals, corresponding with package authors, and examining package source code.

Benchmark tests

We wrote and ran scripts to measure the execution times of key functions implemented by the packages selected for review. We refer to these trials of execution times as *benchmark tests*. For a given collection type, abstract operation defined on that collection type, and package implementing that collection type, we measured the execution time of the given function by the given package. We repeated this test 25 times each for collection objects containing different numbers of elements, and then plotted mean execution time against object size for that collection type, package, and operation. The commands used to generate the benchmark test results are given in the appendix.

As an example of this procedure, consider the map collection type. For a given package p among those that implement the map collection type (**filehash**, **hash**, and **hashmap**), for a given operation o among those defined on the map collection type ('`put`', '`get`', and '`remove`'), and for a positive integer n, we measured the median execution time t of 25 calls to method o of a map object from package p containing n elements. We then plotted the point (n, t) on the graph of *median function execution time vs. number of elements in collection* corresponding to package p and operation o. For a given function, the plot of *median function execution time vs. number of elements in collection* reflects function scalability (or lack thereof): plots with shallow growth suggest the function scales well, while plots with steep growth suggest the function does not scale as well. The values of n we tested ranged from 100 to 100,000, and are similar to the values of n used by other authors for benchmark tests (e.g., O'Neil (2015), Russell (2017)). Benchmark test result plots are available in the appendix.

To help readers interpret benchmark test results, we developed our own custom implementation of each collection type using a base R list. For example, we implemented the map abstract data type using a named list, wherein names served as keys and entries served as values. The '`put`', '`get`', and '`remove`' operations were implemented by appending data to, indexing, and subsetting the list, respectively. Given the centrality and importance of the list to R programming, our base R list implementation of a given collection type can be seen as the "control" or "default" implementation of that collection type.

We measured function execution times using the package **microbenchmark** (Mersmann, 2015). The **microbenchmark** package provides a function that measures the execution time of code with sub-millisecond accuracy. The package uses the operating system-level routine `mach_absolute _time` on MacOSX and is more precise than R's native `System.time` function. Scripts were run in R version 3.4.1 (2017-06-30) using an Apple computer with an Intel Core i5 CPU (2.5 GHz, 2 cores, 4 logical processors), 4 GB of RAM, and 500 GB of storage. Additional details on the benchmark tests are available in the appendix.

Results

Sets

The only package we located on CRAN that implements the set abstract data type is **sets**. The **sets** package is rich in features and provides broad support for basic sets, generalized sets (i.e., sets in which elements are mapped to a real number), and customizable sets (i.e., sets in which the user can specify iteration order and the function used to check for element equality). The basic set class is called "`set`." The "`set`" class is equipped with a wide variety of methods, including '`search`', '`union`', '`intersection`', '`complement`', '`power set`', '`Cartesian product`', and more. The "`set`" class does not have '`insert`' and '`delete`' methods per se, but its '`union`' and '`complement`' methods implement these operations in effect. The package is written predominantly in R and partially in C. Objects of type "`set`" are S3 objects and operations on "`set`" objects are side-effect free.

Under the hood, "`set`" objects are R lists that are ordered to allow for the comparison of nested sets. The function '`search`' essentially creates a hash table, inserts the elements of the calling set into the hash table, and then checks for the existence of the queried element. This procedure is $\mathcal{O}(n)$. The methods '`insert`' (as implemented by calling '`union`' on the set and the object to be added) and '`remove`' (as implemented by calling '`complement`' on the set and the object to be removed) are more involved. Ultimately, both methods perform a comparison-based sort and are therefore $\mathcal{O}(n\ log(n))$ at best. The **sets** package performed worse than a base R List implementation of the set abstract data type on all operations and for all set sizes (Figure 5).

Maps

Among the packages selected for review, **hash**, **hashmap**, and **filehash** implement the map abstract data type.

The map class of the package **hash** is called "`hash`." The "`hash`" class implements the standard map API, as well as methods to invert a "`hash`" object and return the number of key-value pairs in a "`hash`" object. Keys stored in "`hash`" objects must be atomic character vectors, but values can be any R object. The "`hash`" class is essentially a convenience wrapper around R environments. Because environments are implemented internally as hash tables (Wickham, 2014), the '`put`', '`get`', and '`remove`' methods of the "`hash`" class are $\mathcal{O}(1)$ amortized. The "`hash`" class is implemented in S4, and its methods (in general) have side-effects.

The **hashmap** package provides a map class called "`Hashmap`." The "`Hashmap`" class implements a number of useful methods in addition to '`put`', '`get`', and '`remove`', such as methods to return a named vector representation of a "`Hashmap`" object and merge "`Hashmap`" objects. "`Hashmap`" objects can store a variety of atomic vector types as keys and values, but not general R objects. The **hashmap** package is implemented in C++, and the "`Hashmap`" class is essentially a wrapper around a hash table class in the C++ Standard Template Library. Accordingly, '`put`', '`get`', and '`remove`' operations are $\mathcal{O}(1)$ amortized. The "`Hashmap`" class is implemented in S4 and has methods that (in general) operate via side effects. An interesting feature of **hashmap** is that the '`put`', '`get`', and '`remove`' methods are efficiently vectorized.

The map class of the package **filehash**, called "`filehash`," is rather different from "`hash`" and "`Hashmap`." While "`hash`" and "`Hashmap`" store keys and values in working memory, "`filehash`" stores keys and values on disk. This allows the user to work with datasets that take up more space than is available in RAM. The API of **filehash** includes the standard map operations as well as methods to test for the existence of a key and return the number of items in a "`filehash`" object. Keys must be atomic character vectors, while values can be arbitrary R objects. The "`filehash`" class is implemented predominantly in R, but C code powers file input and output. "`filehash`" objects are S4 objects, and operations on "`filehash`" objects (in general) have side effects.

The data structure underlying a "`filehash`" object is somewhat complex. Essentially, a "`filehash`" object is a list containing a reference to a file on disk and a reference to an R environment. The file on disk stores keys and values in serialized format, and the environment maps each key to the byte location of its associated value in the file on disk. The '`put`' and '`remove`' operations are $\mathcal{O}(1)$. The running time of '`get`' is proportional to the number of elements inserted into the map object since '`get`' was last called. Thus, if a user calls '`get`' twice after inserting a large number of elements, the first call will be slow while the second will run in constant time. We expect '`get`' to be a fast operation in most cases.

The **hash**, **hashmap**, and **filehash** packages all performed well on benchmark tests compared to a base R list implementation of the map abstract data type. The base list was fastest for small datasets. However, **hash**, **hashmap**, and **filehash** began outperforming the base list at around 750 - 4,000 elements, 2,000 - 15,000 elements, and 10,000 - 70,000 elements, respectively (Figure 6).

Stacks

Our search for packages that implement the stack abstract data type returned **rstackdeque**, **dequer**, **liqueueR**, **flifo**, and **rstack**.

The package **rstackdeque** provides a stack class called "rstack." The "rstack" class implements standard 'push' and 'pop' operations, as well as functions to obtain the length of a stack, reverse the order of elements in a stack, and inspect the first element of a stack. Internally, an "rstack" object is a singly linked list of R environments. Accordingly, an "rstack" object can store any R object and supports $\mathcal{O}(1)$ 'push' and 'pop'. "rstack" objects are implemented in S3, and operations on "rstack" objects are side-effect free. Thus, **rstackdeque** allows for functional programming, which in fact is one of its major strengths.

The stack class provided by the package **dequer** is called "stack." The "stack" class implements the standard stack API, as well as a method to print the top element of a stack. Instances of "stacks" are S3 objects and can store any type of R object. Internally, "stacks" are implemented as doubly-linked lists in C. Thus, 'push' and 'pop' are $\mathcal{O}(1)$ operations. The **dequer** package has two limitations that should be noted. First, the only way to extract elements from a "stack" is to coerce the entire "stack" into an R list; this procedure is $\mathcal{O}(n)$. Second, due to implementation details of the R protection stack, it is really only possible to instantiate and use a single "stack" object per R session (Schmidt, personal communication).

The class "Stack" from the package **liqueueR** is a simple implementation of the stack abstract data type. "Stack" implements the operations 'push' and 'pop', as well as methods to query the size of a "Stack" object and combine "Stack" objects. Internally, "Stack" objects are R lists. The 'push' operation is implemented by appending an element to the end of the list, and the 'pop' operation is implemented by removing the last element of the list. Because R lists are copied when modified, 'push' and 'pop' are $\mathcal{O}(n)$ operations. "Stacks" can store any type of R object. The "Stack" class is implemented using R6, and so 'push' and 'pop' have side-effects.

The package **flifo** is similar to **liqueueR** in that **flifo** implements the stack abstract data type using an R list. Moreover, the stack class provided by **flifo**, called "lifo," operates via side-effects. There are, however, several unusual design features of **flifo** that warrant special attention. First, when pushing a variable onto a "lifo" object, the variable is deparsed and then deleted from the calling environment. For example, the following code, if executed in the global environment, will result in the deletion of the variable `myVariable`: '`myStack <-lifo(); myVariable <-1:10; push(myStack,myVariable)`'. Second, **flifo** implements "lifo" objects using the S3-object oriented system. Because operations on "lifo" objects are designed to have side-effects, the 'push' and 'pop' functions must record the calling "lifo" object's name and then reassign this object (by name) to the calling environment. This behavior could be more naturally achieved if "lifo" objects were instead implemented using R6. Due to **flifo**'s unusual design features and similarity to **liqueueR**, we choose not to benchmark **flifo**.

Finally, the packages **rstack** and **filehash** implement stack classes, but we choose not to benchmark these classes. The stack class from **rstack** is implemented using the R6 reference system and is designed to support $\mathcal{O}(1)$ 'push' and 'pop'. However, due to a bug in the R6 reference system, the 'push' and 'pop' methods of this class are currently $\mathcal{O}(n)$ (Csàrdi, personal communication). The package **filehash**, which implements the map abstract data type (as described in the *Maps* section), incidentally implements a stack class as well. We discovered this only after downloading and exploring the package, because **filehash** was not returned in our search stack packages on CRAN and Bioconductor (see Table 1). We choose not to benchmark the stack class provided by **filehash** because it is essentially a variation on the map class.

The packages **dequer** and **Rstackdequeue** performed favorably compared to the base R list implementation of a stack, while the package **LiqueuR** performed less favorably. **dequer** and **Rstackdequeue** begain outperforming the base R list at about 700-1,000 elements and 9,000-10,000 elements, respectively. **LiqueuR** exhibited strictly worse performance than the base R list (Figure 7).

Queues

Of the packages that implement stacks (**rstackdeque**, **dequer**, **liqueueR**, **flifo**, and **rstack**), all but **rstack** also implement queues. Overall, the queue classes provided by these packages are quite similar to the corresponding stack classes. The queue class of **rstackdeque** is implemented using linked lists of R environments, supports amortized $\mathcal{O}(1)$ 'enqueue' and 'dequeue', and is side-effect free. **liqueueR**'s queue class is implemented as a simple R list, supports $\mathcal{O}(n)$ 'enqueue' and 'dequeue', and has side-effects. The queue class of **flifo** is virtually the same as that of **liqueueR** in terms of implementation and algorithmic complexity. We do not run benchmark tests on the **flifo** queue class for the same reasons we do not run benchmark tests on the **flifo** stack class. It turns out that **filehash** also implements a queue class, which we likewise do not benchmark for reasons given in the previous

section.

The queue class of **dequer**, much like the stack class of **dequer**, is implemented as a doubly-linked list of nodes, is written in C, and has methods that operate via side-effects. The queue class of **dequer**, however, appears to have some bugs that the stack class of **dequer** does not. First, **dequer**'s queue class works only when it has fewer than approximately 10^6 elements. Second, benchmark tests suggest the 'dequeue' method of **dequer**'s queue class runs in $\mathcal{O}(n)$ time (Figure 8). In theory, this method should run in $\mathcal{O}(1)$ time. The cause of this discrepancy is unclear despite an extensive examination of source code.

The results of the benchmark tests of the queue packages were broadly similar to those of stack packages. The **liqueueR** package performed strictly worse than the base R list implementation of a queue. **Rstackdeque** began outperforming the base R list at about 20,000 - 60,000 elements. Finally, **dequer** was strictly slower than the base R list with respect to the 'dequeue' operation, but was faster with respect to the 'enqueue' operation on queues containing more than 1,000-2,000 elements (Figure 8).

Sequences

Among the packages under review, **stdvectors**, **listenv**, and **S4Vectors** implement the sequence abstract data type.

The package **stvectors** implements a sequence class called "stdvector." The "stdvector" class has a small API: "stdvector" objects support the operations 'append', 'replace', 'getObjectAtIndex', and 'remove' (by index), but not 'search', 'getIndexOfObject', or 'remove' (by element). Internally, a "stdvector" object is a C++ array that grows dynamically. Accordingly, the functions 'getIndexOfObject' and 'replace' are $\mathcal{O}(1)$, the function 'append' is $\mathcal{O}(1)$ amortized, and the function 'remove' (when called on an element in the middle of the sequence) is $\mathcal{O}(n)$. Objects of class "stdvector" are implemented in S3 and can store any type of R object. Operations on "stdvector" objects (in general) have side-effects.

The sequence class provided by the **listenv** package is called "listenv." The "listenv" class has the same API of the "stdvector" class (described above), is written in pure R, and can store any type of R object. Internally, a "listenv" object is a list containing a character vector and a reference to an R environment. Elements are stored in the environment and are indexed by names that are stored in the character vector. The methods 'append' and 'remove' involve modifying both the R environment and the character vector. Because character vectors are copied upon modification, 'append' and 'remove' are $\mathcal{O}(n)$ operations. The methods 'getObjectAtIndex' and 'replace' involve first copying the character vector and then performing a lookup in the environment. The 'getObjectAtIndex' and 'replace' methods are $\mathcal{O}(n)$, but grow very slowly in execution time as n increases. Operations on "listenv" objects have side-effects, and the "listenv" class is implemented using the S3 system.

S4Vectors is a large and richly-featured package that provides broad support for the sequence abstract data type. At the core of **S4Vectors** is "Vector," an abstract class (i.e., class that cannot be instantiated) that supports basic sequence functionality. Most of the classes implemented by **S4Vectors** extend "Vector" in some useful way. For example, the "Rle" class efficiently represents sequences that contain long subsequences of repeating elements; the "Pairs" class stores sequences consisting of ordered pairs; and the "LLint" class stores sequences consisting of very large integers. Users can easily write their own, custom extension of "Vector" class as well. **S4Vectors** is written in both R and C, and the many classes provided by **S4Vectors** are implemented using the S4 object-oriented system. The package **IRanges** (Lawrence et al., 2013), a companion package of **S4Vectors**, implements many additional helpful extensions of the "Vector" class.

We decided to run benchmark tests on the "SimpleList" class of the **S4Vectors** package. The "SimpleList" class is directly analogous to a base R list and provides essentially the same functionality as a base R list. In fact, under the hood, a "SimpleList" is an S4 object containing a base R list. The operations 'append', 'replace', and 'remove' run in $\mathcal{O}(n)$ time on objects of this class, and the 'getObjectAtIndex' operation runs in $\mathcal{O}(1)$ time. It should be noted that the "SimpleList" class is fairly generic; more specialized extensions of the "Vector" class (e.g., "Rle") will likely perform better under certain conditions.

The benchmark test results for packages implementing the sequence abstract data type were mixed. The **stdvectors** package was faster than the base R list with respect to the 'append' and 'remove' operations on objects containing more than 1,000 - 2,000 elements, but was strictly slower with respect to the other operations. The **listenv** package outperformed the base R list with respect to the 'append' operation on objects containing more than 4,000 elements, but performed strictly worse with respect to the other operations. Finally, the **S4Vectors** package was strictly slower than the R list with respect to all operations (Figure 9).

Discussion

The state of R collection packages

CRAN and Bioconductor have many collection packages available for download. At present, deciding which package to use is challenging given the variety of options. In this section we provide practical recommendations to users exploring the collection package ecosystem.

Among the packages we reviewed that implement the map collection (**filehash**, **hash**, and **hashmap**), the best package for most situations is probably **hash**. The package **hash** is fast, and **hash** map objects can store any type of R object. The package **hashmap** has efficiently vectorized functions, but **hashmap** appears to be slower than **hash** and the map class it implements cannot store general R objects. The package **filehash** is useful when working with large datasets (i.e., datasets that consume more memory than is available in RAM), but should be avoided in favor of **hash** when working with smaller datasets.

The only package we located on CRAN that implements the set abstract data type is **sets**. The **sets** package is versatile and provides a rich collection of functions and classes. However, **sets** is slow; '`search`' runs in linear time, and '`insert`' and '`remove`' run in superlinear time. These functions would run in constant time if "`set`" objects were implemented as hash tables rather than R lists. Users interested in working with a solid implementation of the set abstract data type, and less interested in speed, should consider the **sets** package.

We consider **rstackdeque** to be the best package for general-purpose use among the packages we reviewed that implement the stack collection (**rstackdeque**, **dequer**, **liqueueR**, **flifo**, and **rstack**). The **rstackdeque** package is reasonably fast, fully-featured, and supports functional programming. The package **dequer** is faster than **rstackdeque**, but does not allow for functional programming and suffers from several bugs and drawbacks (see *Results*). The packages **liqueueR**, **flifo**, and **rstack** do not allow for functional programming and implement $\mathcal{O}(n)$ '`push`' and '`pop`', which we consider to be prohibitively slow. We also recommend **rstackdeque** among the packages we reviewed that implement the queue abstract data type for similar reasons.

A standard R list is the probably the best implementation of the sequence abstract data type in most situations. R lists are fast and support the full sequence API (as defined in the *Background* section). The **S4Vectors** package provides a wide range of classes that implement sequences. The class that we benchmarked, '`SimpleList`', is slower than a base R list. However, many of the classes implemented by **S4Vectors** (and its companion package **IRanges**) are excellent choices for more specialized, structured sequence data. The sequence class implemented by **stdvectors** outperforms the base R list with respect to the operations '`append`' and '`remove`' for large datasets. However, **stdvectors** does not support the full sequence API and is slower than a base R list with respect to the other sequence operations. The **listenv** sequence class is, for the most part, slower than a base R list. Also, it implements fewer methods.

Many R users only use the data structures provided by base R. While base data structures are reasonably fast, there are cases in which base data structures are much slower than data structures implemented by collection packages. For example, the "`rstack`" class from **rstackdeque** is about an order of magnitude faster than a base R list when operating on large datasets. R programmers should consider using the packages recommended above to speed up program execution times when efficiency is important.

Most of the collection packages reviewed in this paper are used by other packages as imports or dependencies. For instance, **hash** is a dependency of the package **neuroim** (Buchsbaum, 2016). **neuroim** allows users to store and manipulate brain imaging data, and the map class from **hash** serves to associate character identifiers with different brain regions. The **FindMinIC** (Lange et al., 2013) package uses **sets**. **FindMinIC** creates models from all subsets of a given list of predictor variables and then rank-orders the models by information criterion (e.g., AIC). The '`power set`' operator from **sets** is used to generate the space of possible models. **S4Vectors** is among the top 5% of most downloaded packages on Bioconductor and is therefore imported by a large number of other packages. One such package is **GenomicRanges** (Lawrence et al., 2013), which uses **S4Vectors** to represent genomic annotations and alignments. These examples, and many others, illustrate the usefulness of collection packages in R software development.

Looking forward

The collection package ecosystem in R is young and could be improved in several ways. First, set objects could be implemented as hash tables rather than R lists (as is the case with **sets**) to maximize the efficiency of set operations. Second, stack and queue objects could be stored internally as linked lists of *nodes* rather than *environments* (as is the case with **rstackdeque**). Environments in R are hash

tables by default, and are therefore more expensive in space than nodes, which are structures that only store data and a reference to another node. Third, every collection object could be equipped with an iterator to facilitate iteration through the elements of the collection. Minor improvements related to the implementation of the map and sequence abstract data types are also possible.

One compelling way to implement these improvements would be to create a single, unified collection package. Such a package might have "`Map`," "`Set`," "`Stack`," "`Queue`," and "`Sequence`" classes that derive from a generic "`Collection`" abstract class. The "`Collection`" class might have methods to check for the presence of an element ('`contains`' or '`%in%`'), return the number of elements stored ('`size`'), return an iterator ('`iterator`'), and print to the console ('`print`'). The derived classes would implement these methods as well as their own, specialized methods (Figure 4). Each class might be implemented in C or in C++ via the package **Rcpp**. In addition to offering probable performance improvements over current collection packages, such a package would provide a consistent API between different collection classes and aggregate important abstract data types into a single location. We anticipate that developers will disagree on the specific implementation details of the proposed package. For instance, some may want to implement purely functional data structures, while others will prefer non-functional data structures; some may want to implement the classes using the S3 object oriented system, while others will prefer S4, and still others R6; some may want to implement additional abstract data types (e.g., priority queues, graphs, etc.), while others will prefer to keep the package lean and implement only those abstract data types mentioned above. Designing and writing the proposed package will be a major project. The R community should take inspiration from the latest in functional (e.g., OCaml, Haskell) and scripting (e.g., Python, Ruby) languages as it considers this possibility.

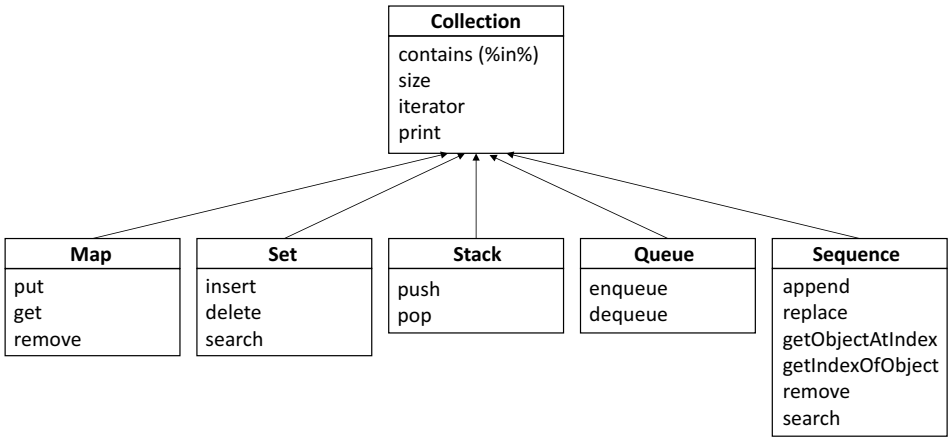


Figure 4: Inheritance diagram of a package that implements the map, set, stack, queue, and sequence collections.

Package authors can take away several general strategies for improving package performance based on the analysis of the packages reviewed for this article. First, authors should avoid unnecessarily copying objects inside functions (as **listenv** does, for instance). Copying objects, especially those that are large, is time-intensive. Authors should also keep in mind that most arguments in R are passed by deep copy to functions. Passing arguments by reference instead (as **rstackdeque** does by using environments) can accelerate function execution times. Second, authors should avoid adding gratuitous side-effects to functions that could be written without side-effects. At its core, R is a functional language. Users expect the data they pass into functions to remain unchanged. Functions with seemingly unnecessary side-effects (like those from **liqueueR** and **flifo**) may catch users off-guard. That said, some functions (like '`insert`' on map data structures) require side-effects to run in constant time. Finally, authors should remember that certain R functions and data structures have $\mathcal{O}(n)$ complexity "hidden" in them. The '`dequeue`' function from **dequer**, and the '`push`' and '`pop`' functions from **rstack**, are supposed to run in $\mathcal{O}(1)$ time. Unfortunately, these functions appear to run in $\mathcal{O}(n)$ time for reasons that are not immediately obvious. Authors can detect surprises like these by benchmarking code before package release.

The utility of reviewing R packages

R's diverse selection of packages is one of its greatest strengths, but also a source of frustration for package users and developers. Users often must sift through many packages to find one that suits their needs, and developers sometimes inadvertently write packages that essentially replicate

the functionality of packages that already exist on CRAN or Bioconductor. A good way for the R community to address this issue is to write more package reviews. Reviews can help users quickly find the best package for their purposes, avert authors from developing packages that in effect already exist, and highlight deficiencies in the R package ecosystem. There exist very few review articles in the archives of *The R Journal* or *The Journal of Statistical Software*, the two academic journals most directly related to R. Given the growth in popularity of R and the proliferation of R packages, we think that additional review articles would be helpful.

Summary

For many statisticians, programmers, and scientists, R is the go-to tool for data processing, modeling, and visualization. However, R's lack of base support for collection data types can be a hindrance, especially as data sets continue to grow in size. In this article, we reviewed collection packages – packages that implement collection data types like sets, maps, stacks, queues, and sequences. We demonstrated that data types implemented by collection packages are in many cases faster than corresponding base R data types, provided specific package recommendations to users navigating the collection package ecosystem, and gave several suggestions for improving the collection capabilities of R. An interesting line of future work would be to investigate collection packages that implement more advanced abstract data types, such as graphs and priority queues. Such data types, though less common than the ones explored in this review, are quite useful in a variety of applications. Data structures and algorithms lie at the core of computer science. As R users develop new collection packages, and effectively use those that already exist, they will wield the language more powerfully.

Acknowledgments

We thank the Howard Hughes Medical Institute for supporting this study. We also thank Jason Brunson, Eliezer Gurarie, and two anonymous reviewers for reading and providing helpful comments on the manuscript.

Bibliography

H. Bengtsson. *Listenv: Environments Behaving (Almost) as Lists*, 2015. URL http://CRAN.R-project.org/package=listenv. R package version 0.6.0. [p455, 458]

P. E. Black. Dictionary of algorithms and data structures. *NISTIR*, 1998. [p456]

C. Brown. *hash: Full feature implementation of hash/associated arrays/dictionaries*, 2013. URL http://CRAN.R-project.org/package=hash. R package version 2.2.6. [p455, 458]

B. R. Buchsbaum. *neuroim: Data Structures and Handling for Neuroimaging Data*, 2016. URL https://CRAN.R-project.org/package=neuroim. R package version 0.0.6. [p463]

A. Collier. *liqueueR: Implements Queue, PriorityQueue and Stack Classes*, 2016. URL http://CRAN.R-project.org/package=liqueueR. R package version 0.0.1. [p455, 458]

T. H. Cormen. *Introduction to Algorithms*. MIT press, 2009. [p455, 456]

G. Csárdi. *rstack: Stack Data Type as an 'R6' Class*, 2016. URL http://CRAN.R-project.org/package=rstack. R package version 1.0.0. [p455, 458]

N. Dale and H. M. Walker. *Abstract Data Types: Specifications, Implementations, and Applications*. Jones & Bartlett Learning, 1996. [p455, 457]

D. Eddelbuettel, R. François, J. Allaire, J. Chambers, D. Bates, and K. Ushey. Rcpp: Seamless R and C++ integration. *Journal of Statistical Software*, 40(8):1–18, 2011. [p455]

M. Giuliano. *stdvectors: C++ Standard Library Vectors in R*, 2017. URL http://CRAN.R-project.org/package=stdvectors. R package version 0.0.5. [p455, 458]

S. Izrailev. *tictoc: Functions for timing R Scripts, as well as implementations of Stack and List structures*, 2014. URL http://CRAN.R-project.org/package=tictoc. R package version 1.0. [p458]

N. Lange, T. Fletcher, and K. Zygmunt. *FindMinIC: Find Models with Minimum IC*, 2013. URL https://CRAN.R-project.org/package=FindMinIC. R package version 1.6. [p463]

M. Lawrence, W. Huber, H. Pagès, P. Aboyoun, M. Carlson, R. Gentleman, M. Morgan, and V. Carey. Software for computing and annotating genomic ranges. *PLoS Computational Biology*, 9, 2013. URL https://doi.org/10.1371/journal.pcbi.1003118. [p462, 463]

B. Liskov and S. Zilles. Programming with abstract data types. In *ACM Sigplan Notices*, volume 9, pages 50–59. ACM, 1974. [p455]

O. Mersmann. *microbenchmark: Accurate Timing Functions*, 2015. URL http://CRAN.R-project.org/package=microbenchmark. R package version 1.4-2.1. [p459]

D. Meyer and K. Hornik. Generalized and customizable sets in R. *Journal of Statistical Software*, 31(2): 1–27, 2009. [p455, 458]

F. Morandat, B. Hill, L. Osvald, and J. Vitek. Evaluating the design of the R language. In *European Conference on Object-Oriented Programming*, pages 104–131. Springer, 2012. [p455]

R. A. Muenchen. The popularity of data analysis software, 2012. URL http://r4stats.com/popularity. [p455]

J. Oehlschlägel. *bit: A Class for Vectors of 1-Bit Booleans*, 2014. URL http://CRAN.R-project.org/package=bit. R package version 1.1-12. [p458]

J. Oehlschlägel. *bit64: A S3 Class for Vectors of 64bit Integers*, 2017. URL http://CRAN.R-project.org/package=bit64. R package version 0.9-7. [p458]

S. T. O'Neil. Implementing persistent O(1) stacks and queues in R. *R Journal*, 7(1), 2015. [p455, 458, 459]

H. Pagés, M. Lawrence, and P. Aboyoun. *S4Vectors: S4 Implementation of Vector-like and List-like Objects*, 2017. R package version 0.16.0. [p455, 458]

R. D. Peng. Interacting with data using the filehash package. *R News*, 6(4):19–24, 2006. URL https://cran.r-project.org/doc/Rnews/. [p455, 458]

R. D. Peng. *filehashSQLite: Simple Key-Value Database Using SQLite*, 2012. URL http://CRAN.R-project.org/package=filehashSQLite. R package version 0.2-4. [p458]

P. Poncet. *flifo: Don't Get Stuck with Stacks in R*, 2017. URL http://CRAN.R-project.org/package=flifo. R package version 0.1.4. [p455, 458]

J. Rougier. *Oarray: Arrays with Arbitrary Offsets*, 2013. URL http://CRAN.R-project.org/package=Oarray. R package version 1.4-5. [p458]

N. Russell. *hashmap: The Faster Hash Map*, 2017. URL http://CRAN.R-project.org/package=hashmap. R package version 0.2.0. [p455, 458, 459]

H. Schildt. *Java: The Complete Reference*. McGraw-Hill Education Group, 2014. [p455]

D. Schmidt. *dequer: Stacks, Queues, and Deques for R*, 2016. URL https://CRAN.R-project.org/package=dequer. R package version 2.0-0. [p455, 458]

B. Stroustrup. *The C++ Programming Language*. Pearson Education, 2013. [p455]

M. van der Loo. *hashr: Hash R Objects to Integers Fast*, 2015. URL http://CRAN.R-project.org/package=hashr. R package version 0.1.0. [p458]

G. van Rossum and F. L. Drake. *The Python Language Reference Manual*. Network Theory Ltd., 2011. [p455]

H. Wickham. *Advanced R*. CRC Press, 2014. [p455, 460]

X. Zhan. *hashFunction: A Collection of Non-Cryptographic Hash Functions*, 2013. URL http://CRAN.R-project.org/package=hashFunction. R package version 1.0. [p458]

Timothy Barry
University of Maryland Department of Biology
Biology-Psychology Building
USA
ORCiD: 0000-0002-4356-627X
timbarry@umd.edu

Appendix

Tables 3 – 6 display the code we used to benchmark the operations of the packages under review. In Table 3, the variables 'key' and 'value' represent a key-value pair. In Tables 2 - 6, the variable 'elem' represents a data element. In all tables, the variable 'y' represents the collection object itself. Benchmark test results are displayed in Figures 5 - 9. In all figures, results for different packages are plotted on different y-scales due to high between-package variance in execution time. The points represent median execution time. The colored bands represent the 25th and 75th percentiles of execution time. The code used to produce these plots is available at github.com/Timothy-Barry/Collections-in-R.

Sets

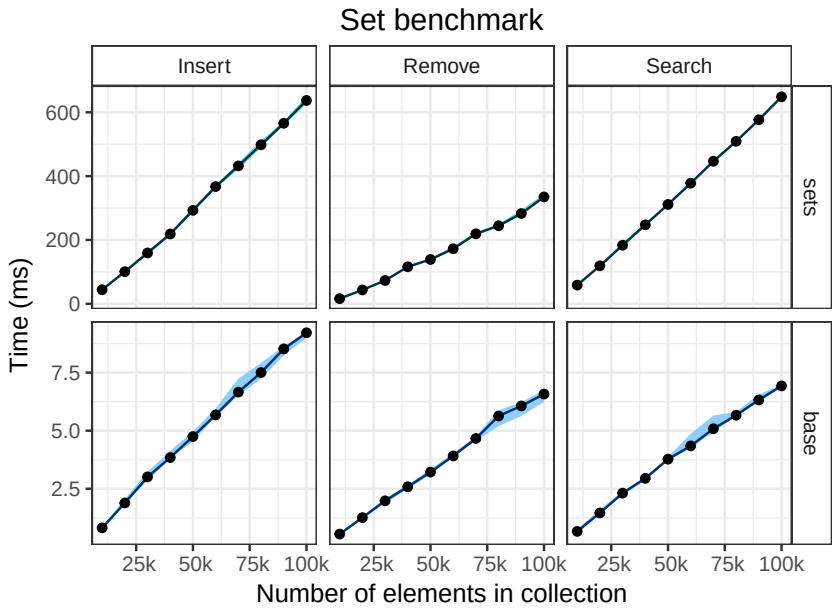


Figure 5: Benchmark test results the **sets** package.

	'Insert'	'Remove'
sets	`y <-set_union(y,elem)`	`y <-set_complement(elem,y)`
base	`if !(elem %in% y) c(y,elem) else y`	`y <-y[!(y == elem)]`
	'Search'	
sets	`set_contains_element(elem,y)`	
base	`elem %in% y`	

Table 2: Commands used in the benchmark for the set collection type.

Maps

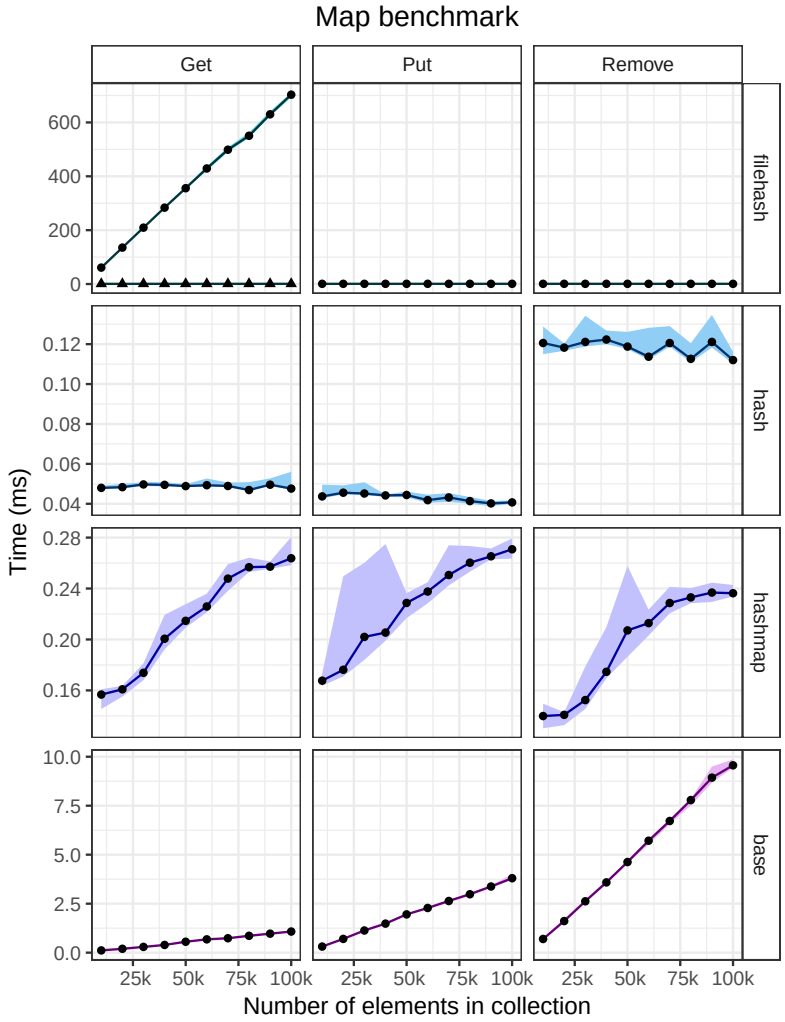


Figure 6: Benchmark test results for packages that implement the map abstract data type. The plot that shows the results of the 'get' benchmark for the package **filehash** has two lines. The line with circular points gives the running time of a single call to 'get'; the line with triangular points gives the running time of the second of two consecutive calls to 'get'.

	'Get'	'Put'	'Remove'
filehash	`dbFetch(y,key)`	`dbInsert(y,key,value)`	`dbDelete(y,key)`
hash	`y[[key]]`	`y[[key]] <-value`	`delete(key,y)`
hashmap	`y[[key]]`	`y[[key]] <-value`	`y$erase(key)`
base	`y[key]`	`y[key] <-value`	`y <-y[!(names(y) %in% key]`

Table 3: Commands used in the benchmark for the map collection type.

Stacks

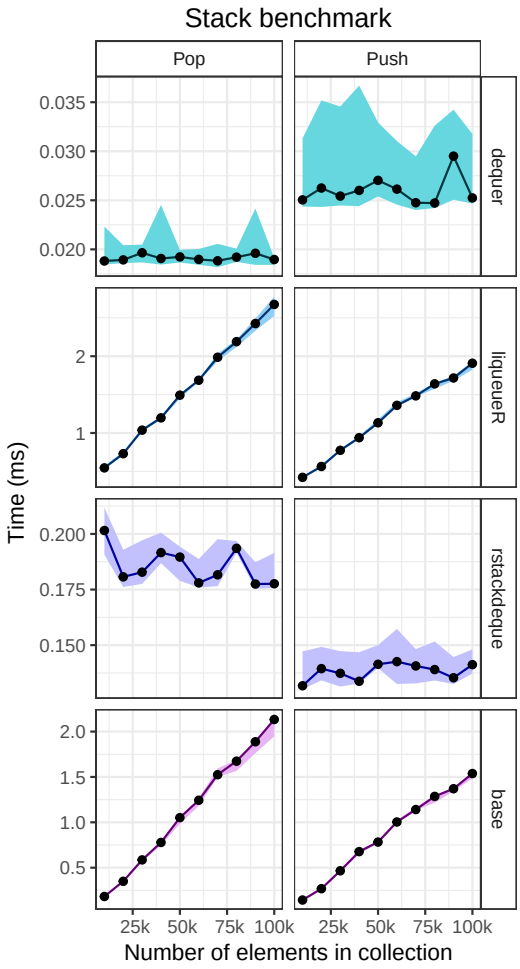


Figure 7: Benchmark test results packages that implement the stack abstract data type.

	'Pop'	'Push'
dequer	`pop(y)`	`push(y,elem)`
liqueR	`y$pop()`	`y$push(elem)`
rstackdeque	`without_top(y)`	`insert_top(y,elem)`
base	`y <-y[1:(length(y)-1)]`	`y <-c(y,elem)`

Table 4: Commands used in the benchmark for the stack collection type.

Queues

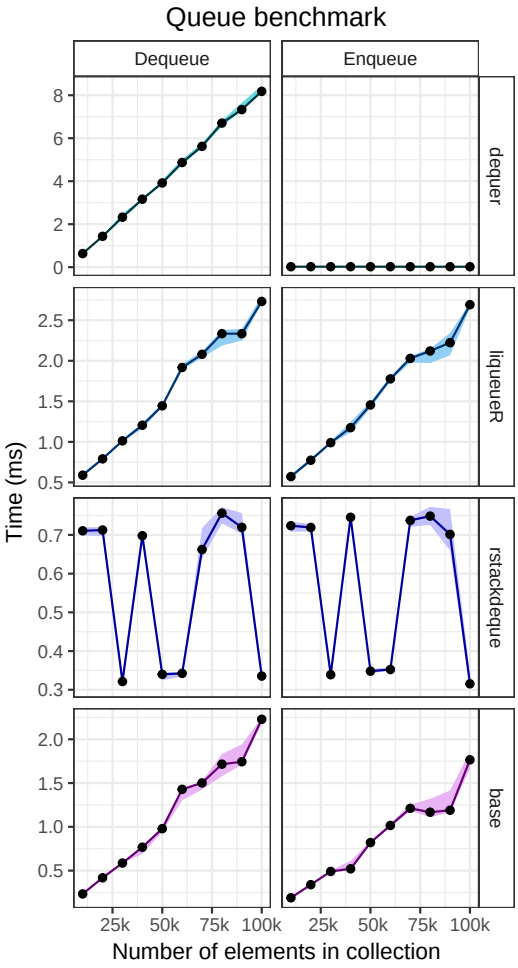


Figure 8: Benchmark test results of packages that implement the queue abstract data type.

	'Dequeue'	'Enqueue'
dequer	`pop(y)`	`pushback(y,elem)`
liqueueR	`y$pop()`	`y$push(elem)`
rstackdeque	`y <-without_front(y)`	`y <-insert_back(y,elem)`
base	`y <-c(elem,y)`	`y <-y[1:(length(y)-1)]`

Table 5: Commands used in the benchmark for the queue collection type.

Sequences

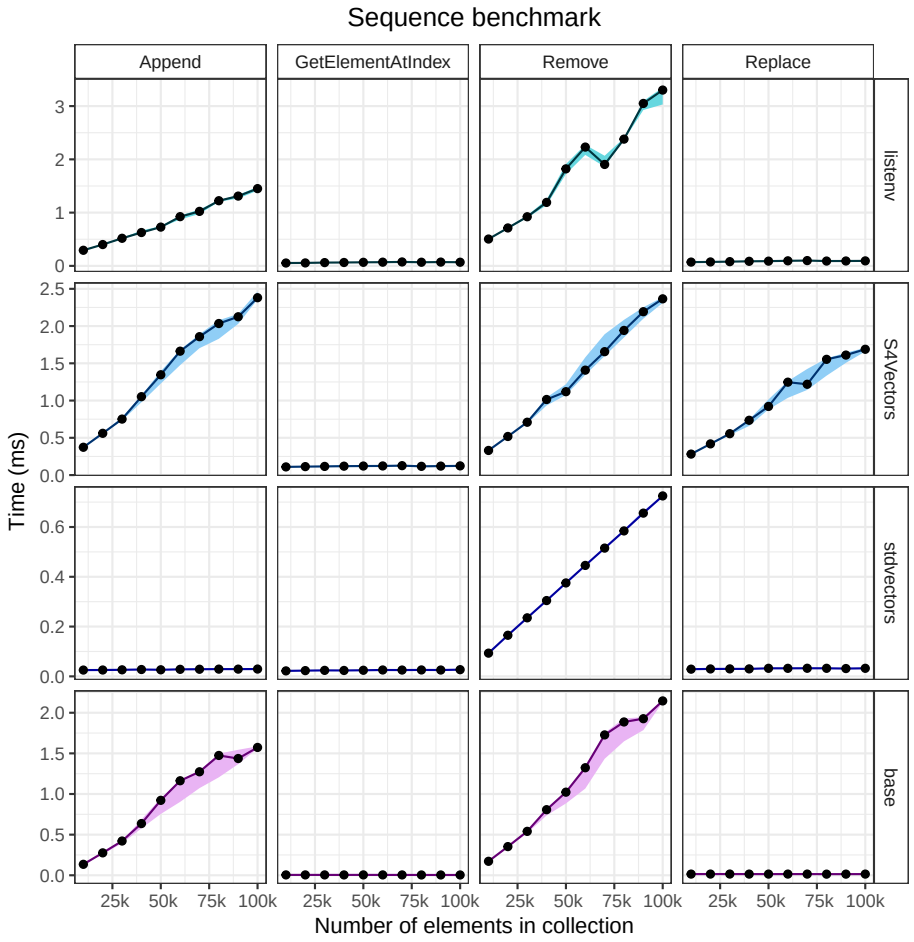


Figure 9: Benchmark test results for packages that implement the sequence abstract data type.

	'Append'	'GetElementAtIndex'
listenv	`y[[length(y) + 1]] <- elem`	`y[[index]]`
S4Vectors	`y[[length(y) + 1]] <-elem`	`y[[index]]`
stdvectors	`stdvectorPushBack(y,elem)`	`stdvectorSubset(y,index)`
base	`y <-c(y,elem)`	`y[index]`
	'Remove'	**'Replace'**
listenv	`y[[index]] <-NULL`	`y[[index]] <-elem`
S4Vectors	`y[[index]] <-NULL`	`y[[index]] <-elem`
stdvectors	`stdvectorErase(y,index,index)`	`stdvectorReplace(y,index,elem)`
base	`y <-y[-index]`	`y[index] <-elem`

Table 6: Commands used in the benchmark for the sequence collection type.

Approximating the Sum of Independent Non-Identical Binomial Random Variables

by Boxiang Liu and Thomas Quertermous

Abstract The distribution of the sum of independent non-identical binomial random variables is frequently encountered in areas such as genomics, healthcare, and operations research. Analytical solutions for the density and distribution are usually cumbersome to find and difficult to compute. Several methods have been developed to approximate the distribution, among which is the saddlepoint approximation. However, implementation of the saddlepoint approximation is non-trivial. In this paper, we implement the saddlepoint approximation in the **sinib** package and provide two examples to illustrate its usage. One example uses simulated data while the other uses real-world healthcare data. The **sinib** package addresses the gap between the theory and the implementation of approximating the sum of independent non-identical binomials.

Introduction

Convolution of independent non-identical binomial random variables appears in a variety of applications, such as analysis of variant-region overlap in genomics (Schmidt et al., 2015), calculation of bundle compliance statistics in healthcare organizations (Benneyan and Taşeli, 2010), and reliability analysis in operations research (Kotz and Johnson, 1984).

Computating the exact distribution of the sum of non-identical independent binomial random variables requires enumeration of all possible combinations of binomial outcomes that satisfy the totality constraint. However, analytical solutions are often difficult to find for sums of greater than two binomial random variables. Several studies have proposed approximate solutions (Johnson et al., 2005; Jolayemi, 1992). In particular, Eisinga et al. (2013) examined the saddlepoint approximation, and compared them to exact solutions. They note that in practice, these approximations are often as good as the exact solution and can be implemented in most statistical software.

Despite the theoretical development of aforementioned approximate solutions, a software implementation in R is still lacking. The **stats** package includes functions for frequently used distribution such as `dbinom` and `dnorm`, and less frequently used distributions such as `pbirthday`, but it does not contain functions for the distribution of the sum of independent non-identical binomials. The **EQL** package provides a `saddlepoint` function to approximate the mean of i.i.d. random variables, but does not apply to the case where the random variables are not identical. In this paper, we implement a saddlepoint approximation in **sinib** (sum of independent non-identical binomial random variables). The package provides the standard suite of distributional functions for the distribution (`psinib`), density (`dsinib`), quantile (`qsinib`), and random deviates (`rsinib`). The package is accompanied by a detailed documentation, and can be easily integrated into existing applications.

The remainder of this paper is organized as follows. We begin by providing an overview of the distribution of the sum of independent non-identical binomial random variables. Next, we give an overview of the saddlepoint approximation. The following section describes the design and implementation of the saddlepoint approximation in the **sinib** package. We provide two examples and assess the accuracy of saddlepoint approximation in these situations. The final section concludes and discusses future direction.

Overview of the distribution

Suppose $X_1,\ldots,X_m$ are independent non-identical binomial random variables such that $S_m = \sum_{i=1}^{m} X_i$. We are interested in finding the distribution of S_m.

$$P(S_m = s) = P(X_1 + X_2 + \cdots + X_m = s) \tag{1}$$

In the special case of $m = 2$, the probability simplifies to

$$P(S_2 = s) = P(X_1 + X_2 = s) = \sum_{i=0}^{s} P(X_1 = i)P(X_2 = s - i) \tag{2}$$

Computation of the exact distribution often involves enumerating all possible combinations of each variable that sum to a given value, which becomes infeasible when n is large. A fast recursion method to compute the exact distribution has been proposed (Butler and Stephens, 2016; Arthur Woodward and Palmer, 1997). The algorithm is as follows:

1. Compute the exact distribution of each X_i.
2. Calculate the distribution of $S_2 = X_1 + X_2$ using Equation 2 and cache the result.
3. Calculate $S_r = S_{r-1} + X_i$ for $r = 3, 4, \ldots, m$.

Although the recursion speeds up the calculation, studies have shown that the result may be numerically unstable due to round-off error in computing $P(S_r = 0)$ if r is large (Eisinga et al., 2013; Yili). Therefore, approximation methods are still widely used in literature.

Saddlepoint approximation

The saddlepoint approximation, first proposed by Daniels (1954) and later extended by Lugannani and Rice (1980), provides highly accurate approximations for the probability and density of many distributions. In brief, let $M(u)$ be the moment generating function, and $K(u) = \log(M(u))$ be the cumulant generating function. The saddlepoint approximation to the PDF of the distribution is given as:

$$\hat{P}_1(S = s) = \frac{\exp(K(\hat{u}) - \hat{u}s)}{\sqrt{2\pi K''(\hat{u})}} \tag{3}$$

where $\hat{u}$ is the unique value that satisfies $K'(\hat{u}) = s$.

Eisinga et al. (2013) applied the saddlepoint approximation to the sum of independent non-identical binomial random variables. Suppose that $X_i \sim Binomial(n_i, p_i)$ for $i = 1, 2, \ldots, m$. The cumulant generating function of $S_m = \sum_{i=1}^{m} X_i$ is:

$$K(u) = \sum_{i=1}^{m} n_i \ln(1 - p_i + p_i \exp(u)) \tag{4}$$

The first- and second-order derivatives of $K(u)$ are:

$$K'(u) = \sum_{i=1}^{m} n_i q_i \tag{5}$$

$$K''(u) = \sum_{i=1}^{m} n_i q_i (1 - q_i) \tag{6}$$

where $q_i = \frac{p_i exp(u)}{(1 - p_i + p_i \exp(u))}$.

The saddlepoint of $\hat{u}$ can be obtained by solving $K'(\hat{u}) = s$. A unique root can always be found because $K(u)$ is strictly convex and therefore $K'(u)$ is monotonically increasing on the real line.

The above shows the first-order approximation of the distribution. The approximation can be improved by adding a second-order correction term (Daniels, 1987; Akahira and Takahashi, 2001):

$$\hat{P}_2(S = s) = \hat{P}_1(S = s)\left\{1 + \frac{K''''(\hat{u})}{8[K''(\hat{u})]^2} - \frac{5[K'''(\hat{u})]^2}{24[K''(\hat{u})]^3}\right\} \tag{7}$$

where

$$K'''(\hat{u}) = \sum_{i=1}^{m} n_i q_i (1 - q_i)(1 - 2q_i)$$

and

$$K''''(\hat{u}) = \sum_{i=1}^{m} n_i q_i (1 - q_i)[1 - 6q_i(1 - q_i)]$$

Although the saddlepoint equation cannot be solved at boundaries $s = 0$ and $s = \sum_{i=1}^{m} n_i$, their exact probabilities can be computed easily:

$$P(S = 0) = \prod_{i=1}^{m} (1 - p_i)^{n_i} \tag{8}$$

$$P(S = \sum_{i=1}^{m} n_i) = \prod_{i=1}^{m} p_i^{n_i} \tag{9}$$

Incorporation of boundary solutions into the approximation gives:

$$\bar{P}(S=s) = \begin{cases} P(S=0), & s=0 \\ [1-P(S=0)-P(S=\sum_{i=1}^{m} n_i)]\frac{\hat{P}_2(S=s)}{\sum_{i=1}^{m} n_i \hat{P}_2(S=i)}, & 0<s<\sum_{i=1}^{m} n_i \\ P(S=\sum_{i=1}^{m} n_i), & s=\sum_{i=1}^{m} n_i \end{cases} \tag{10}$$

We have implemented Equation 10 as the final approxmation of the probability density function. For the cumulative density, Daniels (1987) gave the following approximator:

$$\hat{P}_3(S \geq s) = \begin{cases} 1-\Phi(\hat{w})-\phi(\hat{w})(\frac{1}{\hat{w}}-\frac{1}{\hat{u}_1}), & \text{if } s \neq E(S) \text{ and } \hat{u} \neq 0 \\ \frac{1}{2}-\frac{1}{\sqrt{2\pi}}\big[\frac{K'''(0)}{6K''(0)^{3/2}}-\frac{1}{2\sqrt{K''(0)}}\big], & \text{otherwise} \end{cases} \tag{11}$$

where $\hat{w} = \text{sign}(\hat{u})[2\hat{u}K'(\hat{u}) - 2K(\hat{u})]^{1/2}$ and $\hat{u}_1 = [1-\exp(-\hat{u})][K''(\hat{u})]^{1/2}$. The letters Φ and ϕ denotes the probability and density of the standard normal distribution.

The accuracy can be improved by adding a second-order continuity correction:

$$\hat{P}_4(S \geq s) = \hat{P}_3(S \geq s) - \phi(\hat{w})\Big[\frac{1}{\hat{u}_2}\Big(\frac{\hat{\kappa}_4}{8}-\frac{5\hat{\kappa}_3^2}{24}\Big)-\frac{1}{\hat{u}_2^3}-\frac{\hat{\kappa}_3}{2\hat{u}_2^2}+\frac{1}{\hat{w}^3}\Big] \tag{12}$$

where $\hat{u}_2 = \hat{u}[K''(\hat{u})]^{1/2}$, $\hat{\kappa}_3 = K'''(\hat{u})[K''(\hat{u})]^{-3/2}$, and $\hat{\kappa}_4 = K''''(\hat{u})[K''(\hat{u})]^{-2}$.

We have implemented Equation 12 to approximate the cumulative distribution.

The sinib package

The package uses only base **R** and the **stats** package to minimize compatibility issues. The arguments for the functions in the **sinib** package are designed to have similar meaning to those in the **stats** package, thereby minimizing the learning required. To illustrate, we compare the arguments of the `*binom` and the `*sinib` functions.

From the help page of the binomial distribution, the arguments are as follows:

- x, q: vector of quantiles.
- p: vector of probabilities.
- n: number of observations.
- size: number of trials.
- prob: probability of success on each trial.
- log, log.p: logical; if TRUE, probabilities p are given as log(p).
- lower.tail: logical; if TRUE (default), probabilities are $P[X \leq x]$, otherwise, $P[X > x]$.

Since the distribution of sum of independent non-identical binomials is defined by a vector of trial and probability pairs (each pair for one constituent binomial), it was neccessary to redefine these arguments in the `*sinib` functions. Therefore, the following two arguments were redefined:

- size: integer vector of number of trials.
- prob: numeric vector of success probabilities.

All other arguments remain the same. It is worth noting that when size and prob arguments are given as vectors of length 1, the `*sinib` function reduces to `*binom` functions:

```
# Binomial:
dbinom(x = 1, size = 2, prob = 0.5)
[1] 0.5

# Sum of binomials:
library(sinib)
dsinib(x = 1, size = 2, prob = 0.5)
[1] 0.5
```

The next section shows a few examples to illustrate the usage of **sinib**.

Example usage of sinib

This section shows a few examples to illustrate the usage of **sinib**.

Sum of two binomials

We use two examples to illustrate the use of this package, starting from the simplest case of two binomial random variables with the same mean but different sizes: $X \sim Bin(n, p)$ and $Y \sim Bin(m, p)$. The distribution of $S = X + Y$ has an analytical solution, $S \sim Bin(m + n, p)$. We can therefore use different combinations of (m, n, p) to assess the accuracy of the saddlepoint approximation to the cumulative density function. We use $m, n = \{10, 100, 1000\}$ and $p = \{0.1, 0.5, 0.9\}$ to assess the approximation. The ranges of m and n are chosen to be large and the value of p are chosen to represent both boundaries.

```
library(foreach)
library(data.table)
library(cowplot)
library(sinib)

# Gaussian approximator:
p_norm_app = function(q,size,prob){
        mu = sum(size*prob)
        sigma = sqrt(sum(size*prob*(1-prob)))
        pnorm(q, mean = mu, sd = sigma)
}

# Comparison of CDF between truth and approximation:
data=foreach(m=c(10,100,1000),.combine='rbind')%do%{
        foreach(n=c(10,100,1000),.combine='rbind')%do%{
                foreach(p=c(0.1, 0.5, 0.9),.combine='rbind')%do%{
                        a=pbinom(q=0:(m+n),size=(m+n),prob = p)
                        b=psinib(q=0:(m+n),size=c(m,n),prob=c(p,p))
                        c=p_norm_app(q=0:(m+n),size=c(m,n),prob = c(p,p))
                        data.table(s=seq_along(a),truth=a,saddle=b,norm=c,m=m,n=n,p=p)
                }
        }
}
data[,m:=paste0('m = ',m)]
data[,p:=paste0('p = ',p)]
data = melt(data,measure.vars = c('saddle','norm'))

ggplot(data[n==10],aes(x=truth,y=value,color=p,linetype=variable))+
        geom_line()+
        facet_grid(p~m)+
        theme_bw()+
        scale_color_discrete(name='prob',guide = 'none')+
        xlab('Truth')+
        ylab('Approximation')+
        scale_linetype_discrete(name = '', breaks = c('saddle','norm'),
            labels = c('Saddlepoint','Gassian')) +
        theme(legend.position = 'top')
```

Figure 1 shows that the saddlepoint approximations are close to the ground truths across a range of parameters. For parsimony, this figure only shows the case of $n = 10$. In comparison with the saddlepoint method, the Gaussian method (dashed lines) provides a relatively poor approximation. We can further examine the accuracy by looking at the differences between the approximations and the ground truth.

```
data=foreach(m=c(100),.combine='rbind')%do%{
        foreach(n=c(100),.combine='rbind')%do%{
                foreach(p=c(0.1, 0.5, 0.9),.combine='rbind')%do%{
                        a=pbinom(q=0:(m+n),size=(m+n),prob = p)
                        b=psinib(q=0:(m+n),size=as.integer(c(m,n)),prob=c(p,p))
                        c=p_norm_app(q=0:(m+n),size=c(m,n),prob = c(p,p))
```

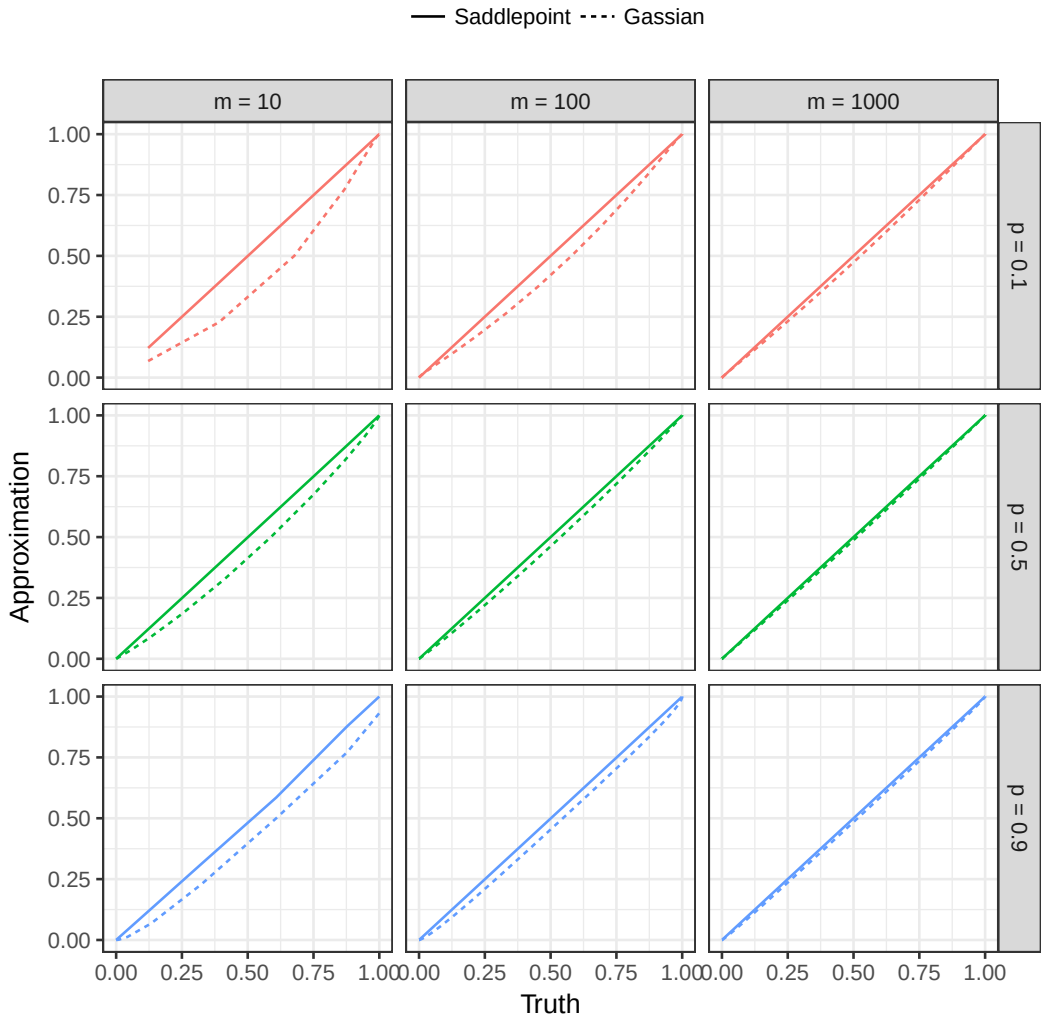


Figure 1: Comparison of CDF between truth and approximation for $n = 10$.

```
                    data.table(s=seq_along(a),truth=a,saddle=b,norm=c,m=m,n=n,p=p)
                }
        }
}
data = melt(data,measure.vars = c('saddle','norm'),value.name = 'approx',
        variable.name = 'Method')
data[,`Relative error` := (truth-approx)/truth]
data[,Error := (truth-approx)]
data[,p:=paste0('p = ',p)]
data = melt(data,measure.vars = c('Relative error','Error'),value.name = 'error',
        variable.name = 'type')

ggplot(data[Method == 'saddle'],aes(x=s,y=error,color=p))+
        geom_point(alpha=0.5)+theme_bw()+
        facet_wrap(type~p,scales='free')+
        xlab('Quantile')+
        ylab(expression(
            'Truth - Approximation'~~~~~~~~~~~~~~~~frac(Truth - Approximation,Truth)))+
        scale_color_discrete(guide='none') +
        geom_vline(xintercept=200*0.5,color='green',linetype='longdash')+
        geom_vline(xintercept=200*0.1,color='red',linetype='longdash')+
        geom_vline(xintercept=200*0.9,color='blue',linetype='longdash')
```

Figure 2 shows the difference between the truth and the approximation for $m = n = 100$. The dashed lines indicate the mean of each random variable. The approximations perform well overall.

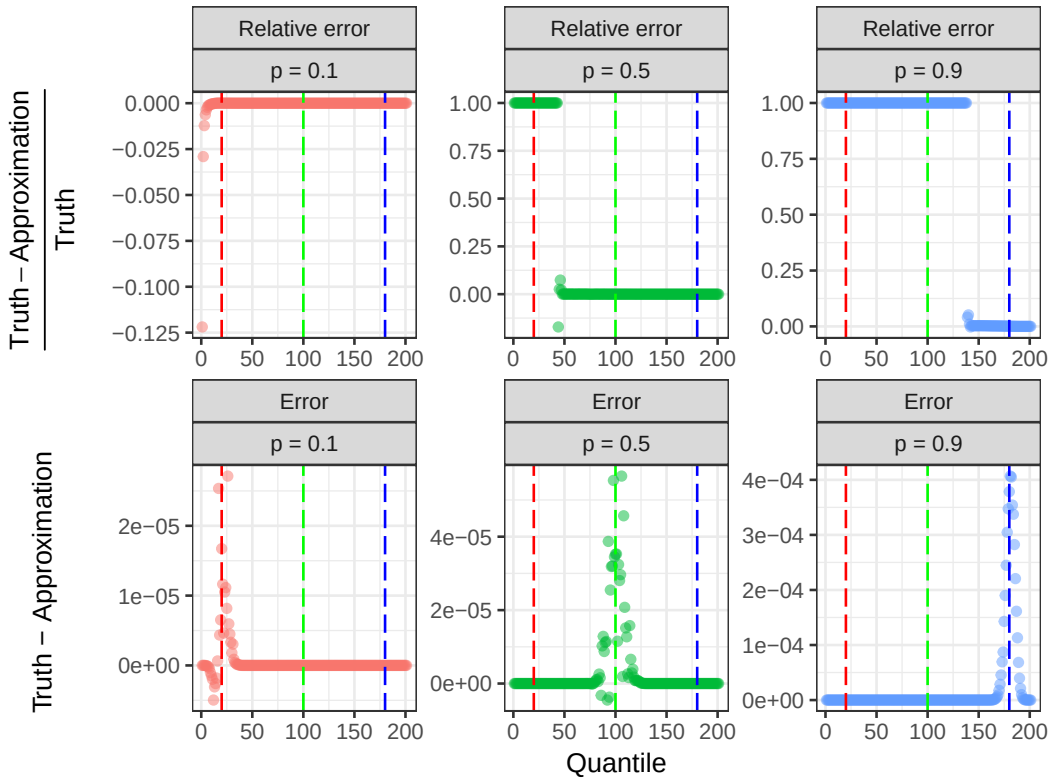


Figure 2: Difference in CDF between the ground truth and the approximation

The largest difference occurs around the mean, which is approximately 4e-4. It is worthwhile to mention that the errors are small for quantiles away from the mean because the the true probabilities are close to zero and one in the tails. To explore the tail behavior, we examine the relative error defined as $\frac{truth-approximation}{truth}$. The relative errors are large for quantiles between zero and the mean because the true probabilities in this interval are close to zero and the saddlepoint approximation returns zero. The relative errors are small for quantiles near and greater than the mean, indicating that the saddlepoint method provides a good approximation in this interval. As a baseline, we calculated the error and relative error derived from the Gaussian approximation (Figure 3). The largest absolute deviation approaches 0.06, two orders of magnitude greater than the deviation obtained from the saddlepoint approximation.

```
# Comparison of PDF between truth and approximation:
d_norm_app = function(x,size,prob){
        mu = sum(size*prob)
        sigma = sqrt(sum(size*prob*(1-prob)))
        dnorm(x, mean = mu, sd = sigma)
}

data=foreach(m=c(10,100,1000),.combine='rbind')%do%{
        foreach(n=c(10,100,1000),.combine='rbind')%do%{
                foreach(p=c(0.1, 0.5, 0.9),.combine='rbind')%do%{
                        a=dbinom(x=0:(m+n),size=(m+n),prob = p)
                        b=dsinib(x=0:(m+n),size=as.integer(c(m,n)),prob=c(p,p))
                        c=d_norm_app(x=0:(m+n),size=c(m,n),prob=c(p,p))
                        data.table(s=seq_along(a),truth=a,saddle=b,norm=c,m=m,n=n,p=p)
                }
        }
}

data[,m:=paste0('m = ',m)]
data[,p:=paste0('p = ',p)]
data = melt(data,measure.vars = c('saddle','norm'))
```

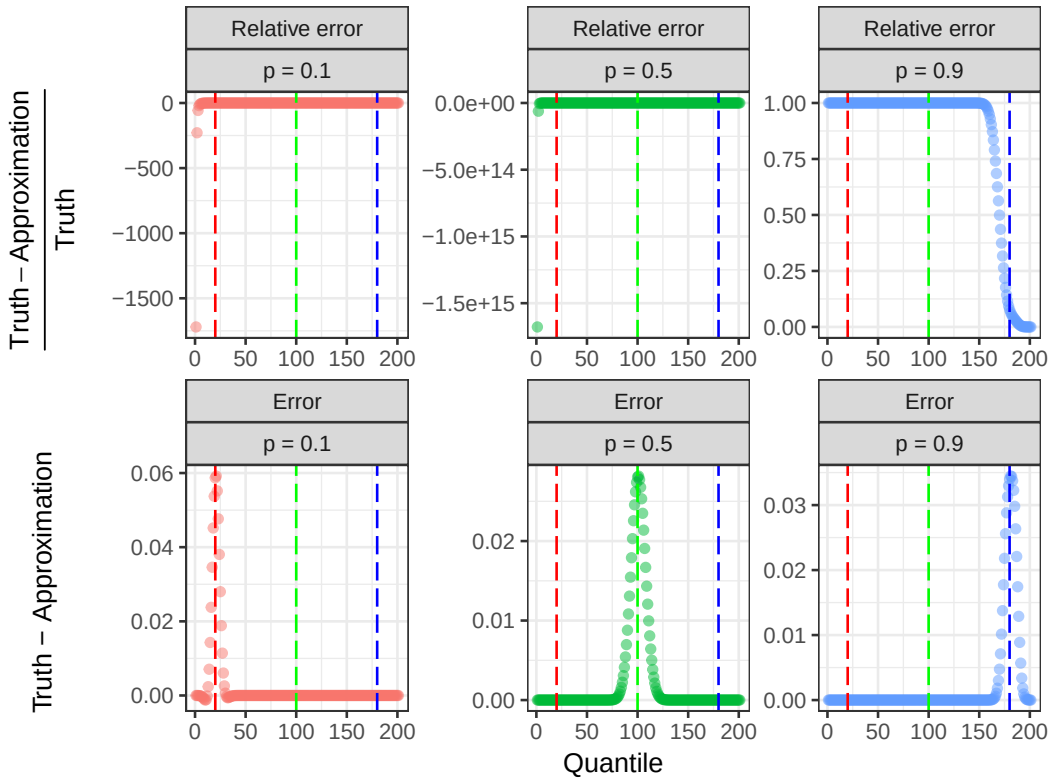


Figure 3: Difference in CDF between the ground truth and Gaussian approximation.

```
ggplot(data[n==10],aes(x=truth,y=value,color=p,linetype=variable))+
    geom_line()+
    facet_wrap(m~p,scales='free')+
    theme_bw()+
    scale_color_discrete(guide = 'none')+
    xlab('Truth')+
    ylab('Approximation') +
    scale_linetype_discrete(name = '', breaks = c('saddle','norm'),
            labels = c('Saddlepoint','Gassian')) +
    theme(legend.position = 'top')
```

We next examine the approximation for the probability density function. Figure 4 shows that the saddlepoint approximation is very close to the ground truth, whereas the Gaussian approximation is farther away. It is worthwhile to mention that the Gaussian method provides a good approximation for $p = 0.5$ because the distribution is symmetrical. Furthermore, we examine the difference between the truth and the approximations. One example for $m = n = 100$ is shown in Figure 5. As before, the saddlepoint approximation degrades around the mean, but the largest deviation is less than 4e-7. As a baseline, we calculated the difference between the true PDF and the Gaussian approximation in Figure 6. The largest deviation from the Gaussian approximation is 0.004, or four orders of magnitude greater than that from the saddlepoint approximation.

```
data=foreach(m=c(100),.combine='rbind')%do%{
        foreach(n=c(100),.combine='rbind')%do%{
                foreach(p=c(0.1, 0.5, 0.9),.combine='rbind')%do%{
                        a=dbinom(x=0:(m+n),size=(m+n),prob = p)
                        b=dsinib(x=0:(m+n),size=as.integer(c(m,n)),prob=c(p,p))
                        c=d_norm_app(x=0:(m+n),size=c(m,n),prob = c(p,p))
                        data.table(s=seq_along(a),truth=a,saddle=b,norm=c,m=m,n=n,p=p)
                }
        }
}
data = melt(data,measure.vars = c('saddle','norm'),value.name = 'approx',
        variable.name = 'Method')
data[,`Relative error` := (truth-approx)/truth]
```

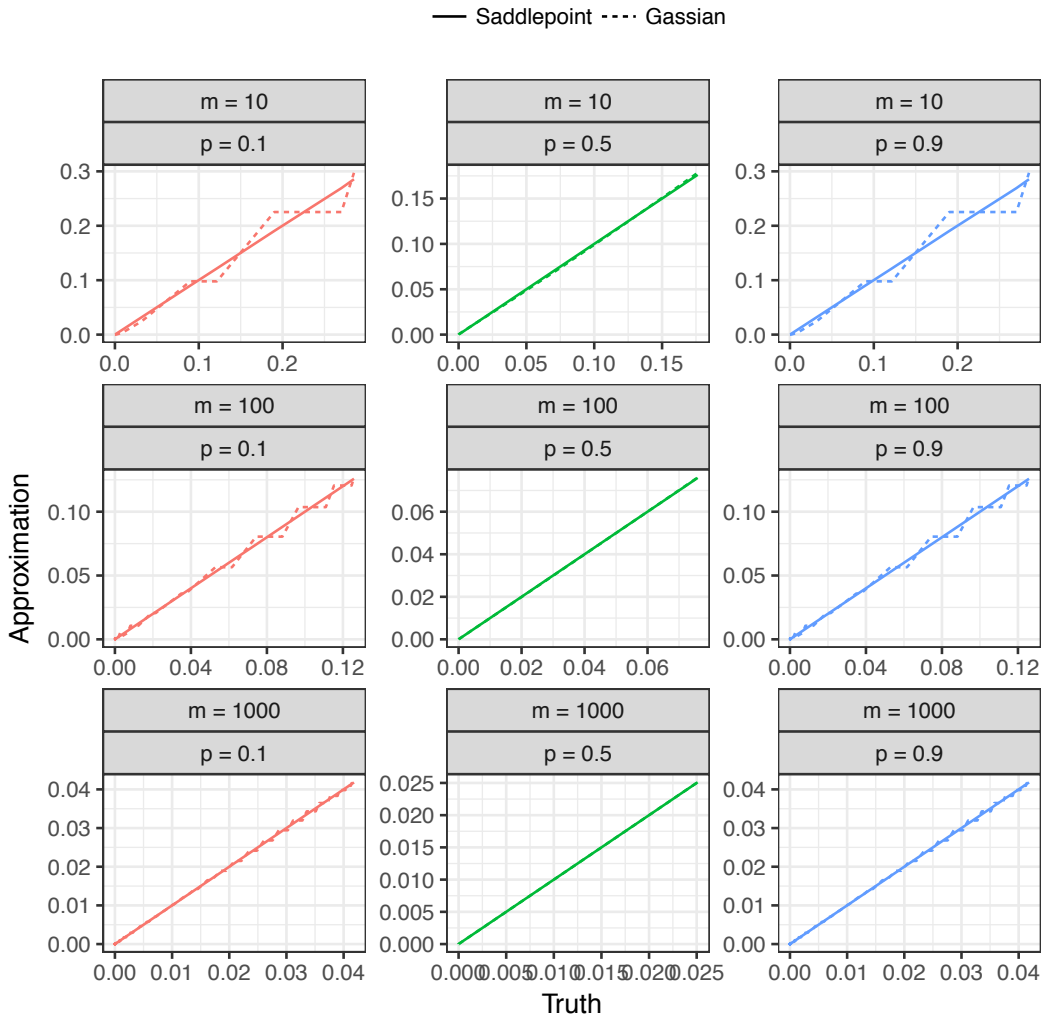


Figure 4: Comparison of PDF between truth and approximation for $n = 10$.

```
data[,Error := (truth-approx)]
data[,p:=paste0('p = ',p)]
data = melt(data,measure.vars = c('Relative error','Error'),
        value.name = 'error', variable.name = 'type')

ggplot(data[Method == 'saddle'],aes(x=s,y=error,color=p))+
        geom_point(alpha=0.5)+theme_bw()+
        facet_wrap(type~p,scales='free')+
        xlab('Quantile')+
        ylab(expression(
            'Truth - Approximation'~~~~~~~~~~~~~~~~frac(Truth - Approximation,Truth)))+
        scale_color_discrete(guide='none') +
        geom_vline(xintercept=200*0.5,color='green',linetype='longdash')+
        geom_vline(xintercept=200*0.1,color='red',linetype='longdash')+
        geom_vline(xintercept=200*0.9,color='blue',linetype='longdash')
```

Healthcare monitoring

In the second example, we used a health system monitoring dataset by Benneyan and Taşeli (2010). To improve compliance with medical devices, healthcare organizations often monitor bundle reliability statistics, each representing a percentage of patient compliance. Suppose n_i and p_i represent the number of patients and percentage of compliant patients for element i in the bundle, and n and p take the following values:

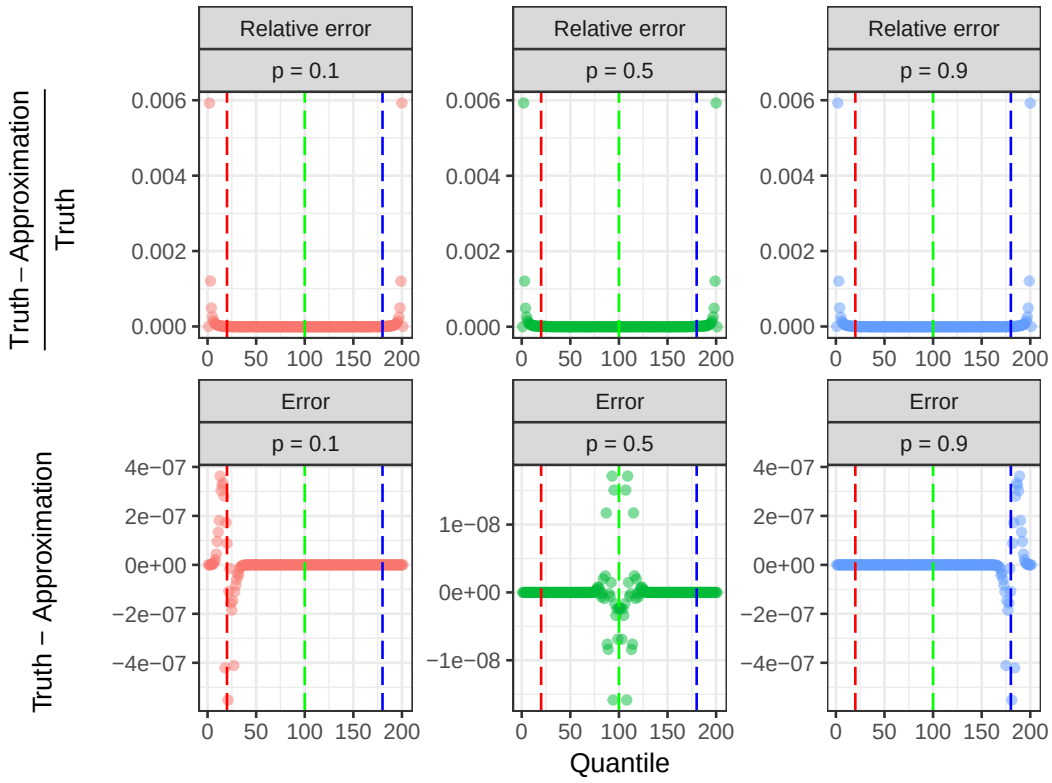


Figure 5: Difference in PDF between truth and the saddlepoint approximation.

```
size=as.integer(c(12, 14, 4, 2, 20, 17, 11, 1, 8, 11))
prob=c(0.074, 0.039, 0.095, 0.039, 0.053, 0.043, 0.067, 0.018, 0.099, 0.045)
```

Since it is difficult to find an analytical solution for the density, we estimated the density with simulation (1e8 trials) and treated it as the ground truth. We then compared simulations with 1e3, 1e4, 1e5, and 1e6 trials, and the saddlepoint approximation to the ground truth. (Note that running simulation will take several minutes.)

```
# Sinib:
approx=dsinib(0:sum(size),size,prob)
approx=data.frame(s=0:sum(size),pdf=approx,type='saddlepoint')

# Gauss:
gauss_approx = d_norm_app(0:sum(size),size,prob)
gauss_approx = data.frame(s=0:sum(size),pdf=gauss_approx,type='gauss')

# Simulation:
data=foreach(n_sim=10^c(3:6,8),.combine='rbind')%do%{
        ptm=proc.time()
        n_binom=length(prob)
        set.seed(42)
        mat=matrix(rbinom(n_sim*n_binom,size,prob),nrow=n_binom,ncol=n_sim)

        S=colSums(mat)
        sim=sapply(X = 0:sum(size), FUN = function(x) {sum(S==x)/length(S)})
        data.table(s=0:sum(size),pdf=sim,type=n_sim)

}

data=rbind(data,gauss_approx,approx)
truth=data[type=='1e+08',]

merged=merge(truth[,list(s,pdf)],data,by='s',suffixes=c('_truth','_approx'))
merged=merged[type!='1e+08',]
```

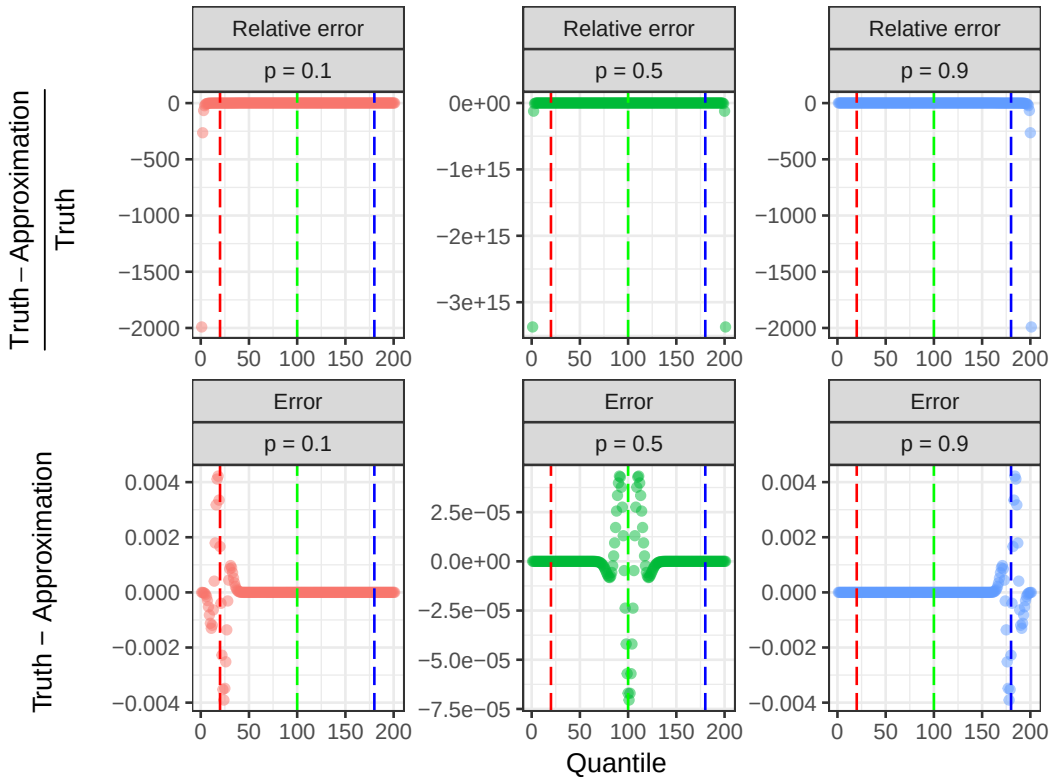


Figure 6: Difference in PDF between truth and the Gaussian approximation.

```
ggplot(merged,aes(pdf_truth,pdf_approx))+
        geom_point()+
        facet_grid(~type)+
        geom_abline(intercept=0,slope=1)+
        theme_bw()+
        xlab('Truth')+
        ylab('Approximation')
```

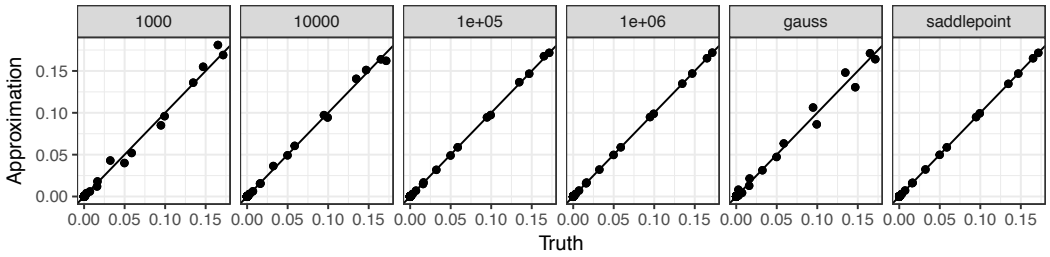


Figure 7: Comparison of PDF between truth and approximation.

Figure 7 shows that the simulation with 1e6 trials and the saddlepoint approximation are indistinguishable from the ground truth, while the Gaussian method and estimates with fewer simulations show clear deviations from the truth. To further examine the magnitude of deviation, we plot the difference in PDF between the truth and the approximation:

```
merged[,Error:=pdf_truth-pdf_approx]
merged[,`Relative Error`:=(pdf_truth-pdf_approx)/pdf_truth]
merged = melt(merged,measure.vars = c('Error','Relative Error'),variable.name = 'error_type',
        value.name = 'error')

p2=ggplot(merged,aes(s,error))+
        geom_point()+
```

```
facet_grid(error_type~type,scales = 'free_y')+
theme_bw()+
xlab('Outcome')+
ylab('Truth-Approx')+
xlim(0,20) +
ylab(expression(frac(Truth - Approximation,Truth)~~~~~~~'Truth - Approximation'))
```

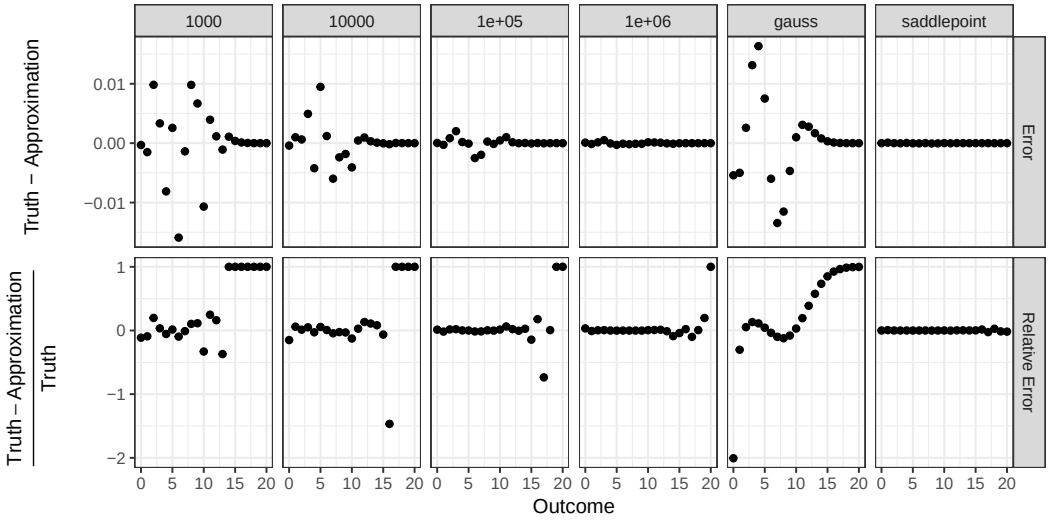


Figure 8: Error and relative error between truth and approximation.

Figure 8 shows that that the saddlepoint method and the simulation with 1e6 draws both provide good approximations, while the Gaussian approximation and simulations of smaller sizes show clear deviations. We also note that the saddlepoint approximation is 5 times faster than the simulation of 1e6 trials.

```
ptm=proc.time()
n_binom=length(prob)
mat=matrix(rbinom(n_sim*n_binom,size,prob),nrow=n_binom,ncol=n_sim)
S=colSums(mat)
sim=sapply(X = 0:sum(size), FUN = function(x) {sum(S==x)/length(S)})
proc.time()-ptm
#    user  system elapsed
#   1.008   0.153   1.173

ptm=proc.time()
approx=dsinib(0:sum(size),size,prob)
proc.time()-ptm
#   user  system elapsed
#  0.025   0.215   0.239
```

Conclusion and future direction

In this paper, we presented an implementation of the saddlepoint method to approximate the distribution of the sum of independent and non-identical binomials. We assessed the accuracy of the method by comparing it with first, the analytical solution in the simple case of two binomials, and second, the simulated ground truth on a real-world dataset in healthcare monitoring. These assessments suggest that the saddlepoint method generally provides an approximation superior to simulation in terms of both speed and accuracy, and outperforms the Gaussian approximation in terms of accuracy. Overall, the **sinib** package addresses the gap between the theory and implementation on the approximation of sum of independent and non-identical binomial random variables.

In the future, we aim to explore other approximation methods such as the Kolmogorov approximation and the Pearson curve approximation described by Butler and Stephens (2016).

Bibliography

M. Akahira and K. Takahashi. A Higher-Order Large-Deviation Approximation for the Discrete Distributions. *Journal of the Japan Statistical Society*, 31(2):257–267, 2001. URL https://doi.org/10.1080/03610929908832322. [p473]

J. Arthur Woodward and C. G. S. Palmer. On the Exact Convolution of Discrete Random Variables. *Applied Mathematics and Computation*, 83(1):69–77, 1997. URL https://doi.org/10.1016/S0096-3003(96)00047-1. [p473]

J. C. Benneyan and A. Taşeli. Exact and Approximate Probability Distributions of Evidence-Based Bundle Composite Compliance Measures. *Health Care Management Science*, 13(3):193–209, 2010. URL https://doi.org/10.1007/s10729-009-9123-x. [p472, 479]

K. Butler and M. A. Stephens. The Distribution of a Sum of Independent Binomial Random Variables. *Methodology and Computing in Applied Probability*, 19(2):557–571, 2016. URL https://doi.org/10.1007/s11009-016-9533-4. [p473, 482]

H. E. Daniels. Saddlepoint Approximations in Statistics. *The Annals of Mathematical Statistics*, 25(4):631–650, 1954. URL http://www.jstor.org/stable/2236650. [p473]

H. E. Daniels. Tail Probability Approximations. *International Statistical Review / Revue Internationale de Statistique*, 55(1):37–48, 1987. URL http://www.jstor.org/stable/1403269. [p473, 474]

R. Eisinga, M. Te Grotenhuis, and B. Pelzer. Saddlepoint Approximations for the Sum of Independent Non-Identically Distributed Binomial Random Variables. *Statistica Neerlandica*, 67(2):190–201, 2013. URL https://doi.org/10.1111/stan.12002. [p472, 473]

N. L. Johnson, A. W. Kemp, and S. Kotz. *Univariate Discrete Distributions*. Johnson/Univariate Discrete Distributions. John Wiley & Sons, Hoboken, NJ, USA, 2005. URL https://doi.org/10.1002/0471715816. [p472]

J. K. Jolayemi. A Unified Approximation Scheme for the Convolution of Independent Binomial Variables. *Applied Mathematics and Computation*, 49(2-3):269–297, 1992. URL https://doi.org/10.1016/0096-3003(92)90030-5. [p472]

S. Kotz and N. L. Johnson. Effects of False and Incomplete Identification of Defective Items on the Reliability of Acceptance Sampling. *Operations Research*, 32(3):575–583, 1984. URL https://pubsonline.informs.org/doi/abs/10.1287/opre.32.3.575. [p472]

R. Lugannani and S. Rice. Saddle Point Approximation for the Distribution of the Sum of Independent Random Variables. *Advances in Applied Probability*, 12(2):475, 1980. URL http://www.jstor.org/stable/1426607. [p473]

E. M. Schmidt, J. Zhang, W. Zhou, J. Chen, K. L. Mohlke, Y. E. Chen, and C. J. Willer. GREGOR: Evaluating Global Enrichment of Trait-Associated Variants in Epigenomic Features Using a Systematic, Data-Driven Approach. *Bioinformatics*, 31(16):2601–2606, 2015. URL https://doi.org/10.1093/bioinformatics/btv201. [p472]

H. Yili. On Computing the Distribution Function for the Sum of Independent and Non-Identical Random Indicators . URL https://pdfs.semanticscholar.org/fe97/c1358ec01c86cb8bbc4574fa064748f37e94.pdf. [p473]

Boxiang Liu
Stanford University
300 Pasteur Drive, Stanford, CA
United States
ORCID: 0000-0002-2595-4463
bliu2@stanford.edu

Thomas Quertermous
Stanford University
300 Pasteur Drive, Stanford, CA
United States
tomq1@stanford.edu

cchs: An R Package for Stratified Case-Cohort Studies

by Edmund Jones

Abstract The **cchs** package contains a function, also called `cchs`, for analyzing data from a stratified case-cohort study, as used in epidemiology. For data from this type of study, `cchs` calculates Estimator III of Borgan et al. (2000), which is a score-unbiased estimator for the regression coefficients in the Cox proportional hazards model. From the user's point of view, the function is similar to `coxph` (in the **survival** package) and other widely used model-fitting functions. Convenient software has not previously been available for Estimator III since it is complicated to calculate. SAS and S-Plus code-fragments for the calculation have been published, but `cchs` is easier to use and more efficient in terms of time and memory, and can cope with much larger datasets. It also avoids several minor approximations and simplifications.

Introduction

One common type of study in epidemiology is the cohort study, in which participants are recruited and followed over time. Participants who experience a certain type of event (e.g., a heart attack or the diagnosis of a disease) are called cases, and participants who do not are called non-cases or controls. A related type of study is the case-cohort study, which is nested within a cohort study. In a case-cohort study, some of the covariates are only measured for the cases and the subcohort—the subcohort is a randomly selected subset of the cohort. These covariates are not measured for the other participants. A case-cohort study can often estimate effect-sizes almost as accurately as the corresponding cohort study, but at much lower cost since measuring all the covariates on all the participants is expensive.

The case-cohort dataset contains only the cases and the subcohort, and is usually analyzed with a Cox proportional hazards regression model (Cox, 1972). A common way to fit the Cox model is to use the estimator of Prentice (1986), who proposed the case-cohort design. More advanced kinds of analysis sometimes use information on the full cohort, such as the values of those covariates that were measured for all the participants.

In a stratified case-cohort study, the selection of the subcohort is stratified. The cohort is divided into strata and each stratum is assigned a sampling fraction, which is the number of participants in that stratum who will be in the subcohort divided by the total number of participants in the stratum. The purpose of the stratification is usually to improve the efficiency of the estimator. To fit a Cox model to data from a stratified case-cohort study, one possible estimator is the time-fixed version of Estimator III of Borgan et al. (2000). This estimator is optimal in the sense that it is score-unbiased; this criterion is explained in the section "Comparison with other estimators". In this article, "Estimator III" means the time-fixed version of Borgan et al.'s Estimator III, unless otherwise stated.

The subject of this article is the **cchs** package (version 0.4.0; Jones, 2017), which has made Estimator III available in a convenient form. The package consists of a single function, also called `cchs`, which calculates the estimator, and an example dataset, called `cchsData`.

Previous software for calculating Estimator III includes SAS code (SAS Institute Inc., 1999) that appears in Langholz and Jiao (2007a) and S-Plus code (Tibco Software Inc., 2007) in the appendix of Cologne et al. (2012). S-Plus is no longer available for purchase but is very similar to R; Cologne et al. (2012)'s code works in R if one function, `match.data.frame`, is imported from S-Plus.

The previously published SAS and S-Plus code-fragments both analyze specific datasets with specific models. They could be rewritten to work more generally, but even then with some datasets they would give slightly inaccurate results, and with others they would use far too much computational time or memory to be usable, as explained later in the article. Langholz and Jiao wrote more general SAS code (available at Langholz and Jiao, 2007b), but this still has the problems with inaccurate results and computational resources and is less easy to use than `cchs`. Cologne et al. (2012) state that Estimator III can be calculated by an application called Epicure, which is now sold by Risk Sciences International. The estimator was also calculated by Borgan et al. themselves, but their code is not publicly available.

The `cch` function in the **survival** package (Therneau, 2017; Therneau and Grambsch, 2000) can calculate several estimators for case-cohort data, including the estimator of Prentice (1986) and Estimators I and II of Borgan et al. (2000). The functionality of `cchs` could be added to `cch` as a single additional option. But Estimator III is complicated to calculate and `cchs` has numerous features that `cch` lacks. Other software for stratified case-cohort studies includes the **survey** and **NestedCohort** packages, which use inverse probability weights (Lumley, 2004, 2017a; Mark and Katki, 2006; Katki and Mark, 2008), and the code for Estimator II in Samuelsen et al. (2007).

The next five sections discuss the model, the estimator, and how to use `cchs`. The function is easy to use for anyone who is accustomed to the syntax of model-fitting functions in R such as `lm` or `coxph`. The subsequent sections discuss how `cchs` works internally compared to the previously published SAS and S-Plus code. The previously published code makes several approximations and is unable to cope with large datasets, but `cchs` avoids these drawbacks by manipulating the data in an economical way, leading to huge savings in computational time and memory.

Parts of `cchs` are based on code from `coxph`, `cch`, and the appendix of Cologne et al. (2012), and the mathematical formulas and some of the computational methods are from Therneau and Li (1999), Borgan et al. (2000), Langholz and Jiao (2007a), and Cologne et al. (2012), but a major part of the computational methods is original, as discussed in the section "How cchs works".

The idea of creating `cchs` came from work on EPIC-InterAct (InterAct Consortium, 2011) and EPIC-CVD (Danesh et al., 2007), two stratified case-cohort studies that are nested within the cohort from EPIC, the European Prospective Investigation of Cancer and Nutrition (Bingham and Riboli, 2004). EPIC-InterAct is a study of incident type 2 diabetes with data from 29 centers in eight European countries, where the subcohort was stratified by center or country. EPIC-CVD is a study of cardiovascular disease that uses most of the same centers as EPIC-InterAct and the same subcohort in those centers. In Jones et al. (2015), data from EPIC-InterAct was used to compare five different models for data from stratified case-cohort studies, and a pre-release version of `cchs` was used to fit two of those models.

The model and the estimator

The Cox model for survival-time data is commonly defined by this equation:

$$h_i(t) = h_0(t) \exp(\beta^\top z_i),$$

where

h_i is the hazard function for participant i,

t is the time to the event,

h_0 is the baseline hazard function,

β is the regression coefficients (log hazard ratios), and

z_i is the covariates for participant i.

If all the event-times are distinct, then the partial likelihood is

$$\prod_j \frac{\exp(\beta^\top z_{i_j})}{\sum_k Y_k(t_j) \exp(\beta^\top z_k)},$$

where

t_j is the jth event-time,

i_j is the participant whose event happened at time t_j,

$$Y_k(t_j) = \begin{cases} 1 & \text{if } t_j \in (t_{0k}, t_{1k}] \\ 0 & \text{otherwise} \end{cases},$$

t_{0k} is the entry-time for participant k (often zero for all participants), and

t_{1k} is the exit-time for participant k (i.e. the time at which they had the event or were censored).

The product is over all the cases and the sum is over all the participants. The time-interval $(t_{0k}, t_{1k}]$ is participant k's "at-risk time" and $Y_k(t_j)$ is an indicator variable for whether participant k is at risk at time t_j.

For case-cohort data, z_i is only known for the cases and subcohort members, so the partial likelihood cannot be evaluated and it is impossible to estimate the regression coefficients using the method of maximum likelihood. Instead, the coefficients are estimated by maximizing pseudo-likelihoods, which are approximations of the partial likelihood. The pseudo-likelihood for Estimator III is

$$\prod_j \frac{\alpha_{s(i_j)}^{-1} \exp(\beta^\top z_{i_j})}{\sum_{k \in R(t_j)} Y_k(t_j) \alpha_{s(k)}^{-1} \exp(\beta^\top z_k)},$$

where

$s(i)$ is the stratum that contains participant i,

α_s is the sampling fraction for stratum s (the proportion of stratum s that appears in the subcohort),

$$R(t_j) = \begin{cases} C & \text{if } i_j \in C \\ C \cup \{i_j\} \setminus \{J_{s(i_j)}\} & \text{if } i_j \notin C' \end{cases}$$

C is the subcohort, and

J_s is a participant randomly selected from $C \cap s$ (where s is a stratum).

Borgan et al. (2000) called this a "swapper" method because of the way $R(t_j)$ is defined for $i_j \notin C$. As shown above, $J_{s(i_j)}$ is removed from C and i_j is added. If $J_{s(i_j)}$ is not at risk at time t_j, then its removal from $R(t_j)$ makes no difference to the pseudo-likelihood since $Y_{J_{s(i_j)}}(t_j) = 0$. It is recommended to use an asymptotic covariance estimator rather than a robust one (Jiao, 2001); see the section "The calculation of the covariance matrix".

Langholz and Jiao (2007a) discuss two situations in which a case-cohort study might be stratified, which they call "exposure stratified" and "confounder stratified." In the exposure stratified situation, the strata are defined by levels of one or more covariates, and the purpose of the stratification is to increase the efficiency of the estimator. In the confounder stratified situation, the strata are defined by a covariate that is believed to confound the relation between the exposure of interest and the event-time (e.g., the center in a multi-center study). For an exposure stratified study, it is usual to fit a single unstratified Cox model to the data, and this is is the situation for which Estimator III was designed. For a confounder stratified study, it is more natural to fit a stratified Cox model—in which each stratum s has its own baseline hazard $h_{0s}(t)$ and $h_i(t) = h_{0s(i)}(t)\exp(\beta^\top z_i)$—and for this, the estimator of Prentice (1986) should be used; but it is still possible to fit an unstratified model using Estimator III.

Comparison with other estimators

Borgan et al. (2000) described three estimators for the Cox model with data from a stratified case-cohort study. These have been investigated in several simulation studies: Borgan et al. used cohorts of size 1000 with subcohorts of size 100, and found that there was little to choose between Estimators II and III, but both performed better than Estimator I; Cologne et al. (2012) made similar findings but reported that Estimator III did slightly better than Estimator II when the sampling fraction was very small; and Samuelsen et al. (2007) found that Estimator II performed well.

Of the three estimators, only Estimator III is score-unbiased. This is one of the main desirable criteria for estimators for case-cohort data. It means that the expectation of the pseudo-score (the derivative of the pseudo-likelihood) is zero when the coefficients take their true values (Godambe, 1960, 1976; Lindsay, 1982; Severini, 2011; Cologne et al., 2012). On the other hand, Estimator II has smaller asymptotic variance (this was hinted at in Borgan et al., 2000, stated in Samuelsen et al., 2007, and explained most clearly in Samuelsen et al., 2006).

Recall that "Estimator III" refers to the time-fixed version of that estimator, as calculated by `cchs`. Borgan et al. also described time-dependent versions of the three estimators, which have smaller asymptotic variance (Kulich and Lin, 2004). The time-dependent version of Estimator III is score-unbiased (Borgan et al., 2000), but it is even more complicated to calculate than the time-fixed version.

Estimator II can be improved using covariate data on the whole cohort, if that is available (Breslow et al., 2009a,b; Yan et al., 2017). The estimator of Prentice (1986) can also be adapted to work with stratified case-cohort studies. Its pseudo-likelihood is

$$\prod_j \frac{\exp(\beta^\top z_{i_j})}{\sum_{k \in C \cup \{i_j\}} Y_k(t_j)\exp(\beta^\top z_k)}.$$

This estimator is only score-unbiased if for $i_j \notin C$ the C in the above expression is replaced by $(C \cap s(i_j))$. This is less efficient than Estimator III since the denominator includes fewer participants.

For stratified case-cohort data it is also possible to use general methods based on inverse probability weights. These methods are suitable for a wide range of complex sampling schemes (Mark and Katki, 2006; Lumley, 2017b), but no claim is made that the estimators are score-unbiased. This approach has been used by Gray (2009), Liu et al. (2012), and Payne et al. (2016). In other research, Kulich and Lin (2004) presented a framework that covered many case-cohort estimators and the possibility of stratified subcohort-selection, and Zeng and Lin (2007) and Kim et al. (2013) described wide classes of sampling schemes and regression models for survival analysis, of which the stratified case-cohort design and the Cox model are special cases.

How to use cchs

From the user's point of view, cchs is reasonably similar to coxph. The most important arguments are:

formula, which specifies the model (the left-hand side must be a "Surv" object),

data, a data frame or environment that contains the variables in the formula,

inSubcohort, an indicator for whether each participant is in the subcohort,

stratum, the stratum variable,

samplingFractions, the sampling fraction for each stratum,

cohortStratumSizes, the size of each stratum in the full cohort, and

precision, discussed below.

If the data is supplied as a data frame, it is assumed to be in the usual form where each row corresponds to one observation (i.e., one participant in the case-cohort study) and each column corresponds to a variable. The inSubcohort and stratum arguments can either be columns or elements of data or separate objects that are named in the call to cchs.

One of samplingFractions or cohortStratumSizes must be specified, but not both. If the latter is specified, then cchs uses it to calculate the former—for each stratum, it divides the size of the subcohort by the size of the cohort, which is given by cohortStratumSizes. If the sampling fractions are not stored as a column of data, then they can be passed to cchs as a named vector. For example, if the stratum variable takes the values "London" and "Paris" and the sampling fractions are 0.1 and 0.2 respectively, then this vector should be c(London = 0.1, Paris = 0.2).

The precision argument must be set if there are any tied event-times. Estimator III requires all event-times to be distinct, so cchs changes tied event-times by small amounts, as suggested by Langholz and Jiao (2007a), and it uses precision to work out suitable values. If the times are all integers, then precision should be set to 1. In datasets from EPIC-InterAct and EPIC-CVD, the times are recorded to the nearest day but stored as numbers of years, so precision should be set to 1 / 365.25.

Additional arguments are discussed in "Obscure aspects of the calculation".

An example analysis

cchsData is an artificial stratified case-cohort dataset that was created from the nwtco dataset in the **survival** package. The original nwtco data comes from two clinical trials run by the National Wilms Tumor Study Group in the United States (D'Angio et al., 1989; Green et al., 1998). Different artificial case-cohort datasets based on nwtco have been used by Kulich and Lin (2004), Breslow et al. (2009a), and Breslow et al. (2009b).

In cchsData, the event is relapse of Wilms tumor and the subcohort-selection is stratified by the result of a histological examination. The most important variables are time (the time to the event or censoring), isCase (the event-indicator), inSubcohort (the indicator for being in the subcohort), localHistol (the stratum variable), and sampFrac (the stratum-specific sampling fractions). The times are stored as numbers of days, and some of them are tied, so the precision argument of cchs should be set to 1 (or 1/2, 1/3, etc.).

The sampling fractions in cchsData are 5% and 20%. The subcohort is about half as big as the set of cases. In case-cohort studies in general, the sampling fraction is typically in the area of 5% but sometimes larger (Sharp et al., 2014), and the subcohort is usually larger than the set of cases (Juraschek et al., 2013; Lamb et al., 2013; Jones et al., 2015) but occasionally smaller (Huxley et al., 2013).

The following command uses Estimator III to fit a Cox model to cchsData. The covariates are age at diagnosis (ageAtDiagnosis) and stage of cancer at diagnosis (stage):

```
> cchs(Surv(time, isCase) ~ ageAtDiagnosis + stage, data = cchsData,
+ inSubcohort = inSubcohort, stratum = localHistol,
+ samplingFractions = sampFrac, precision = 1)
```

The cchs function returns an object of S3 class "cchs" that contains many of the same elements as a "coxph" object and several additional elements. When a "cchs" object is printed, information about the dataset and the changes to the tied event-times appears before the coefficients. The output from the above command is:

```
Call: cchs(formula = Surv(time, isCase) ~ stage + ageAtDiagnosis,
    data = cchsData, inSubcohort = inSubcohort, stratum = localHistol,
    samplingFractions = sampFrac, precision = 1)
Number of observations/rows: 785, of which
        subcohort non-cases: 214
        subcohort cases:     48
        non-subcohort cases: 523
179 of 571 discretized event-times were changed by up to 0.01 to deal
 with ties.
Number of strata for subcohort-selection: 2
Coefficients (HR = hazard ratio, CI = 95% confidence interval, SE = standard
error):
                     HR   CIlower  CIupper           p     logHR    SElogHR
ageAtDiagnosis 1.101685 1.029489 1.178944 0.005104173 0.09684087 0.03458127
stageII        1.397695 0.915168 2.134635 0.121219767 0.33482413 0.21606100
stageIII       1.794611 1.156081 2.785816 0.009150320 0.58478850 0.22436755
stageIV        2.367953 1.382691 4.055282 0.001686931 0.86202589 0.27449190
```

This output can be interpreted like output from any other function that fits the Cox model. For example, the hazard is estimated to be 79% higher for patients who were at stage III when diagnosed compared to patients who were at stage I, adjusting for age at diagnosis. The level for the confidence intervals in the output is determined by the `confidenceLevel` argument to `cchs`, whose default is 0.95.

If the output of `cchs` is stored as an object called `result`, then the coefficients are available as `result$coefficients` and the covariance matrix is available as `result$var`. The table at the bottom of the output is stored as `result$coeffsTable` for convenience. For example, this can be used to get the confidence intervals for the hazard ratio for `ageAtDiagnosis`:

```
> result <- cchs(Surv(time, isCase) ~ ageAtDiagnosis + stage, data = cchsData,
+ inSubcohort = inSubcohort, stratum = localHistol,
+ samplingFractions = sampFrac, precision = 1)
> result$coeffsTable["ageAtDiagnosis", c("CIlower", "CIupper")]
```

Other issues for the user

Calling `cchs` twice with the same arguments will not necessarily give exactly the same results, which is unusual for estimators that maximize a likelihood or pseudo-likelihood. This happens for two reasons: first, Estimator III involves randomness in the choice of $J_{s(i_j)}$; and second, if there are tied event-times, these are changed in a random way. To get exactly the same results, set the random seed (using `set.seed`) just before the call to `cchs`.

Another approach would be to call `cchs` multiple times, using the `replicate` command, then combine the results, perhaps by using Rubin's rules (Rubin, 1987; Schafer, 1999), which are normally used for multiple imputation. With this approach, for each term in the model the coefficient estimate is the mean of the separate estimates, and the variance is the mean of the separate variances plus a term to account for the between-replication variability. It might be worth expressly comparing the within-replication variability (the separate variance estimates) with the between-replication variability (the variability due to the randomness of the estimator). Figure 1 shows the point-estimates and confidence intervals that appear in the previous section, and the quartiles and ranges of the point-estimates produced by 1000 calls to `cchs` with the same arguments. Here, the between-replication variability is small, and very small compared to the within-replication variability.

In `coxph`, model formulas can contain certain special terms. Of these, `cchs` allows `offset` terms, but it does not allow `cluster` (for identifying correlated groups of rows), `strata` (for fitting a stratified Cox model), or `tt` (for time-varying covariates). If the data are from a stratified case-cohort study and it is desired to fit a stratified Cox model where the strata in the baseline hazard are the same as the strata for the subcohort-selection, then Prentice's estimator should be used (Langholz and Jiao, 2007a).

All error and warning messages produced by `cchs` are designed to be meaningful. This is achieved by checking the arguments and raising an error if any of them have illegal values, and by adding extra messages to any errors or warnings raised by `model.frame` or `coxph` when they are called internally by `cchs`. The addition of the extra messages results in the call stack, which can be viewed by typing `traceback()`, being longer and more complicated than usual. Therefore if `traceback()` is going to be used to diagnose problems, it is advisable to switch the extra messages off by setting `annotateErrors = FALSE`.

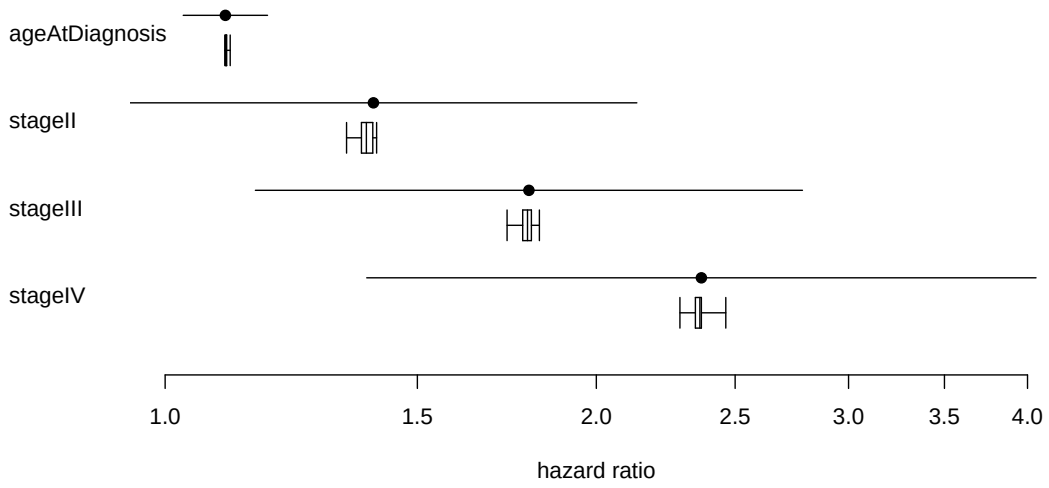


Figure 1: The dots and horizontal lines show the point-estimates and confidence intervals from the output in "An example analysis"; this is within-replication variability. The boxes and whiskers show the hazard ratios from 1000 replications of cchs, using the dataset and model in the same section; this is between-replication variability. The boxes show the quartiles and the whiskers show the full ranges.

How cchs works

The cchs function works differently from the previously published SAS and S-Plus code in several ways, and these novel aspects of its implementation affect the amount of computational time and memory that it uses and enable it to work with much bigger datasets. They also affect the numbers in the output. These issues are discussed here for cautious or curious users who want to know how cchs works or compare it with other software for Estimator III.

Functions that calculate estimators for the Cox model with case-cohort data invariably work by manipulating the data or the model-formula in various ways—for example, duplicating rows and changing entry-times, or appending weights to the rows and including the weight in the model-formula—and then passing the data and model-formula to a second model-fitting function (in R, coxph) whose original purpose is to fit the Cox model with the standard estimator. The manipulation is done in such a way that when the second function attempts to maximize the partial likelihood, what it actually maximizes is the pseudo-likelihood for the case-cohort estimator. The second function returns the estimates of the regression coefficients and their covariance matrix, and the top-level function usually then makes adjustments to the covariance matrix.

This method works because the standard partial likelihood and the various pseudo-likelihoods all have similar forms. They have the same number of multiplicative terms, one for each case, and the same values in the numerators (apart from multiplicative factors, which are easy to deal with). The only differences are the sets of participants who appear in the denominators and, for some pseudo-likelihoods, multiplicative factors in the denominators. These sets are dealt with by duplicating or excluding rows and changing entry- and exit-times, and any multiplicative factors are dealt with by using an offset term in the model-formula.

For Estimator III, the manipulations are rather complicated because of the swapping. The cchs function manipulates the data as follows:

1. Define a small number ϵ, which is smaller than any of the differences between consecutive event-times.
2. For each $i_j \notin C$, change the entry-time to $t_j - \epsilon$, where t_j is the exit-time.
3. For each stratum s:
   - 3.1 Choose a random participant J_s from $C \cap s$. Denote the entry-time for J_s by t_{0s} and the exit-time by t_{1s}.
   - 3.2 Let $U_s = \{t_j : i_j \in s, i_j \notin C, t_{0s} < t_j \leq t_{1s}\}$. Sort U_s into increasing order as $u_{s(1)}, \ldots, u_{s(|U_s|)}$.
   - 3.3 Replace J_s's row by $|U_s| + 1$ new rows with variables as follows.

     Row 1: entry-time t_{0s}, exit-time $u_{s(1)} - \epsilon$, event-indicator zero.

     Row 2: entry-time $u_{s(1)}$, exit-time $u_{s(2)} - \epsilon$, event-indicator zero.

...

Row $|U_s|$: entry-time $u_{s(|U_s|-1)}$, exit-time $u_{s(|U_s|)} - \epsilon$, event-indicator zero.

Row $|U_s| + 1$: entry-time $u_{s(|U_s|)}$, exit-time t_{1s}, event-indicator as in the original row.

These steps are best understood by referring to the earlier definition of $R(t_j)$ and bearing in mind that coxph takes a row's at-risk time to be the half-open interval, (entry-time, exit-time]. The purpose of step 1 is to choose ϵ to be sufficiently small that steps 2 and 3 do not have any unintended effects on other terms in the pseudo-likelihood. Step 2 has the effect of removing the non-subcohort cases from $R(t_j)$, except for i_j itself, which is not removed. Step 3 makes the further changes to $R(t_j)$ that are needed if $i_j \notin C$. Note that it deals with one J_s, not one i_j, at a time. Step 3.1 chooses J_s. Step 3.2 defines U_s to be the event-times that lie in the range of J_s's at-risk time and correspond to non-subcohort cases in s. Step 3.3 excises a short stretch of at-risk time from J_s at each time in U_s by splitting J_s's row into multiple rows with separate stretches of at-risk time.

The effect of step 3 is to remove $J_{s(i_j)}$ from $R(t_j)$ when $i_j \notin C$, as required by the definition of $R(t_j)$. For a non-subcohort case in s whose event-time is not in U_s, it makes no difference whether $J_{s(i_j)}$ is in $R(t_j)$ or not, because $Y_{J_{s(i_j)}}(t_j)$ is zero; so for such cases this manipulation is not necessary.

One other change to the data is needed. An extra column is created that contains $-\log \alpha_{s(k)}$ for participant k, and this is included as an offset term in the model-formula that cchs passes to coxph. This has the effect of inserting the $\alpha_{s(k)}^{-1}$ factors that are needed in the denominators.

The calculation of the covariance matrix

The cchs function uses an asymptotic covariance estimator, as recommended by Jiao (2001). Internally, after it manipulates the data and calls coxph, the final step is to adjust the covariance matrix. This is done by using the matrix of "dfbeta" residuals, D, produced by `residuals.coxph`; d_{ij} is the change in the estimate of regression coefficient j when row i is dropped from the data (or an approximation of that change). The covariance matrix is adjusted by adding $\sum_s m_s(1-\alpha_s)\operatorname{Cov} D_{C\cap s}$, where $m_s = |C \cap s|$, $D_{C\cap s}$ is the rows of D that correspond to $C \cap s$, and $\operatorname{Cov} D_{C\cap s}$ is the empirical covariance matrix $D_{C\cap s}^\top D_{C\cap s}/(m_s - 1)$.

This adjustment corresponds to Equation (17) in Borgan et al. (2000). The use of D is based on the formula for the adjustment to the covariance matrix with unstratified case-cohort data, which appears on page 102 of Therneau and Li (1999) and as Equation (5) in Langholz and Jiao (2007a). There is a small discrepancy between the formulas in these two publications. To explain this, let D_C be the rows of D that correspond to C and let $\alpha = m/n$ be the sampling fraction, where m is the size of the subcohort and n is the size of the cohort. According to Therneau and Li, the quantity to add to the covariance matrix is $(1-\alpha)D_C^\top D_C$, but according to Langholz and Jiao it is $\frac{m(n-m)}{n}\operatorname{Cov} D_C = (1-\alpha)\frac{m}{m-1}D_C^\top D_C$. These expressions differ by a factor of $m/(m-1)$, which is unimportant if m is large, but makes a difference if m is small. The cchs function does the calculation in the same way as Langholz and Jiao and Cologne et al. (2012).

How the SAS code and S-Plus code work

The previously published SAS and S-Plus code-fragments for Estimator III manipulate the data in a completely different way from cchs:

1. Define a small number ϵ, which is smaller than any of the differences between consecutive event-times.
2. For each $i_j \notin C$, change the entry-time to $t_j - \epsilon$.
3. Replace the dataset with a new dataset that is constructed as follows. For each i_j:
   - 3.1 Make one copy of all the rows.
   - 3.2 Drop those of the new rows that are not at risk at t_j.
   - 3.3 For all rows, set the entry-time to $t_j - \epsilon$ and the exit-time to t_j, and set the event-indicator to 1 for i_j and 0 for the other rows.
4. For each $i_j \notin C$, choose a random participant $J_{s(i_j)}$ from $C \cap s(i_j)$ and remove the row that contains $J_{s(i_j)}$ and has exit-time t_j.

Step 3 radically changes the dataset and greatly increases its size. The result of the manipulations is that when the data is passed to coxph in S-Plus or `proc phreg` in SAS, each row corresponds to exactly

one additive term in the numerator or denominator of one multiplicative term of the pseudo-likelihood. The $\alpha_{s(k)}^{-1}$ terms and the adjustment to the covariance matrix are dealt with in the same way as in `cchs`.

The advantage of this method is that steps 3 and 4 are relatively simple—the "swapping" in step 4 just consists of choosing and removing a single row each time. The disadvantages are that if the original dataset is medium or large in size then this method uses a gigantic amount of memory and requires `coxph` to deal with a gigantic dataset. For step 3.1, the S-Plus code creates a data frame that contains one row for every combination of a case and a participant. In EPIC-CVD, a typical dataset that was supplied to an investigator contained about $31,000$ participants, of whom $14,000$ were cases, and this took up about 4.4MB as a binary file. So the data frame created by step 3 would have $31,000 \times 14,000 \approx 434$ million rows and take up about 62GB. This is too big for a desktop computer to store in memory, and even after some rows are removed in step 4, the data frame would be far too big for `coxph` to process in a reasonable amount of time.

In contrast, when `cchs` is used on the same dataset, the manipulated dataset has fewer than $31,000 + 14,000 = 45,000$ rows. This $45,000$ was calculated by supposing that for every case an extra row is created, whereas in reality some cases are in the subcohort and it is likely that some "swapper" rows are not at risk at the relevant event-times, and for these no extra rows are created, so the true number of rows in the manipulated dataset is less than $45,000$.

The manipulation of the data by the SAS and S-Plus code can be regarded as maximal, in the sense that each row ultimately corresponds to a single additive term in a single multiplicative term of the pseudo-likelihood. The manipulation by `cchs` can be regarded as minimal, since it makes only the manipulations that are strictly necessary in order for `coxph` to calculate the correct pseudo-likelihood, and it leaves the data with the smallest possible number of rows.

Obscure aspects of the calculation

Three obscure issues arise in calculating Estimator III. Under normal circumstances users of `cchs` can ignore these, but for development or testing it may be necessary to understand these issues or, as described below, set the relevant logical arguments to different values.

The first issue has to do with the `dropNeverAtRiskRows` argument, which determines whether rows should be dropped if their at-risk time does not contain any of the event-times. The dropped rows make no difference to the regression coefficients output by `cchs`, but they do affect the covariance-estimates and confidence intervals, because of the approximation that `residuals.coxph` uses to calculates the dfbeta residuals. The SAS code and S-Plus code drop these rows (in step 3.2), but Borgan et al. (2000) and Langholz and Jiao (2007a) make no mention of this issue. In `cchs`, `dropNeverAtRiskRows` is `TRUE` by default but can be set to `FALSE` if desired.

The second issue has to do with the `dropSubcohEventsDfbeta` argument. When the adjustment to the covariance matrix is calculated, the correct way is to use all the rows of the matrix of dfbeta residuals that correspond to subcohort members. The SAS code and S-Plus code both omit the rows that correspond to subcohort cases at their event-times, presumably because this makes the code simpler and is unlikely to change the output greatly. By default, `cchs` calculates the adjustment to the covariance matrix in the strictly correct way. But if `dropSubcohEventsDfbeta` is `TRUE`, then it calculates the adjustment in the same way as the SAS and S-Plus code, as follows. For each $i_j \in C$, it splits that row of the data at $t_j - \epsilon$ and replaces it with two rows; the first row has entry-time t_{0i_j}, exit-time $t_j - \epsilon$, and event-indicator 0; and the second row has entry-time $t_j - \epsilon$, exit-time t_j, and event-indicator 1. When calculating the adjustment to the covariance matrix, `cchs` excludes the row of the dfbeta residuals that corresponds to the second of these two rows.

The third issue is that the "swapper" $J_{s(i_j)}$ is supposed to be selected once for each stratum (see Section 3 of Borgan et al., 2000), but the SAS and S-Plus both select it once for each case. The `cchs` function selects the swapper correctly. It has no logical argument to specify how the swapper should be selected, since it manipulates the data in a completely different way from the SAS and S-Plus code.

Discussion

In case-cohort studies, stratified selection of the subcohort seems to be common (Sharp et al., 2014), so there are likely to be plenty of investigations where `cchs` may be useful. The way in which `cchs` manipulates the data makes it faster and more computationally efficient than the previously published SAS and S-Plus code. The three obscure issues described in the previous section should not greatly affect most analyses, but will make a difference with small datasets.

Acknowledgments

The author would like to thank Michael Sweeting for his helpful comments. This work was funded by the UK Medical Research Council (G0800270), the British Heart Foundation (SP/09/002), the UK National Institute for Health Research (Cambridge Biomedical Research Centre), the European Research Council (268834), and the European Commission Framework Programme 7 (HEALTH-F2-2012-279233).

Bibliography

S. Bingham and E. Riboli. Diet and cancer—the European Prospective Investigation into Cancer and Nutrition. *Nature Reviews Cancer*, 4(3):206–215, 2004. URL https://doi.org/10.1038/nrc1298. [p485]

Ø. Borgan, B. Langholz, S. O. Samuelsen, L. Goldstein, and J. Pogoda. Exposure stratified case–cohort designs. *Lifetime Data Analysis*, 6(1):39–58, 2000. URL https://doi.org/10.1023/A:1009661900674. [p484, 485, 486, 490, 491]

N. E. Breslow, T. Lumley, C. M. Ballantyne, L. E. Chambless, and M. Kulich. Improved Horvitz–Thompson estimation of model parameters from two-phase stratified samples: Applications in epidemiology. *Statistics in Biosciences*, 1(1):32–49, 2009a. URL https://doi.org/10.1007/s12561-009-9001-6. [p486, 487]

N. E. Breslow, T. Lumley, C. M. Ballantyne, L. E. Chambless, and M. Kulich. Using the whole cohort in the analysis of case–cohort data. *American Journal of Epidemiology*, 169(11):1398–1405, 2009b. URL https://doi.org/10.1093/aje/kwp055. [p486, 487]

J. Cologne, D. L. Preston, K. Imai, M. Misumi, K. Yoshida, T. Hayashi, and K. Nakachi. Conventional case–cohort design and analysis for studies of interaction. *International Journal of Epidemiology*, 41(4): 1174–1186, 2012. URL https://doi.org/10.1093/ije/dys102. [p484, 485, 486, 490]

D. R. Cox. Regression models and life-tables. *Journal of the Royal Statistical Society B*, 34(2):187–220, 1972. URL https://www.jstor.org/stable/2985181. [p484]

J. Danesh, R. Saracci, G. Berglund, E. Feskens, K. Overvad, S. Panico, S. Thompson, A. Fournier, F. Clavel-Chapelon, M. Canonico, R. Kaaks, J. Linseisen, H. Boeing, T. Pischon, C. Weikert, A. Olsen, A. Tjønneland, S. P. Johnsen, M. K. Jensen, J. R. Quirós, C. A. G. Svatetz, M.-J. S. Pérez, N. Larrañaga, C. N. Sanchez, C. M. Iribas, S. Bingham, K.-T. Khaw, N. Wareham, T. Key, A. Roddam, A. Trichopoulou, V. Benetou, D. Trichopoulos, G. Masala, S. Sieri, R. Tumino, C. Sacerdote, A. Mattiello, W. M. M. Verschuren, H. B. B. de Mesquita, D. E. Grobbee, Y. T. van der Schouw, O. Melander, G. Hallmans, P. Wennberg, E. Lund, M. Kumle, G. Skeie, P. Ferrari, N. Slimani, T. Norat, and E. Riboli. EPIC-Heart: The cardiovascular component of a prospective study of nutritional, lifestyle and biological factors in 520,000 middle-aged participants from 10 European countries. *European Journal of Epidemiology*, 22(2):129–141, 2007. URL https://doi.org/10.1007/s10654-006-9096-8. [p485]

G. J. D'Angio, N. Breslow, J. B. Beckwith, A. Evans, E. Baum, A. Delorimier, D. Fernbach, E. Hrabovsky, B. Jones, P. Kelalis, H. B. Othersen, M. Tefft, and P. R. M. Thomas. Treatment of Wilms' tumor: Results of the third National Wilms' Tumor Study. *Cancer*, 64(2):349–360, 1989. URL https://doi.org/10.1002/1097-0142(19890715)64:2<349::AID-CNCR2820640202>3.0.CO;2-Q. [p487]

V. P. Godambe. An optimum property of regular maximum likelihood estimation. *The Annals of Mathematical Statistics*, 31(4):1208–1211, 1960. URL https://doi.org/10.1214/aoms/1177705693. [p486]

V. P. Godambe. Conditional likelihood and unconditional optimum estimating equations. *Biometrika*, 63(2):277–284, 1976. URL https://doi.org/10.1093/biomet/63.2.277. [p486]

R. J. Gray. Weighted analyses for cohort sampling designs. *Lifetime Data Analysis*, 15(1):24–40, 2009. URL https://doi.org/10.1007/s10985-008-9095-z. [p486]

D. M. Green, N. E. Breslow, J. B. Beckwith, J. Z. Finklestein, P. E. Grundym, P. R. Thomas, T. Kim, S. J. Shochat, G. M. Haase, M. L. Ritchey, P. P. Kelalis, and G. J. D'Angio. Comparison between single-dose and divided-dose administration of dactinomycin and doxorubicin for patients with Wilms' tumor: a report from the National Wilms' Tumor Study Group. *Journal of Clinical Oncology*, 16(1):237–256, 1998. URL https://doi.org/10.1200/JCO.1998.16.1.237. [p487]

R. R. Huxley, F. L. Lopez, R. F. MacLehose, J. H. Eckfeldt, D. Couper, C. Leiendecker-Foster, R. C. Hoogeveen, L. Y. Chen, E. Z. Soliman, S. K. Agarwal, and A. Alonso. Novel association between plasma matrix metalloproteinase-9 and risk of incident atrial fibrillation in a case-cohort study: The Atherosclerosis Risk in Communities study. *PLOS One*, 8(3):1–8, 2013. URL https://doi.org/10.1371/journal.pone.0059052. [p487]

InterAct Consortium. Design and cohort description of the interact project: An examination of the interaction of genetic and lifestyle factors on the incidence of type 2 diabetes in the EPIC study. *Diabetologia*, 54(9):2272–2282, 2011. URL https://doi.org/10.1007/s00125-011-2182-9. [p485]

J. Jiao. *Comparison of Variance Estimators in Case–Cohort Studies*. PhD thesis, University of Southern California, 2001. URL http://digitallibrary.usc.edu/cdm/ref/collection/p15799coll16/id/242761. [p486, 490]

E. Jones. *cchs: Cox Model for Case–Cohort Data with Stratified Subcohort-Selection*, 2017. URL https://cran.r-project.org/package=cchs. R package version 0.4.0. [p484]

E. Jones, M. J. Sweeting, S. J. Sharp, and S. G. Thompson. A method making fewer assumptions gave the most reliable estimates of exposure–outcome associations in stratified case–cohort studies. *Journal of Clinical Epidemiology*, 68(12):1397–1405, 2015. URL https://doi.org/10.1016/j.jclinepi.2015.04.007. [p485, 487]

S. P. Juraschek, G. P. S. Shantha, A. Y. Chu, E. R. Miller III, E. Guallar, R. C. Hoogeveen, C. M. Ballantyne, F. L. Brancati, M. I. Schmidt, J. S. Pankow, and J. H. Young. Lactate and risk of incident diabetes in a case-cohort of the Atherosclerosis Risk in Communities (ARIC) study. *PLOS One*, 8(1):1–7, 2013. URL https://doi.org/10.1371/journal.pone.0055113. [p487]

H. A. Katki and S. D. Mark. Survival analysis for cohorts with missing covariate information. *R News*, 8(1):14–19, 2008. URL https://www.r-project.org/doc/Rnews/Rnews_2008-1.pdf#section*.31. [p484]

J. P. Kim, W. Lu, T. Sit, and Z. Ying. A unified approach to semiparametric transformation models under general biased sampling schemes. *Journal of the American Statistical Association*, 108(501):217–227, 2013. URL https://doi.org/10.1080/01621459.2012.746073. [p486]

M. Kulich and D. Lin. Improving the efficiency of relative-risk estimation in case–cohort studies. *Journal of the American Statistical Association*, 99(467):832–844, 2004. URL https://doi.org/10.1198/016214504000000584. [p486, 487]

M. M. Lamb, M. D. Simpson, J. Seifert, F. W. Scott, M. Rewers, and J. M. Norris. The association between IgG4 antibodies to dietary factors, islet autoimmunity and type 1 diabetes: The diabetes autoimmunity study in the young. *PLOS One*, 8(2):1–7, 2013. URL https://doi.org/10.1371/journal.pone.0057936. [p487]

B. Langholz and J. Jiao. Computational methods for case–cohort studies. *Computational Statistics & Data Analysis*, 51(8):3737–3748, 2007a. URL https://doi.org/10.1016/j.csda.2006.12.028. [p484, 485, 486, 487, 488, 490, 491]

B. Langholz and J. Jiao. Case–cohort computation. http://web.archive.org/web/20100720003853/http://hydra.usc.edu/timefactors/Examples/Case-cohort%20computational%20methods/Case-cohort%20computation.html, 2007b. [p484]

B. Lindsay. Conditional score functions: Some optimality results. *Biometrika*, 69(3):503–512, 1982. URL https://doi.org/10.1093/biomet/69.3.503. [p486]

D. Liu, T. Cai, and Y. Zheng. Evaluating the predictive value of biomarkers with stratified case-cohort design. *Biometrics*, 68(4):1219–1227, 2012. URL https://doi.org/10.1111/j.1541-0420.2012.01787.x. [p486]

T. Lumley. Analysis of complex survey samples. *Journal of Statistical Software*, 9(8):1–19, 2004. URL https://doi.org/10.18637/jss.v009.i08. [p484]

T. Lumley. *survey: Analysis of Complex Survey Samples*, 2017a. URL https://cran.r-project.org/package=survey. R package version 3.32-1. [p484]

T. Lumley. *Two-Phase Designs in Epidemiology*, 2017b. URL https://cran.r-project.org/web/packages/survey/vignettes/epi.pdf. [p486]

S. D. Mark and H. A. Katki. Specifying and implementing nonparametric and semiparametric survival estimators in two-stage (nested) cohort studies with missing case data. *Journal of the American Statistical Association*, 101(474):460–471, 2006. URL https://doi.org/10.1198/016214505000000952. [p484, 486]

R. Payne, M. Neykov, M. K. Jensen, and T. Cai. Kernel machine testing for risk prediction with stratified case cohort studies. *Biometrics*, 72(2):372–381, 2016. URL https://doi.org/10.1111/biom.12452. [p486]

R. L. Prentice. A case–cohort design for epidemiologic cohort studies and disease prevention trials. *Biometrika*, 73(1):1–11, 1986. URL https://doi.org/10.1093/biomet/73.1.1. [p484, 486]

D. B. Rubin. *Multiple Imputation for Nonresponse in Surveys*. John Wiley & Sons, New York, 1987. URL https://doi.org/10.1002/9780470316696. [p488]

S. O. Samuelsen, H. Ånestad, and A. Skrondal. Stratified case–cohort analysis of general cohort sampling designs. Technical report, Department of Mathematics, University of Oslo, 2006. URL https://www.duo.uio.no/handle/10852/10351. [p486]

S. O. Samuelsen, H. Ånestad, and A. Skrondal. Stratified case–cohort analysis of general cohort sampling designs. *Scandinavian Journal of Statistics*, 34(1):103–119, 2007. URL https://doi.org/10.1111/j.1467-9469.2006.00552.x. [p484, 486]

SAS Institute Inc. *SAS Software, Version 8*. Cary, NC, 1999. URL http://www.sas.com/. [p484]

J. L. Schafer. Multiple imputation: a primer. *Statistical Methods in Medical Research*, 8(1):3–15, 1999. URL https://doi.org/10.1177/096228029900800102. [p488]

T. A. Severini. Frequency properties of inferences based on an integrated likelihood function. *Statistica Sinica*, 21(1):433–447, 2011. URL https://www.jstor.org/stable/24309279. [p486]

S. J. Sharp, M. Poulaliou, S. G. Thompson, I. R. White, and A. M. Wood. A review of published analyses of case–cohort studies and recommendations for future reporting. *PLOS One*, 9(6):e101176, 2014. URL https://doi.org/10.1371/journal.pone.0101176. [p487, 491]

T. M. Therneau. *survival: A Package for Survival Analysis in S*, 2017. URL https://cran.r-project.org/package=survival. R package version 2.41-3. [p484]

T. M. Therneau and P. M. Grambsch. *Modeling Survival Data: Extending the Cox Model*. Springer-Verlag, New York, 2000. [p484]

T. M. Therneau and H. Li. Computing the Cox model for case–cohort designs. *Lifetime Data Analysis*, 5 (2):99–112, 1999. URL https://doi.org/10.1023/A:1009691327335. [p485, 490]

Tibco Software Inc. *S-Plus Software, Version 8*. Palo Alto, CA, 2007. URL http://www.tibco.com/. [p484]

Y. Yan, H. Zhou, and J. Cai. Improving efficiency of parameter estimation in case-cohort studies with multivariate failure time data. *Biometrics*, 73(3):1042–1052, 2017. URL https://doi.org/10.1111/biom.12657. [p486]

D. Zeng and D. Y. Lin. Maximum likelihood estimation in semiparametric regression models with censored data. *Journal of the Royal Statistical Society B*, 69(4):507–564, 2007. URL https://doi.org/10.1111/j.1369-7412.2007.00606.x. [p486]

Edmund Jones
Cardiovascular Epidemiology Unit
Department of Public Health and Primary Care
University of Cambridge
United Kingdom
edmundjones79@gmail.com

Small Area Disease Risk Estimation and Visualization Using R

by Paula Moraga

Abstract Small area disease risk estimation is essential for disease prevention and control. In this paper, we demonstrate how R can be used to obtain disease risk estimates and quantify risk factors using areal data. We explain how to define disease risk models and how to perform Bayesian inference using the **INLA** package. We also show how to make interactive maps of estimates using the **leaflet** package to better understand the disease spatial patterns and communicate the results. We show an example of lung cancer risk in Pennsylvania, United States, in year 2002, and demonstrate that R represents an excellent tool for disease surveillance by enabling reproducible health data analysis.

Introduction

Disease risk mapping analyses can help to better understand the spatial variation of the disease, and allow the identification of important public health determinants. These analyses are essential to inform programmes of disease prevention and control. The increased availability of geospatial disease and population data has enabled to study a number of health outcomes worldwide such as influenza and cancer in developed countries (Moraga and Ozonoff, 2013; Moraga and Kulldorff, 2016), and neglected tropical diseases (Moraga et al., 2015; Hagan et al., 2016).

Areal disease data often arise when disease outcomes observed at point level locations are aggregated over subareas of the study region due to several reasons such as patient confidentiality. Producing disease risk estimates at areal level is complicated by the fact that raw rates can be very unstable in areas with small populations and for rare diseases, and also by the presence of spatial correlation that may exist due to spatially correlated risk factors (Leroux et al., 2000). Thus, generalized linear mixed models are often used to obtain disease risk estimates since they enable to improve local estimates by accommodating spatial correlation and the effects of explanatory variables. Bayesian inference in these models may be performed using the Integrated Nested Laplace Approximation (INLA) approach (Rue et al., 2009) which is a computational alternative to MCMC that allows to do approximate Bayesian inference in latent Gaussian models. This approach is implemented in the R package called **INLA** (Rue et al., 2017) (`http://www.r-inla.org/`).

Small area disease estimates can be visualized through maps, greatly facilitating effective communication. R provides excellent tools for visualization including packages for making interactive maps such as **leaflet** (Cheng et al., 2017). The maps created with **leaflet** support interactive panning and zooming which is very convenient to examine small areas in detail.

In this paper, we illustrate the use of R for performing disease risk mapping analysis using areal data. First, we introduce disease risk models for areal data and give a brief overview of INLA. In Section Z.4 we show how to estimate lung cancer risk and quantify risk factors in Pennsylvania, United States, in year 2000. Specifically, we discuss how to compute the observed and expected disease counts in the Pennsylvania counties, how to obtain disease risk estimates by fitting a spatial disease risk model using **INLA**, and how to build interactive maps showing the risk estimates using **leaflet**. Finally, the conclusions are presented.

Disease risk models

Disease risk estimates in areas can be obtained by computing the Standardized Incidence Ratios (SIRs). For area i, $i = 1, \ldots, n$, the SIR is obtained as the ratio of the observed to the expected disease counts: $\text{SIR}_i = Y_i / E_i$. The expected counts represent the total number of disease cases that one would expect if the population of the specific area behaved the way the standard (or regional) population behaves. The expected counts can be calculated using indirect standardization as

$$E_i = \sum_{j=1}^{m} r_j^{(s)} n_j,$$

where $r_j^{(s)}$ is the disease rate in stratum j of the standard population, and n_j is the population in stratum j of the specific area. The SIR corresponding to area i, SIR_i, indicates whether the area i has more ($\text{SIR}_i > 1$), equal ($\text{SIR}_i = 1$) or fewer ($\text{SIR}_i < 1$) cases observed than expected from the standard population. When applied to mortality data, the ratio is commonly known as the Standardized

Mortality Ratio or SMR.

Although in some situations SIRs can give a sense of the disease's spatial variability, very extreme values can occur in areas with small populations owing to the small sample sizes involved. In contrast, disease models are preferred to obtain disease risks estimates because they enable to incorporate covariates and borrow information from neighboring areas to improve local estimates, resulting in the smoothing or shrinking of extreme values based on small sample sizes (Gelfand et al., 2000). A common approach is to model the observed counts Y_i, $i = 1, \ldots, n$, using a Poisson distribution with mean $E_i \times \theta_i$, where E_i is the expected counts and θ_i is the relative risk in area i. Then, the log risks are modeled with a sum of an intercept to model the overall disease risk level, and random effects that account for extra-Poisson variability in the observed data (Lawson, 2009). Areas with relative risks $\theta > 1$ and $\theta < 1$ are areas with high and low risks, respectively. Areas with $\theta = 1$ have the same risk as expected from the standard population.

The general model in disease mapping is expressed as

$$Y_i \sim Po(E_i \times \theta_i),\ i = 1, \ldots, n,$$

$$\log(\theta_i) = \alpha + u_i + v_i.$$

Here, α denotes the overall risk level, u_i is a spatial structured random effect that models the spatial dependence between the relative risks, and v_i is an unstructured exchangeable random effect that models uncorrelated noise. Often, other covariates or random effects are also included to quantify risk factors and deal with other sources of variability.

A model commonly used in disease mapping is the Besag-York-Mollié (BYM) model (Besag et al., 1991). In this model, the spatially structured component u_i is modelled with the conditional autoregressive (CAR) distribution which smoothes the data according to a certain adjacency structure given by a neighborhood matrix that specifies two areas are neighbours if they have a common boundary. The CAR distribution is expressed as

$$u_i|\mathbf{u}_{-\mathbf{i}} \sim N\left(\bar{u}_{\delta_i}, \frac{\sigma_u^2}{n_{\delta_i}}\right),$$

where $\bar{u}_{\delta_i} = n_{\delta_i}^{-1} \sum_{j \in \delta_i} u_j$, and δ_i and n_{δ_i} represent, respectively, the set of neighbours and the number of neighbours of area i. The unstructured component v_i is modelled using independent and identically distributed normal variables with zero-mean and variance equal to σ_v^2.

INLA

Traditionally, Bayesian inference has been implemented via MCMC methods which make inference tractable for complex models but may present convergence and computation time problems. Integrated Nested Laplace Approximation (INLA) is a computational less-intensive alternative to MCMC designed to perform approximate Bayesian inference in latent Gaussian models (Rue et al., 2009). These models include a very wide and flexible class of models ranging from generalized linear mixed to spatial and spatio-temporal models. Specifically, models are of the form

$$y_i|\mathbf{x}, \boldsymbol{\theta} \sim \pi(y_i|x_i, \boldsymbol{\theta}),$$

$$\mathbf{x}|\boldsymbol{\theta} \sim N(\mathbf{0}, \boldsymbol{Q}(\boldsymbol{\theta})^{-1}),$$

$$\boldsymbol{\theta} \sim \pi(\boldsymbol{\theta}),$$

where $\mathbf{y}$ are the observed data, $\mathbf{x}$ represents a Gaussian field, and $\boldsymbol{\theta}$ are hyperparameters. Observations y_i are assumed to belong to an exponential family with mean $\mu_i = g^{-1}(\eta_i)$. The linear predictor η_i accounts for effects of various covariates in an additive way,

$$\eta_i = \alpha + \sum_{k=1}^{n_\beta} \beta_k z_{ki} + \sum_{j=1}^{n_f} f^{(j)}(u_{ji}).$$

Here, α is the intercept, $\{\beta_k\}$'s quantify the linear effects of covariates $\{z_{ki}\}$ on the response, and $\{f^{(j)}(\cdot)\}$'s are a set of non-linear or smooth functions defined in terms of some covariates $\{u_{ji}\}$. This formulation permits to accommodate a wide range of models thanks to the very different forms that the functions $\{f^{(j)}\}$ can take including the disease risk models previously introduced. INLA uses a combination of analytical approximation and numerical integration to obtain approximated posterior distributions of the parameters that can then be post-processed to compute quantities of interest like posterior expectations and quantiles.

The INLA approach is implemented in the R package **INLA**. This package is not on CRAN because it uses some external C libraries that make difficult to build the binaries. Therefore, when we install the package we need to use `install.packages()` adding the URL of the **INLA** repository:

```
install.packages("INLA", repos = "https://inla.r-inla-download.org/R/stable",
                 dep = TRUE)
```

To fit a model using **INLA** we need to take two steps. First, we need to write the linear predictor of the model as a formula object in R. Then, we run the model calling the `inla()` function where we specify the formula, the family, the data and other options. Results can be inspected with the `summary()` function and the posterior distributions can be post-processed using a set of specific functions provided by **INLA**. Further details about how to use all these functions will be given in the disease mapping example in next Section.

Example: lung cancer risk in Pennsylvania

In this Section we present an example of small area disease mapping study where we estimate the risk of lung cancer in Pennsylvania counties in year 2002. We use the data contained in the R package **SpatialEpi** (Kim and Wakefield, 2016). The data contain the counties population which was obtained from the 2000 decennial census, and the lung cancer and smoking proportions which were obtained from the Pennsylvania Department of Health. We show how to calculate the observed and expected disease cases, and the the SIRs in each of the counties. We also obtain disease risk estimates and quantify risk factors by fitting a Bayesian model using **INLA**. Finally, we show how to make interactive maps of the risk estimates using **leaflet**.

Data

We start by loading the **SpatialEpi** package and attaching the `pennLC` data.

```
library(SpatialEpi)
data(pennLC)
```

By typing `?pennLC` we see `pennLC` is a list with the following elements:

- `geo`: a data frame of county ids, and longitude and latitude of the geographic centroid of each county,
- `data`: a data frame of county ids, number of cases, population and strata information,
- `smoking`: a data frame of county ids and proportion of smokers,
- `spatial.polygon`: a `SpatialPolygons` object with the map of Pennsylvania.

`pennLC$data` contains the number of lung cancer cases and the population at county level, stratified on race (white and non-white), gender (female and male) and age group (under 40, 40-59, 60-69 and 70+).

We now create a data frame called `d` with columns containing the counties ids, the observed and expected number of cases, the smoking proportions and the SIRs. Specifically, `d` will contain the following columns:

- `id`: id of each county,
- `Y`: observed number of cases in each county,
- `E`: expected number of cases in each county,
- `smoking`: smoking proportion in each county,
- `SIR`: SIR of each county.

Observed cases

`pennLC$data` contains the cases in each county stratified by race, gender and age. We can obtain the number of cases in each county, `Y`, by aggregating the rows of `pennLC$data` by county and adding up the observed number of cases.

```
d <- aggregate(x = pennLC$data$cases, by = list(county = pennLC$data$county),
               FUN = sum)
```

`aggregate()` returns a data frame where the first row is the county and the second column is the observed number of cases in each of the counties. We set the column names of the returned object equal to `id` and `Y`.

```
names(d) <- c("id", "Y")
```

Expected cases

Now we calculate the indirectly standardized expected number of cases in each county as explained in Section Z.2. That is, we use the strata-specific rates from the the Pennsylvania population (standard population), and apply them to the population distribution of the county. The expected counts represent the total number of disease cases one would expect if the population in the county behaved the way the Pennsylvania population behaves. We can do this by using the `expected()` function of **SpatialEpi**. This function has three arguments, namely,

- `population`: a vector of population counts for each strata in each area,
- `cases`: a vector with the number of cases for each strata in each area,
- `n.strata`: number of strata considered.

Vectors `population` and `cases` have to be sorted by area first and then, within each area, the counts for all strata need to be listed in the same order. All strata need to be included in the vectors, including strata with 0 cases. Hence, to get the expected counts we first sort the data using the `order()` function where we specify the order as county, race, gender and finally age.

```
pennLC$data <- pennLC$data[order(pennLC$data$county, pennLC$data$race,
                                 pennLC$data$gender, pennLC$data$age), ]
```

Then we call the `expected()` function to obtain the expected counts `E` in each county. In the function we set `population` equal to `pennLC$data$population` and `cases` equal to `pennLC$data$cases`. There are 2 races, 2 genders and 4 age groups for each county, so number of strata is set to 2 x 2 x 4 = 16.

```
population <- pennLC$data$population
cases <- pennLC$data$cases
n.strata <- 16
E <- expected(population, cases, n.strata)
```

Now we add the vector `E` to the data frame `d` which contains the counties ids (`id`) and the observed counts (`Y`), making sure the `E` elements correspond to the counties in `d$id` in the same order. To do that, we use `match()` to calculate the vector of the positions that match `d$id` in `unique(pennLC$data$county)` which are the corresponding counties of `E`. Then we rearrange `E` using that vector.

```
d$E <- E[match(d$id, unique(pennLC$data$county))]
```

Smokers proportions

We also add to `d` the variable `smoking` which represents the proportion of smokers in each county. We add this variable using the `merge()` function where we specify the columns for merging as `id` in `d` and `county` in `pennLC$smoking`.

```
d <- merge(d, pennLC$smoking, by.x = "id", by.y = "county")
```

SIRs

Finally, we compute the vector of SIRs as the ratio of the observed to the expected counts, and add it to the data frame `d`.

```
d$SIR <- d$Y/d$E
```

Add data to map

The map of Pennsylvania counties is given by the `SpatialPolygons` object called `pennLC$spatial.polygon`. Using this object and the data frame `d` we can create a `SpatialPolygonsDataFrame` called `map`, that will allow us to make maps of the variables in `d`. In order to do that, we first set the row names of the data frame `d` equal to `d$id`. Then we merge `pennLC$spatial.polygon` and `d` matching the `SpatialPolygons` member Polygons ID slot values with the data frame row names.

```
library(sp)
rownames(d) <- d$id
map <- SpatialPolygonsDataFrame(pennLC$spatial.polygon, d, match.ID = TRUE)
head(map@data)
```

```
##                  id    Y          E smoking       SIR
## adams         adams   55   69.62730   0.234 0.7899200
## allegheny allegheny 1275 1182.42804   0.245 1.0782897
## armstrong armstrong   49   67.61012   0.250 0.7247435
## beaver       beaver  172  172.55806   0.276 0.9967660
## bedford     bedford   37   44.19013   0.228 0.8372910
## berks         berks  308  300.70598   0.249 1.0242563
```

Mapping variables

We can visualize the observed and expected disease counts, the SIRs, as well as the smokers proportions in an interactive chropleth map using the **leaflet** package. We create the map by first calling `leaflet()` and adding the default OpenStreetMap map tiles to the map with `addTiles()`. Then we add the Pennsylvania counties with `addPolygons()` where we specify the areas boundaries color (`color`) and the stroke width (`weight`). We fill the areas with the colours given by the color palette function generated with `colorNumeric()`, and set `fillOpacity` to a value less than 1 to be able to see the background map. We use `colorNumeric()` to create a color palette function that maps data values to colors according to a given palette. We create the function using the parameters `palette` with the color function that values will be mapped to, and `domain` with the possible values that can be mapped. Finally, we add the legend by specifying the color palette function (`pal`) and the values used to generate colors from the palette function (`values`). We set `opacity` to the same value as the opacity in the areas, and specify a title and a position for the legend.

```
library(leaflet)
l <- leaflet(map) %>% addTiles()
pal <- colorNumeric(palette = "YlOrRd", domain = map$SIR)
l %>% addPolygons(color = "grey", weight = 1, fillColor = ~pal(SIR),
                  fillOpacity = 0.5) %>%
  addLegend(pal = pal, values = ~SIR, opacity = 0.5, title = "SIR",
            position = "bottomright")
```

We can improve the map by highlighting the counties when the mouse hovers over them, and showing information about the observed and expected counts, SIRs, and smoking proportions. We do this by adding the arguments `highlightOptions`, `label` and `labelOptions` to `addPolygons()`. We choose to highlight the areas using a bigger stroke width (`highlightOptions(weight = 4)`). We create the labels using HTML syntax. First, we create the text to be shown using the function `sprintf()` which returns a character vector containing a formatted combination of text and variable values and then applying `htmltools::HTML()` which marks the text as HTML. In `labelOptions` we specify the labels `style`, `textsize`, and `direction`. Possible values for `direction` are `left`, `right` and `auto` and this specifies the direction the label displays in relation to the marker. We choose `auto` so the optimal direction will be chosen depending on the position of the marker.

```
labels <- sprintf("<strong> %s </strong> <br/> Observed: %s <br/> Expected: %s <br/>
                   Smokers proportion: %s <br/> SIR: %s",
                  map$id, map$Y, round(map$E, 2), map$smoking, round(map$SIR, 2)) %>%
  lapply(htmltools::HTML)

l %>% addPolygons(color = "grey", weight = 1, fillColor = ~pal(SIR), fillOpacity = 0.5,
                  highlightOptions = highlightOptions(weight = 4), label = labels,
                  labelOptions = labelOptions(style = list("font-weight" = "normal",
                                                           padding = "3px 8px"),
                                              textsize = "15px",
                                              direction = "auto")) %>%
  addLegend(pal = pal, values = ~SIR, opacity = 0.5, title = "SIR",
            position = "bottomright")
```

Figure 1 shows a snapshot of the interactive map created using `leaflet` showing the SIRs in the Pennsylvania counties. We can examine the map and see which counties have SIR equal to 1 indicating observed counts are the same as expected counts, and which counties have SIR greater (or smaller) than 1, indicating observed counts are greater (or smaller) than expected counts.

This map gives a sense of the disease risk across Pennsylvania. However, SIRs are misleading and insufficiently reliable in counties with small populations. In contrast, model-based approaches enable to incorporate covariates and borrow information from neighboring counties to improve local estimates, resulting in the smoothing of extreme values based on small sample sizes. In the next section we will show how to obtain disease risk estimates using a Bayesian model using **INLA**.

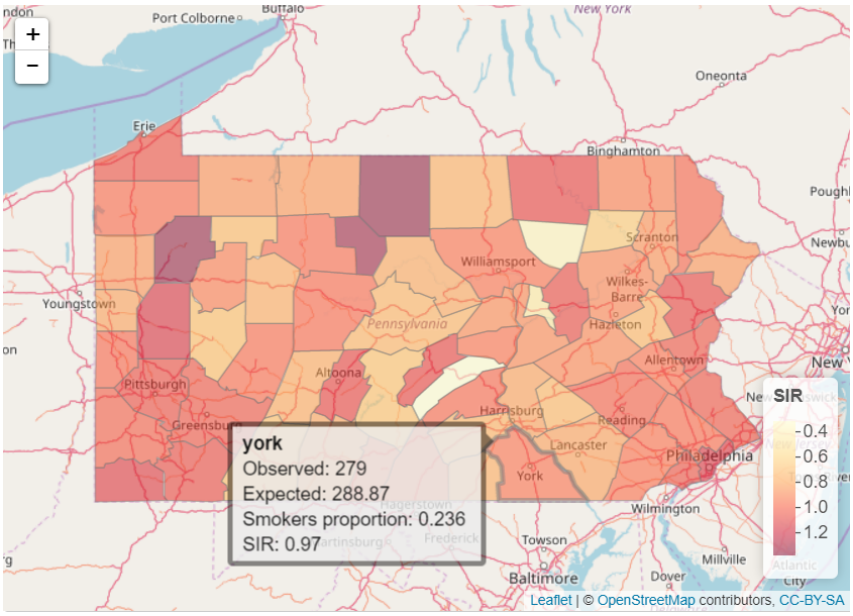


Figure 1: Snapshot of the interactive map created using `leaflet` showing the lung cancer SIRs in Pennsylvania counties in 2002.

Modeling

In this Section we specify the model for the data, and detail the required steps to fit the model and obtain the disease risk estimates using **INLA**.

Model

We specify a model assuming that the observed counts Y_i are conditionally independently Poisson distributed,

$$Y_i|\theta_i \sim Po(E_i \times \theta_i),\ i = 1, \ldots, n,$$

where E_i is the expected count and θ_i is the relative risk in area i. The logarithm of θ_i is expressed as follows:

$$\log(\theta_i) = \beta_0 + \beta_1 \times smoking_i + u_i + v_i,$$

where β_0 is the intercept, β_1 is the coefficient of the smokers proportion covariate, u_i is an structured spatial effect, $u_i|\mathbf{u_{-i}} \sim N(\bar{u}_{\delta_i}, \frac{\sigma_u^2}{n_{\delta_i}})$, and v_i is an unstructured spatial effect, $v_i \sim N(0, \sigma_v^2)$.

Neighbourhood matrix

We create the neighbourhood matrix needed to define the spatial random effect using the `poly2nb()` and the `nb2INLA()` functions of the **spdep** package (Bivand, 2017). First, we use `poly2nb()` to create a neighbours list based on areas with contiguous boundaries. Then, we use `nb2INLA()` to convert this list into a file with the representation of the neighbourhood matrix as required by **INLA** that is saved in the working directory. Then we read the file using the `inla.read.graph()` function of **INLA**, and store it in the object g which we will later use for specifying the spatial disease model with **INLA**.

```
library(spdep)
library(INLA)
nb <- poly2nb(map)
head(nb)

## [[1]]
## [1] 21 28 67
##
## [[2]]
## [1]  3  4 10 63 65
##
## [[3]]
```

```
## [1]  2 10 16 32 33 65
##
## [[4]]
## [1]  2 10 37 63
##
## [[5]]
## [1]  7 11 29 31 56
##
## [[6]]
## [1] 15 36 38 39 46 54

nb2INLA("map.adj", nb)
g <- inla.read.graph(filename = "map.adj")
```

Inference using INLA

As stated in Section Z.4.3, the model includes two random effects, namely, u_i for modeling spatial residual variation, and v_i for modeling unstructured noise. We need to include two vectors in the data that denote the indices of these random effects. We call `re_u` the vector denoting u_i, and `re_v` the vector denoting v_i. We set both `re_u` and `re_v` equal to $1, \ldots, n$, where n is the number of counties. In our example, $n = 67$ and this can be obtained with the number of rows in the data (`nrow(map@data)`).

```
map$re_u <- 1:nrow(map@data)
map$re_v <- 1:nrow(map@data)
```

We specify the model formula by including the response in the left-hand side, and the fixed and random effects in the right-hand side. Random effects are set using `f()` with parameters equal to the name of the variable and the chosen model. For u_i, we use `model = "besag"` with neighbourhood matrix given by `g`. For v_i we choose `model = "iid"`.

```
formula <- Y ~ smoking + f(re_u, model = "besag", graph = g) + f(re_v, model = "iid")
```

We fit the model by calling the `inla()` function. We specify the formula, family, data, and the expected counts, and set `control.predictor` equal to `list(compute = TRUE)` to compute the posterior means of the linear predictors.

```
res <- inla(formula, family = "poisson", data = map@data, E = E,
            control.predictor = list(compute = TRUE))
```

Results

We can inspect the results object `res` using `summary()`.

```
summary(res)

##
## Call:
## c("inla(formula = formula, family = \"poisson\", data = map@data, ",  "    E = E,
## control.predictor = list(compute = TRUE))")
##
## Time used:
##  Pre-processing    Running inla Post-processing           Total
##          0.3179          1.1198          0.2155          1.6531
##
## Fixed effects:
##                mean     sd 0.025quant 0.5quant 0.975quant    mode kld
## (Intercept) -0.3236 0.1503    -0.6212  -0.3234    -0.0279 -0.3231   0
## smoking      1.1567 0.6247    -0.0809   1.1582     2.3853  1.1619   0
##
## Random effects:
## Name   Model
##  re_u   Besags ICAR model
## re_v   IID model
##
## Model hyperparameters:
##                          mean       sd 0.025quant 0.5quant 0.975quant
```

```
## Precision for re_u    93.05    49.95      30.95    81.84     220.42
## Precision for re_v 17956.92 18118.49    1155.86 12549.06   65967.91
##                       mode
## Precision for re_u   63.63
## Precision for re_v 3103.11
##
## Expected number of effective parameters(std dev): 18.61(4.366)
## Number of equivalent replicates : 3.60
##
## Marginal log-Likelihood:  -320.53
## Posterior marginals for linear predictor and fitted values computed
```

We see the intercept $\hat{\beta}_0$= -0.3236 with a 95% credible interval equal to (-0.6212, -0.0279), and the coefficient of smoking is $\hat{\beta}_1 = 1.1567$ with a 95% credible interval equal to (-0.0810, 2.3853). This indicates that the smokers proportion has a positive although non significant effect on disease risk. We can plot the posterior distribution of the smoking coefficient. We do this by calculating a spline smoothing of the marginal distribution of the coefficient with `inla.smarginal()` and then plot it with `ggplot()` of **ggplot2** package (Wickham and Chang, 2016) (see Figure 2).

```
library(ggplot2)
marginal <- inla.smarginal(res$marginals.fixed$smoking)
marginal <- data.frame(marginal)
ggplot(marginal, aes(x = x, y = y)) + geom_line() +
  labs(x = expression(beta[1]), y = "Density") +
  geom_vline(xintercept = 0, col = "blue") + theme_bw()
```

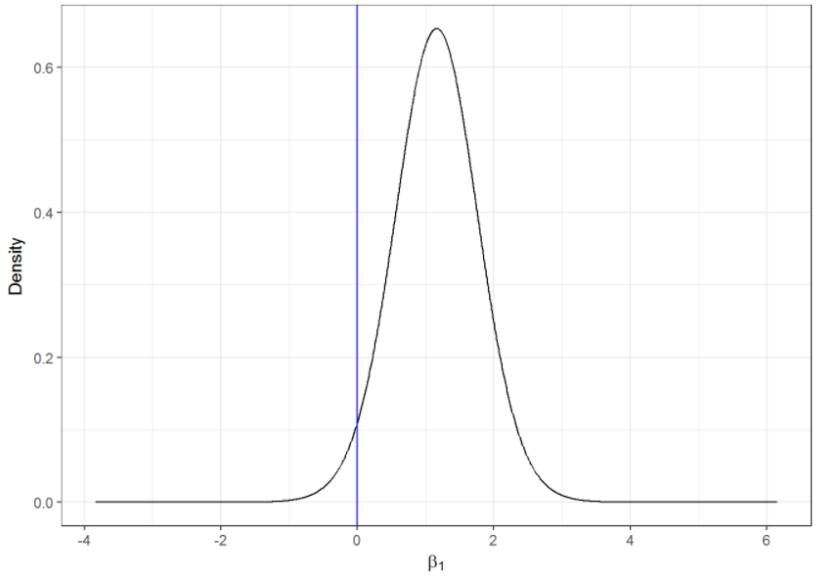


Figure 2: Posterior distribution of the coefficient of covariate smokers proportion.

The disease risk estimates and uncertainty for each of the counties are given by the mean posterior and the 95% credible intervals of $\theta_i, i = 1, \ldots, n$ which are in the data frame `res$summary.fitted.values`. Here, column `mean` is the mean posterior and `0.025quant` and `0.975quant` are the 2.5 and 97.5 percentiles, respectively. We add these data to `map` to be able to make maps of these variables. We assign column `mean` to the estimate of the relative risk, and columns `0.025quant` and `0.975quant` to the lower and upper limits of 95% credible intervals of the risks.

```
head(res$summary.fitted.values)
```

```
##                          mean         sd 0.025quant  0.5quant 0.975quant
## fitted.Predictor.01 0.8793912 0.05856462  0.7633052 0.8797123  0.9943822
## fitted.Predictor.02 1.0597516 0.02768817  1.0067707 1.0592960  1.1153106
## fitted.Predictor.03 0.9632142 0.05186119  0.8555748 0.9649813  1.0612639
## fitted.Predictor.04 1.0270184 0.05121187  0.9270286 1.0267054  1.1289998
## fitted.Predictor.05 0.9076782 0.05497745  0.7978867 0.9081541  1.0156557
```

```
## fitted.Predictor.06 0.9951907 0.04023105  0.9184425 0.9943119  1.0770673
##                           mode
## fitted.Predictor.01 0.8808707
## fitted.Predictor.02 1.0583769
## fitted.Predictor.03 0.9685918
## fitted.Predictor.04 1.0262831
## fitted.Predictor.05 0.9096874
## fitted.Predictor.06 0.9927333

map$RR <- res$summary.fitted.values[, "mean"]
map$LL <- res$summary.fitted.values[, "0.025quant"]
map$UL <- res$summary.fitted.values[, "0.975quant"]
```

Mapping disease risk

We show the estimated disease risk in an interactive map using `leaflet`. In the map, we add labels that appear when mouse hovers over the counties showing information about observed and expected counts, SIRs, smokers proportions, RRs, and lower and upper limits of 95% credible intervals.

```
pal <- colorNumeric(palette = "YlOrRd", domain = map$RR)

labels <- sprintf("<strong> %s </strong> <br/> Observed: %s <br/> Expected: %s <br/>
                  Smokers proportion: %s <br/> SIR: %s <br/> RR: %s (%s, %s)",
                  map$id, map$Y,  round(map$E, 2),  map$smoking, round(map$SIR, 2),
                  round(map$RR, 2), round(map$LL, 2), round(map$UL, 2)) %>%
  lapply(htmltools::HTML)

leaflet(map) %>% addTiles() %>%
  addPolygons(color = "grey", weight = 1, fillColor = ~pal(RR),  fillOpacity = 0.5,
              highlightOptions = highlightOptions(weight = 4), label = labels,
              labelOptions = labelOptions(style = list("font-weight" = "normal",
                                                       padding = "3px 8px"),
                                          textsize = "15px", direction = "auto")) %>%
  addLegend(pal = pal, values = ~RR, opacity = 0.5, title = "RR",
            position = "bottomright")
```

A snapshot of the interactive map created is shown in Figure 3. We observe counties with greater disease risk are located in the west and south east of Pennsylvania, and counties with lower risk are located in the center. The 95% credible intervals give a measure of the uncertainty in the risk estimates.

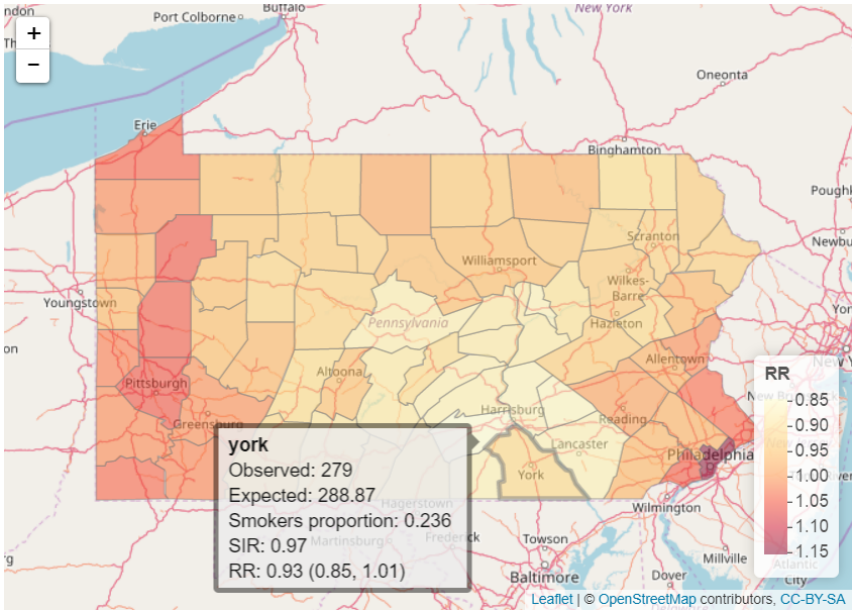


Figure 3: Snapshot of the interactive map created using `leaflet` showing the lung cancer RRs in Pennsylvania counties in 2002.

By typing `range(map@data$SIR)` and `range(map@data$RR)` we see that the range of SIRs is much wider about 1 compared to the range of RRs: (0.32 to 1.37) versus (0.85 to 1.15). We can also see the shrinkage of the RRs toward 1 by comparing maps of SIRs and RRs created using the same scale on the SIR gradient legend (see Figure 4). For example, in the SIRs map, in the central, less populated part of the state, there are 3 counties with extreme high values (dark colour) and 3 counties with extreme low values (light colour). In the RRs map, these extreme values shrink toward values closer to 1.

```
pal <- colorNumeric(palette = "YlOrRd", domain = map$SIR)

leaflet(map) %>% addTiles() %>%
  addPolygons(color = "grey", weight = 1, fillColor = ~pal(RR),  fillOpacity = 0.5,
              highlightOptions = highlightOptions(weight = 4), label = labels,
              labelOptions = labelOptions(style = list("font-weight" = "normal",
                                                       padding = "3px 8px"),
                                          textsize = "15px", direction = "auto")) %>%
  addLegend(pal = pal, values = ~RR, opacity = 0.5, title = "RR",
            position = "bottomright")
```

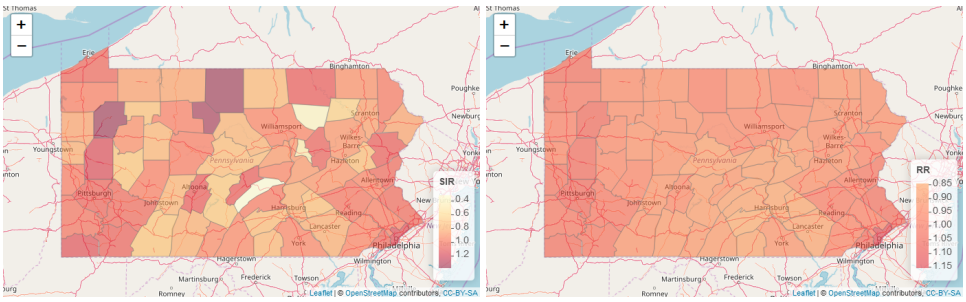


Figure 4: Snapshot of the interactive maps created using `leaflet` showing the lung cancer SIRs (left) and RRs (right) in Pennsylvania counties in 2002 using the same scale.

Summary

In this article we have shown how to obtain small area disease risk estimates, and generate interactive maps that help the understanding and interpretation of the results. First, we have introduced disease risk models using areal data, and have given an overview of the **INLA** package for performing Bayesian inference. Then, we have given a practical example where we have estimated lung cancer risk in Pennsylvania in 2002. We have conducted the analyses using several R packages such as **spdep** for spatial data manipulation, **SpatialEpi** for calculating the expected disease counts in the Pennsylvania counties, **INLA** for performing Bayesian inference, and **leaflet** and **ggplot2** for visualization of the results.

One limitation of disease models based on areal data is that they are often subject to ecological bias. This bias occurs when associations obtained from analyses that use variables at an aggregated level lead to conclusions different from analyses that use the same variables measured at an individual level (Robinson, 1950). Therefore, whenever point data are available, it is preferable to use disease models without aggregating data and predict disease risk in a continuous surface (Moraga et al., 2017; Diggle et al., 2013).

It is also possible to use R to build tools to better communicate the results to stakeholders and the general public. For instance, summaries and maps of disease risk estimates can be presented in interactive dashboards using **flexdashboard** (Allaire, 2017), and web applications using **shiny** (Chang et al., 2017). One example of such web application is the **SpatialEpiApp** package (Moraga, 2017a,b) which is useful for disease risk mapping and the detection of clusters. This is an easy to use application where users simply need to upload their data and click several buttons that execute the tasks and process the outputs, making spatial analysis methods accessible to multiple disciplines. **SpatialEpiApp** creates interactive visualizations by using the packages **leaflet** for rendering maps, **dygraphs** (Vanderkam et al., 2017) for plotting time series, and **DT** (Xie, 2016) for displaying data tables, and enables the generation of reports by using **rmarkdown** (Allaire et al., 2017). In conclusion, R represents an excellent tool for disease surveillance by enabling reproducible health data analysis.

Bibliography

J. Allaire. *Flexdashboard: R Markdown Format for Flexible Dashboards*, 2017. URL https://CRAN.R-project.org/package=flexdashboard. R package version 0.5. [p504]

J. Allaire, Y. Xie, J. McPherson, J. Luraschi, K. Ushey, A. Atkins, H. Wickham, J. Cheng, and W. Chang. *Rmarkdown: Dynamic Documents for R*, 2017. URL https://CRAN.R-project.org/package=rmarkdown. R package version 1.8. [p504]

J. Besag, J. York, and A. Mollié. Bayesian image restoration with applications in spatial statistics (with discussion). *Annals of the Institute of Statistical Mathematics*, 43:1–59, 1991. URL https://doi.org/10.1007/BF00116466. [p496]

R. Bivand. *Spdep: Spatial Dependence: Weighting Schemes, Statistics and Models*, 2017. URL https://CRAN.R-project.org/package=spdep. R package version 0.6-13. [p500]

W. Chang, J. Cheng, J. Allaire, Y. Xie, and J. McPherson. *Shiny: Web Application Framework for R*, 2017. URL https://CRAN.R-project.org/package=shiny. R package version 1.0.5. [p504]

J. Cheng, B. Karambelkar, and Y. Xie. *Leaflet: Create Interactive Web Maps with the JavaScript 'Leaflet' Library*, 2017. URL http://rstudio.github.io/leaflet/. R package version 1.1.0.9000. [p495]

P. J. Diggle, P. Moraga, B. Rowlingson, and B. Taylor. Spatial and Spatio-Temporal Log-Gaussian Cox Processes: Extending the Geostatistical Paradigm. *Statistical Science*, 28(4):542–563, 2013. URL https://doi.org/10.1214/13-STS441. [p504]

A. E. Gelfand, P. J. Diggle, M. Fuentes, and P. Guttorp. *Handbook of Spatial Statistics*. Chapman and Hall/CRC, Boca Raton, Fl., 2000. [p496]

J. E. Hagan, P. Moraga, F. Costa, N. Capian, G. S. Ribeiro, E. A. Wunder Jr, R. D. Felzemburgh, R. B. Reis, N. Nery, F. S. Santana, D. Fraga, B. L. dos Santos, A. C. Santos, Q. A., W. Tassinari, M. S. Carvalho, M. G. Reis, P. J. Diggle, and A. I. Ko. Spatio-Temporal Determinants of Urban Leptospirosis Transmission: Four-Year Prospective Cohort Study of Slum Residents in Brazil. *Public Library of Science: Neglected Tropical Diseases*, 10(1):e0004275, 2016. URL https://doi.org/10.1371/journal.pntd.0004275. [p495]

A. Y. Kim and J. Wakefield. *SpatialEpi: Methods and Data for Spatial Epidemiology*, 2016. URL https://CRAN.R-project.org/package=SpatialEpi. R package version 1.2.2. [p497]

A. B. Lawson. *Bayesian Disease Mapping: Hierarchical Modeling In Spatial Epidemiology*. Chapman and Hall/CRC, Boca Raton, Fl., 2009. [p496]

B. G. Leroux, X. Lei, and N. Breslow. Estimation of disease rates in small areas: A new mixed model for spatial dependence. In M. E. Halloran and D. Berry, editors, *Statistical Models in Epidemiology, the Environment, and Clinical Trials*, The IMA Volumes in Mathematics and its Applications, pages 179–191. Springer-Verlag, New York, NY, 2000. [p495]

P. Moraga. *SpatialEpiApp: A Shiny Web Application for the Analysis of Spatial and Spatio-Temporal Disease Data*, 2017a. URL https://CRAN.R-project.org/package=SpatialEpiApp. R package version 0.3. [p504]

P. Moraga. SpatialEpiApp: A Shiny Web Application for the Analysis of Spatial and Spatio-Temporal Disease Data. *Spatial and Spatio-temporal Epidemiology*, 23:47–57, 2017b. URL https://doi.org/10.1016/j.sste.2017.08.001. [p504]

P. Moraga and M. Kulldorff. Detection of Spatial Variations in Temporal Trends with a Quadratic Function. *Statistical Methods for Medical Research*, 25(4):1422–1437, 2016. URL https://doi.org/10.1177/0962280213485312. [p495]

P. Moraga and A. Ozonoff. Model-Based Imputation of Missing Data from the 122 Cities Mortality Reporting System (122 CMRS). *Stochastic Environmental Research and Risk Assessment*, 29(5):1499–1507, 2013. URL https://doi.org/10.1007/s00477-014-0974-4. [p495]

P. Moraga, J. Cano, R. F. Baggaley, J. O. Gyapong, S. Njenga, B. Nikolay, E. Davies, M. P. Rebollo, R. L. Pullan, M. J. Bockarie, D. Hollingsworth, M. Gambhir, and S. J. Brooker. Modelling the Distribution and Transmission Intensity of Lymphatic Filariasis in Sub-Saharan Africa Prior to Scaling up Interventions: Integrated Use of Geostatistical and Mathematical Modelling. *Public Library of Science: Neglected Tropical Diseases*, 8:560, 2015. URL https://doi.org/10.1186/s13071-015-1166-x. [p495]

P. Moraga, S. Cramb, K. Mengersen, and M. Pagano. A Geostatistical Model for Combined Analysis of Point-Level and Area-Level Data Using INLA and SPDE. *Spatial Statistics*, 21:27–41, 2017. URL https://doi.org/10.1016/j.spasta.2017.04.006. [p504]

W. S. Robinson. Ecological Correlations and the Behavior of Individuals. *American Sociological Review*, 15(3):351–357, 1950. URL https://doi.org/10.2307/2087176. [p504]

H. Rue, S. Martino, and N. Chopin. Approximate Bayesian inference for latent Gaussian models using integrated nested Laplace approximations (with discussion). *Journal of the Royal Statistical Society B*, 71:319–392, 2009. URL https://doi.org/10.1111/j.1467-9868.2008.00700.x. [p495, 496]

H. Rue, S. Martino, F. Lindgren, D. Simpson, A. Riebler, E. T. Krainski, and G.-A. Fuglstad. *INLA: Functions Which Allow to Perform Full Bayesian Analysis of Latent Gaussian Models Using Integrated Nested Laplace Approximations*, 2017. R package version 17.06.20. [p495]

D. Vanderkam, J. Allaire, J. Owen, D. Gromer, P. Shevtsov, and B. Thieurmel. *Dygraphs: Interface to 'Dygraphs' Interactive Time Series Charting Library*, 2017. URL https://CRAN.R-project.org/package=dygraphs. R package version 1.1.1.4. [p504]

H. Wickham and W. Chang. *Ggplot2: Create Elegant Data Visualisations Using the Grammar of Graphics*, 2016. URL https://CRAN.R-project.org/package=ggplot2. R package version 2.2.1. [p502]

Y. Xie. *DT: A Wrapper of the JavaScript Library 'DataTables'*, 2016. URL https://CRAN.R-project.org/package=DT. R package version 0.2. [p504]

Paula Moraga
Centre for Health Informatics, Computing and Statistics (CHICAS)
Lancaster Medical School
Lancaster University
Lancaster, LA1 4YW
United Kingdom
ORCiD: 0000-0001-5266-0201
Webpage: https://paula-moraga.github.io/
p.e.moraga-serrano@lancaster.ac.uk

SetMethods: an Add-on R Package for Advanced QCA

by Ioana-Elena Oana and Carsten Q. Schneider

Abstract This article presents the functionalities of the R package **SetMethods**, aimed at performing advanced set-theoretic analyses. This includes functions for performing set-theoretic multi-method research, set-theoretic theory evaluation, Enhanced Standard Analysis, diagnosing the impact of temporal, spatial, or substantive clusterings of the data on the results obtained via Qualitative Comparative Analysis (QCA), indirect calibration, and visualising QCA results via XY plots or radar charts. Each functionality is presented in turn, the conceptual idea and the logic behind the procedure being first summarized, and afterwards illustrated with data from Schneider et al. (2010).

Introduction

Set-theoretic methods, in general (Goertz and Mahoney, 2012), and Qualitative Comparative Analysis, in particular, are becoming increasingly popular within different disciplines in the social sciences and neighboring fields (Rihoux et al., 2013). Parallel to conceptual developments and increasing numbers of applied studies, accelerating progress in terms of software development can be witnessed. While less than a decade ago only two functioning software packages were available to users (*fsQCA* Ragin et al. (2006) and *Tosmana* Cronqvist (2011)), there are now over a dozen different software solutions offered (see `http://compasss.org/software.htm`). Many of them are developed within the R software environment, with R package **QCA** (Dusa, 2007) being not only the one with the longest history, but also the most complete and complex.

In this paper, we discuss the different functionalities of the R package **SetMethods** (Medzihorsky et al., 2016). It is best perceived of as an add-on tool to package **QCA** and allows applied researchers to perform advanced set-theoretic analyses. More precisely, **SetMethods** enables researchers to perform Set-Theoretic Multi-Method Research, the Enhanced Standard Analysis (ESA), Set-Analytic Theory Evaluation, to run diagnostics in the presence of clustered data structures, and to display their results in various ways.

We proceed as follows. Each of the different functionalities within **SetMethods** is presented in a separate section. Within each section, we first briefly summarize the conceptual idea behind the analysis in question, then describe the computational logic of the function for performing the analysis, after which we demonstrate the use of the function by displaying the R syntax and selected output by using an example from published research.

Even though the main purpose is to present the functionality of R package **SetMethods**, this article is also useful for researchers who perform their QCA in software environments other than R because we present the logic of several of the main advanced set-analytic procedures in a concise and transparent manner.

The empirical example

In order to illustrate the use of the different functions in **SetMethods**, we use the empirical example by Schneider et al. (2010) which uses fuzzy-sets for explaining capitalist variety and export performance in high-tech industries. More precisely, the research question focuses on the institutional determinants of export performance in high-tech industries. The outcome consists of the export performance in high-tech industries (EXPORT). The conditions used are: employment protection (EMP), collective bargaining (BARGAIN), university training (UNI), occupational training (OCCUP), stock market size (STOCK), and mergers and aquisitions (MA). The authors analyze 76 cases, representing 19 countries at four time points.

For the sake of simplicity, we use the same data for illustrating all the functions. Our goal is not, of course, to contribute to the substantive discussion on varieties of capitalism or institutional context. This is why we will take the liberty to alter the analytic setup if needed for demonstration purposes, by, for instance, dropping cases or conditions or by changing the outcome to be explained.

```
# We load the SetMethods package:
```

```
library(SetMethods)

# First rows of the Schneider et al.(2010) data called SCHF from package
# SetMethods:

data(SCHF)
head(SCHF)
```

```
##               EMP BARGAIN  UNI OCCUP STOCK   MA EXPORT
## Australia_90 0.07    0.90 1.00  0.68  0.45 0.33   0.19
## Austria_90   0.70    0.98 0.01  0.91  0.01 0.05   0.25
## Belgium_90   0.94    0.95 0.14  0.37  0.26 0.14   0.14
## Canada_90    0.04    0.21 0.99  0.11  0.62 0.31   0.28
## Denmark_90   0.59    0.78 0.10  0.55  0.53 0.10   0.34
## Finland_90   0.70    0.97 0.20  0.95  0.02 0.13   0.17
```

Set-theoretic multi-method research (MMR)

The term Set-Theoretic Multi-Method Research (MMR) captures all those empirical approaches for combining cross-case analyses with within-case studies in which both levels of analysis follow the goal of investigating sets and their relations. At the cross-case level the most common methodological tool is Qualitative Comparative Analysis (QCA) and at the within-case level, process tracing. Both tools can be rooted in (fuzzy) set theory (see, e.g. Ragin (2008) for QCA and Mikkelsen (2017) for process tracing). In principle and practice any sequence of analyses can and is performed in the applied literature. In the following, we focus on the sequence 'cross-case QCA first, followed by within-case analyses'. This sequence may or may not be continued by another QCA. As any decent QCA, it is certainly preceded by a thorough accumulation of case knowledge in order to select and calibrate conditions. These crucial research steps, however, fall outside of the definitional scope of set-theoretic MMR.[1]

In the following, we limit our discussion to set-theoretic MMR after a cross-case analysis of sufficiency, as discussed in Schneider and Rohlfing (2013, 2016), Schneider and Rohlfing (manuscript), and Rohlfing and Schneider (2018).[2] We first briefly summarize the different types of cases and the purpose of their within-case analysis (Section Identifying types of cases). After this, we discuss the four different feasible comparative within-case analyses in Figure 1 and their analytic purposes (Section Identifying best-matching pairs of cases for comparative process tracing). For each form of within-case analysis, we explain the use of the `mmr` function and the formula used for finding cases (Schneider and Rohlfing, manuscript) [3]. The general structure of function `mmr` is illustrated in Figure 1. Users need to specify whether they want to perform single or comparative within-case analysis and then on which cases the analysis is performed.

Identifying types of cases

Key for combining QCA with process tracing is the sorting of cases to different case types based on the QCA solution formula. The literature identifies five different types (Schneider and Rohlfing, 2013). Membership in a type is defined by the membership scores of a case in the outcome Y, on the one hand, and the sufficient term T or the solution formula S, on the other hand. Table 1 summarizes the definition of each case type and the analytic purpose of the within-case analysis in single cases. Figure 2 visualizes the location of each case type in an XY plot.

Typical cases and deviant cases consistency are defined based on their membership in a sufficient term T, whereas deviant cases coverage and IIR cases are defined based on their membership solution formula S. Deviant cases consistency are subdivided into deviant in degree and deviant in kind. The latter are always preferable for within-case analysis. IIR cases are not useful for single-case studies, but they play an important role for comparative within-case analyses (see Section Identifying best-matching pairs of cases for comparative process tracing).

Table 1 is adapted from Schneider and Rohlfing (manuscript).

[1] For a systematic discussion of the pre-QCA case studies, see Rihoux and Lobe (2009).

[2] For MMR after an analysis of necessity, see Rohlfing and Schneider (2013).

[3] For a systematic test of the mathematical formulas used for selecting single cases or pairs of cases for set-theoretic MMR see the Appendix of this paper.

Figure 1: Types of Post-QCA Case Studies

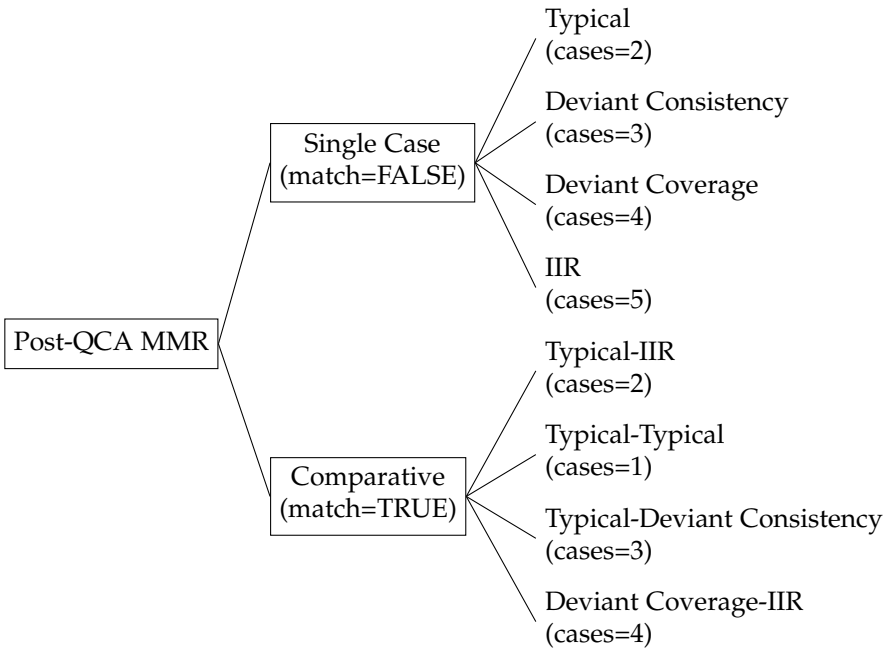
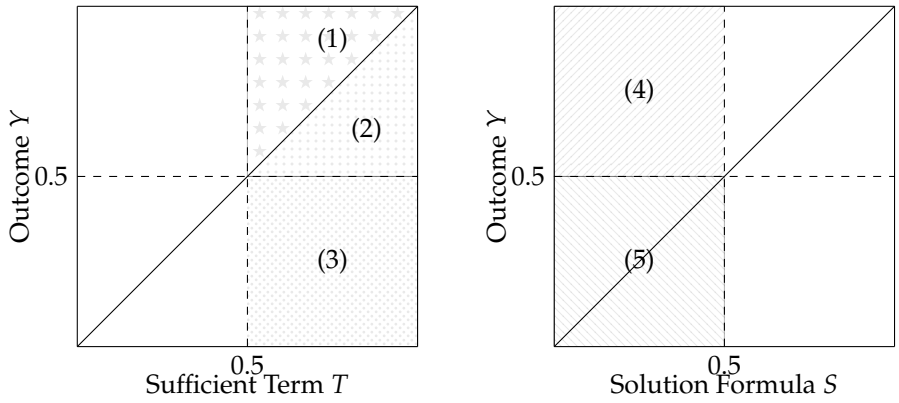


Table 1: Types of cases in fsQCA of sufficiency

Type of case	Membership in T	Y		Goal of within-case analysis
(1) Typical	$>.5$	$>.5$	$T \leq Y$	identify mechanism M
(2) Deviant consistency (degree)	$>.5$	$>.5$	$T \geq Y$	not recommended
(3) Deviant consistency (kind)	$>.5$	$<.5$		identify missing INUS
	S			
(4) Deviant coverage	$<.5$	$>.5$		identify missing conjunction
(5) IIR	$<.5$	$<.5$		not useful

Figure 2: Types of Cases, XY Plot



Typical cases

Process tracing in typical cases aims at empirically probing the causal mechanism(s) linking the sufficient term S to outcome Y. For conjunction S to be causal, each conjunct C of S must be causal, i.e. they must make a difference to outcome Y by making a difference to mechanism M. This requires as many within-case analyses of typical cases as there are conjuncts in the sufficient conjunction. For each analysis, one is the focal conjunct FC and the others are the complementary conjuncts CC. The focal conjunct FC is the conjunct for which we want to find out whether it makes a difference for the mechanism M, while the complementary conjuncts CC represent the other conjuncts of the sufficient term S (Schneider and Rohlfing, manuscript). For causal inference on the configuration we proceed by taking each conjunct at a time as the focal conjunct FC. Additionally, we also apply the *test severity principle*. With fuzzy-sets the membership in mechanism M can only vary within the corridor established by the membership in FC (the lowest value M can take) and Y (the highest value M can take) for preserving the causal chain $FC \rightarrow M \rightarrow Y$ (Schneider and Rohlfing, manuscript). The smaller the corridor, the smaller the range of membership values M can take. Therefore, the most *severe test* for M is the one in which $FC = S = Y$ because the only consistent membership score in M equals $FC = S = Y$.

The *best-available* typical case fulfills the following criteria: a) the focal conjunct is the one that defines the membership of the typical case in the term ($FC \leq CC$); b) the corridor for mechanism M as defined by the sufficient term S (from a) we also have $S = FC$) and Y is small; c) membership in the sufficient term S is high; d) the case is uniquely covered by the sufficient term S.

Figure 3 visualizes the test severity principle in two different ways. The XY plot in the upper panel shows that for cases closer to the diagonal, test severity increases. The length of the vertical and horizontal arrows, respectively, visualizes the range of fuzzy set membership scores for M that would still be consistent. The larger this range, the less severe the test. The Euler diagram in the lower panel visualizes the same by contrasting S_1 almost as big as Y with S_2 being much smaller than Y. The former leaves little and the latter a lot of room for M.

The ideal typical case is located in the upper-right corner of the XY plot in Figure 3 with $FC = S = Y = 1$. In applied QCA, such cases usually do not exist in the data at hand. Function `mmr()` identifies the best available typical case in a given data set.

Function `mmr()` first sorts each typical case based on whether $FC \leq CC$ (rank 1) or $FC > FC$ (rank 2). Cases in each rank are then further sorted according to Formula 1. Smaller values indicate better suitable cases.[4]

$$\begin{aligned} TYP = \ & (Y - S) && \text{small corridor for mechanism} \\ & + (1 - S) && \text{large membership in the sufficient term} \end{aligned} \tag{1}$$

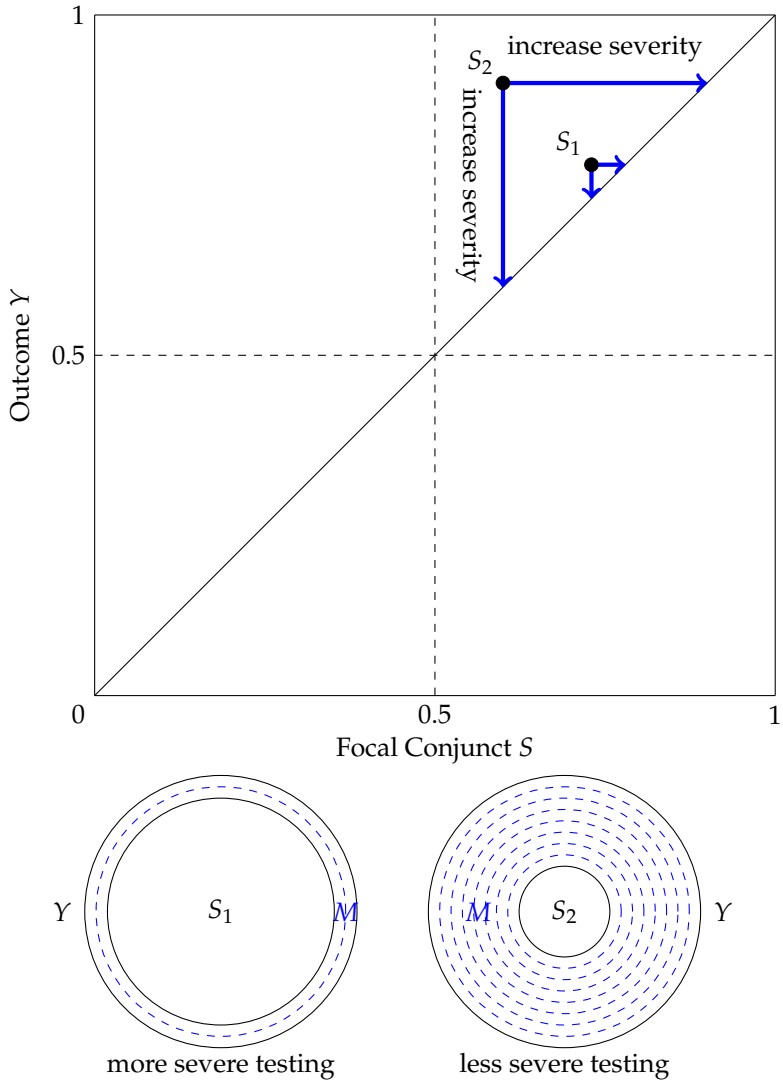
where Y = outcome, S = sufficient term

Applied to our example, function `mmr()` works as follows. After minimizing the truth table TT_y and producing the parsimonious and intermediate solutions sol_yp and sol_yi using package **QCA**, we input these solutions [5] into the `mmr()` function while setting arguments `match` to `FALSE` and `cases`

[4] This holds for all `mmr()` formulas: the smaller the value, the more suitable the case (pair) is.

[5] For inputing solutions that have model ambiguity, argument `sol` can be used to specify which solution the user wants to work with. If a single number is used, this number indicates which model of the conservative or

Figure 3: Two visualizations of test severity



to 2. As argument `term` is set to 1 the output shows the typical cases for each focal condition in the first sufficient term, together with some additional information. The information included in the output comprises of membership values of the typical cases in the focal conjuct, complementary conjuncts, the whole sufficient term, and the outcome (in this case *EXPORT*), formula values *St*, whether the case is the most typical according to the formula, which rank does the case sit in, and whether the case is uniquely covered by the sufficient term. The order of the information that users should look for in this output is whether the case is uniquely covered, what rank is the case in (the smaller, the better), and what formula value *St* does the case have (the smaller, the better). For example, for focal conjunct *emp* in sufficient term $emp * bargain * OCCUP$, `Switzerland_03` appears to be the *best available* typical case, being uniquely covered, being in Rank 1, and having the smallest formula value (St=0.59).

```
# We create the truth table:

TT_y <- truthTable(SCHF, outcome = "EXPORT",
                   conditions = c("EMP","BARGAIN","UNI",
                                  "OCCUP","STOCK", "MA"),
```

parsimonious solution according to the order in the "qca" object the user wants to work with. However, since QCA solutions (conservative, parsimonious, intermediate) are in a subset relationship with each other, they tend to have more complicated structures in which model ambiguity is tied from one solution to the other. For this cases the argument `sol` allows users to specify the models they want to choose by using a character string of the form "c1p3i2" where c = conservative solution, p = parsimonious solution and i = intermediate solution.

```
                  incl.cut = .9,
                  complete = TRUE,
                  PRI = TRUE,
                  sort.by = c("out", "incl", "n"))

# Get the parsimonious solution:

sol_yp <- minimize(TT_y, include = "?", details = TRUE,
                  show.cases = TRUE)

# Get the intermediate solution:

sol_yi <- minimize(TT_y, include = "?", details = TRUE,
                  show.cases = TRUE, dir.exp = c(0,0,0,0,0,0))

# Get typical cases for the first term of the second intermediate solution:

mmr (results = sol_yi, outcome = "EXPORT", neg.out = FALSE,
     sol = "c1p1i2", match = FALSE, cases = 2, term = 1)


## Typical Cases - Focal Conjunct emp :
## ----------
##                Focal Conjunct Comp. Conjunct Term Membership EXPORT   St
## Switzerland_03           0.70           0.71            0.70   0.99 0.59
## Switzerland_99           0.75           0.54            0.54   0.98 0.69
##                most_typical Rank uniquely_cov
## Switzerland_03         TRUE    1         TRUE
## Switzerland_99        FALSE    2         TRUE
##
## Typical Cases - Focal Conjunct bargain :
## ----------
##                Focal Conjunct Comp. Conjunct Term Membership EXPORT   St
## Switzerland_99           0.54           0.74            0.54   0.98 0.90
## Switzerland_03           0.76           0.70            0.70   0.99 0.53
##                most_typical Rank uniquely_cov
## Switzerland_99        FALSE    1         TRUE
## Switzerland_03         TRUE    2         TRUE
##
## Typical Cases - Focal Conjunct OCCUP :
## ----------
##                Focal Conjunct Comp. Conjunct Term Membership EXPORT   St
## Switzerland_03           0.71           0.70            0.70   0.99 0.58
## Switzerland_99           0.74           0.54            0.54   0.98 0.70
##                most_typical Rank uniquely_cov
## Switzerland_03         TRUE    2         TRUE
## Switzerland_99        FALSE    2         TRUE
```

Deviant cases consistency

Deviant cases consistency are puzzling because their membership in the sufficient term S exceeds that in the outcome Y, i.e. $S > Y$. This becomes even more puzzling if $S > 0.5 \& Y < 0.5$, that is, if we have deviant cases consistency in kind rather than just in degree (see Table 1). The more S exceeds Y, the bigger the empirical puzzle, especially if membership in S is high. Within-case analysis of a deviant case consistency aims at identifying the reasons why mechanism M either absent or prevented from producing Y. The reason must be an INUS condition omitted from S. Formula 2 identifies the best available deviant case consistency in a data set.

$$\begin{aligned} DCN = \ & [1 - (S - Y)] && \text{far from to the diagonal} \\ & + (1 - S) && \text{large membership in the sufficient term} \end{aligned} \tag{2}$$

where Y = outcome, S = sufficient term

Using the same data from Schneider et al. (2010) and focusing on the parsimonious solution, function `mmr()` identifies the deviant consistency cases for each sufficient term. For obtaining this we need to keep argument `match` set to `FALSE`, as we are doing single case identification, but set argument `cases` to `3`, the identifier for deviant cases consistency (see Figure 1). The output shows the deviant consistency cases (first column) grouped by sufficient term (second column) together with term membership, outcome membership, formula value *Sd*, and whether the case is the most deviant for a particular term. In the output we see that, for example, for term $emp * OCCUP$ the most deviant case consistency is `Switzerland_90` with the smallest formula value (Sd=0.67). Figure 4 shows all the deviant cases consistency (cases in the lower right corner) for the first sufficient path $emp * OCCUP$ of the parsimonious solution.

```
# Get deviant cases consistency for the parsimonious solution:

mmr (results = sol_yp, outcome = "EXPORT", neg.out = FALSE,
     sol = 1, match = FALSE, cases = 3)

## Deviant Consistency Cases :
## ----------
##             cases              term term_membership EXPORT   Sd
## 2  Switzerland_90         emp*OCCUP            0.82   0.31 0.67
## 1    Australia_90         emp*OCCUP            0.68   0.19 0.83
## 3    Australia_95         emp*OCCUP            0.68   0.31 0.95
## 4    Australia_99         emp*OCCUP            0.68   0.38 1.02
## 14   Australia_95 BARGAIN*UNI*STOCK            0.90   0.31 0.51
## 7    Australia_03 BARGAIN*UNI*STOCK            0.90   0.35 0.55
## 6        Spain_99 BARGAIN*UNI*STOCK            0.79   0.27 0.69
## 8       Norway_03 BARGAIN*UNI*STOCK            0.79   0.32 0.74
## 41   Australia_99 BARGAIN*UNI*STOCK            0.79   0.38 0.80
## 21     Denmark_95 BARGAIN*UNI*STOCK            0.76   0.40 0.88
## 5      Belgium_99 BARGAIN*UNI*STOCK            0.72   0.40 0.96
## 31     Finland_95 BARGAIN*UNI*STOCK            0.73   0.49 1.03
## 42       Spain_03    occup*STOCK*ma            0.74   0.30 0.82
## 32     Denmark_95    occup*STOCK*ma            0.73   0.40 0.94
## 15      Canada_90    occup*STOCK*ma            0.62   0.28 1.04
## 22      Canada_95    occup*STOCK*ma            0.60   0.30 1.10
##    most_deviant
## 2          TRUE
## 1         FALSE
## 3         FALSE
## 4         FALSE
## 14         TRUE
## 7         FALSE
## 6         FALSE
## 8         FALSE
## 41        FALSE
## 21        FALSE
## 5         FALSE
## 31        FALSE
## 42         TRUE
## 32        FALSE
## 15        FALSE
## 22        FALSE


# Plot each sufficient path of the parsimonious solution:

pimplot(data = SCHF, results = sol_yp, outcome = "EXPORT", case_labels = FALSE)
```

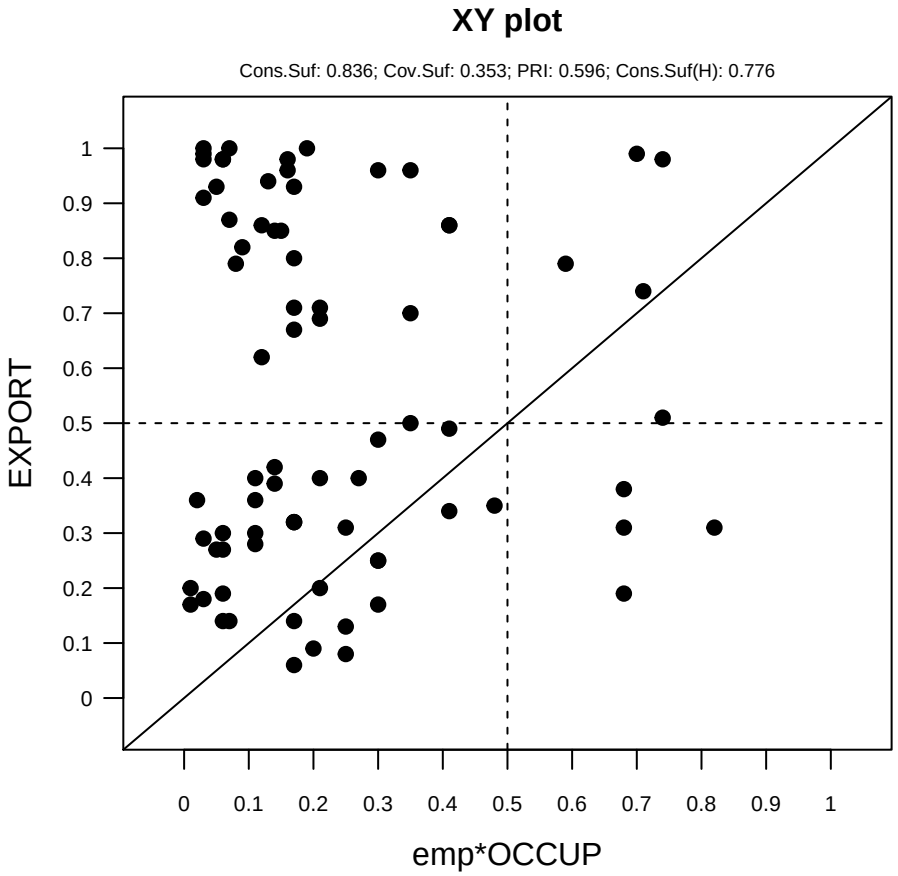


Figure 4: Deviant Cases Consistency

Deviant cases coverage

Deviant cases coverage are puzzling because they are members of the outcome without, however, being members of any known sufficient term. Within-case analysis aims at identifying sufficient term S_+ omitted from the solution formula, which triggers mechanism a M and outcome Y.

Since deviant cases coverage are defined by what they are *not* - members of the solution formula (see Table 1) - this solution formula is not a good place to start selecting the best available deviant cases coverage. Instead, this type of case is selected based on their membership in their truth table row TT. For each TT with at least one deviant case coverage, a within-case analysis can be performed. If more than one deviant case coverage populates the same TT, Formula 3 identifies the best available case for within-case analysis.

$$\begin{aligned} DCV = & \ |Y - TT| && \text{small corridor for mechanism} \\ & + (1 - TT) && \text{large membership in the truth table row} \end{aligned} \tag{3}$$

where Y = outcome, TT = membership in the Truth Table row

Similar to the Formula 1 for identifying the best available typical case, the goal is to minimize the difference between the membership scores in Y and TT and to prefer higher membership in TT. This is achieved by formula 3. Since the primary goal in this within-case analysis is not to draw causal inference but to identify a missing conjunction, there is no need to decompose TT into its constituent sets.

Applied to our example, the following code displays the list of deviant cases coverage (notice argument `cases` is set to 4), the membership they have in the entire solution formula, their values on the Formula 3, the truth table row they belong to (columns starting with TT indicating the specific combination of conditions the case presents), the membership they have in that specific truth table row, and membership in the outcome. The cases are sorted by truth table row and ranked according to their appropriateness using formula values Sd. For example, we can notice that truth table row

$emp * bargain * UNI * occup * STOCK * MA$ (rows 9, 10, 4, and 5 of the output) is populated by 4 deviant coverage cases, out of which `UK_90` is the best available for within case analysis, having the smallest formula value (Sd=0.51).

```
# Get deviant cases coverage for the parsimonious solution:

mmr (results = sol_yp, outcome = "EXPORT", neg.out = FALSE,
     sol = 1, match = FALSE, cases = 4)

## Deviant Coverage Cases :
## ----------
##              case solution_membership   Sd TT_EMP TT_BARGAIN TT_UNI
## 8           UK_03                0.12 0.22      0          0      1
## 3      Germany_99                0.17 0.33      1          1      0
## 11          UK_99                0.18 0.34      0          0      1
## 12         USA_99                0.20 0.39      0          0      1
## 7       Sweden_95                0.36 0.47      1          1      0
## 1       France_95                0.41 0.50      1          1      0
## 9           UK_90                0.36 0.51      0          0      1
## 10          UK_95                0.36 0.59      0          0      1
## 4      Ireland_03                0.32 0.64      0          0      1
## 5      Ireland_99                0.32 0.64      0          0      1
## 2      Germany_03                0.40 0.67      1          1      0
## 6  Netherlands_95                0.45 0.68      1          1      0
##    TT_OCCUP TT_STOCK TT_MA TT_row_membership EXPORT
## 8         0        1     1               0.88   0.98
## 3         1        1     1               0.71   0.67
## 11        0        1     1               0.82   0.98
## 12        0        1     1               0.80   0.99
## 7         1        1     1               0.62   0.71
## 1         1        0     0               0.56   0.62
## 9         0        1     1               0.64   0.79
## 10        0        1     1               0.64   0.87
## 4         0        1     1               0.68   1.00
## 5         0        1     1               0.68   1.00
## 2         1        0     0               0.51   0.69
## 6         1        1     1               0.51   0.70
```

Individually irrelevant cases

Individually irrelevant (IIR) cases owe their name to the fact that single within-case analyses in this type of cases is not useful. IIR cases do play a crucial role in two forms of comparative within-case analysis (see Section Identifying best-matching pairs of cases for comparative process tracing). Even if not useful for single case studies, identifying IIR cases is informative as their list - together with the deviant cases coverage - indicate the diversity among cases without the outcome. The more different truth table rows are populated by IIR cases (and deviant cases coverage), the more heterogeneous this group of cases is.

Function `mmr()` lists all individually irrelevant cases with respect to the entire solution formula (also called globally uncovered IIR cases) and sorts each of them into the truth table to which they belong best. Since these cases are not informative for single case studies and are being used just to indicate diversity among the cases without the outcome, the function does not involve a formula ranking of IIR cases.

```
# Get individually irrelevant cases for the parsimonious solution:

mmr (results = sol_yp, outcome = "EXPORT", neg.out = FALSE,
     sol = 1, match = FALSE, cases = 5)

## Individually Irrelevant Cases :
## ----------
##              case solution_membership TT_EMP TT_BARGAIN TT_UNI TT_OCCUP
```

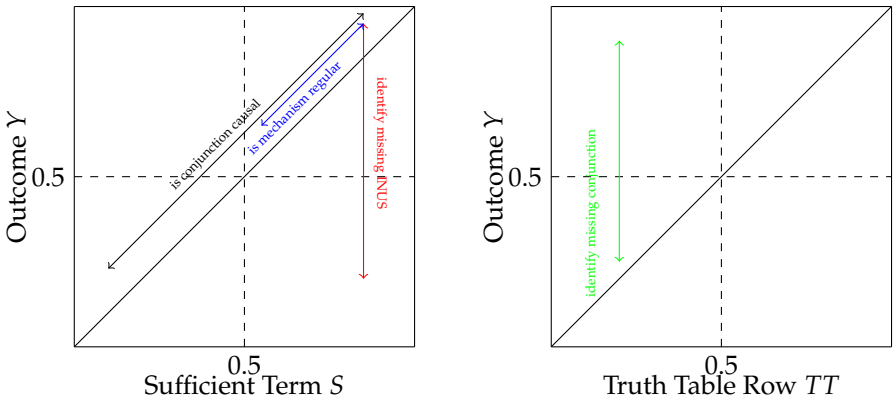
```
## 19 New Zealand_90               0.17      0          0      0        0
## 18 New Zealand_03               0.25      0          0      1        0
## 21 New Zealand_99               0.25      0          0      1        0
## 6       Canada_03               0.38      0          0      1        0
## 7       Canada_99               0.20      0          0      1        0
## 20 New Zealand_95               0.20      0          0      1        0
## 4      Belgium_90               0.26      1          1      0        0
## 10      France_90               0.23      1          1      0        0
## 14       Italy_90               0.01      1          1      0        0
## 25       Spain_90               0.03      1          1      0        0
## 1      Austria_90               0.30      1          1      0        1
## 2      Austria_95               0.30      1          1      0        1
## 3      Austria_99               0.30      1          1      0        1
## 9      Finland_90               0.30      1          1      0        1
## 11     Germany_90               0.05      1          1      0        1
## 12     Germany_95               0.17      1          1      0        1
## 13       Italy_03               0.25      1          1      0        1
## 15       Italy_95               0.03      1          1      0        1
## 17 Netherlands_90               0.14      1          1      0        1
## 8      Denmark_90               0.45      1          1      0        1
## 16       Italy_99               0.38      1          1      0        1
## 5      Belgium_95               0.21      1          1      0        1
## 27      Sweden_90               0.07      1          1      0        1
## 26       Spain_95               0.06      1          1      1        0
## 22      Norway_90               0.12      1          1      1        1
## 23      Norway_95               0.49      1          1      1        1
## 24      Norway_99               0.45      1          1      1        1
##    TT_STOCK TT_MA TT_row_membership EXPORT
## 19        0     1               0.58   0.06
## 18        0     1               0.75   0.13
## 21        0     1               0.75   0.08
## 6         1     1               0.62   0.36
## 7         1     1               0.80   0.40
## 20        1     1               0.76   0.09
## 4         0     0               0.63   0.14
## 10        0     0               0.54   0.42
## 14        0     0               0.70   0.20
## 25        0     0               0.69   0.17
## 1         0     0               0.70   0.25
## 2         0     0               0.70   0.25
## 3         0     0               0.70   0.47
## 9         0     0               0.70   0.17
## 11        0     0               0.92   0.27
## 12        0     0               0.74   0.32
## 13        0     0               0.67   0.31
## 15        0     0               0.71   0.18
## 17        0     0               0.57   0.39
## 8         1     0               0.53   0.34
## 16        1     0               0.62   0.29
## 5         1     1               0.53   0.20
## 27        1     1               0.74   0.36
## 26        0     0               0.84   0.19
## 22        0     0               0.65   0.14
## 23        0     0               0.51   0.14
## 24        0     1               0.55   0.32
```

Identifying best-matching pairs of cases for comparative process tracing

The literature identifies four feasible within-case comparisons after a QCA between two types of cases each (Schneider and Rohlfing, manuscript). With each comparison a different analytic goal is pursued. Figure 5 summarizes these goals. The two comparisons 'along the main diagonal' pursue a causal inference goal, whereas the two 'vertical' comparisons aim at improving the QCA model specification by identifying either an INUS condition missing from a known sufficient term or an

entire new sufficient term missing from the solution formula.

Figure 5: Forms of feasible comparisons



Matching typical and individually irrelevant (IIR) cases

The purpose of the within-case comparison between a typical case and and IIR case is to empirically investigate whether a sufficient term is a difference-maker, i.e. causal, not only for the outcome (Y) at the cross-case level, but also for the mechanism M at the within-case level. Similarly, the within-case comparison of two typical cases empirically probes whether the same mechanism M links the sufficient term S to outcome Y in typical cases that are as different from each other as possible. For both forms of comparison, it holds that if S is a conjunction, each of its conjuncts C must be a difference-maker. Hence, the comparisons between a typical case and an IIR case (or another typical case) must be performed for each single conjunct C at a time. The following sections provide more details for each form of comparison and spells out the sorting mechanisms and mathematical formulas that underly the respective functions in **SetMethods**.

Table 2: Possible membership constellations between focal (FC) and complementary conjuncts (CC) in comparison of typical and IIR case

Rank	Typical	IIR	Difference FC	Deter-minate	Attribution typical	Attribution IIR
1	$FC \leq CC$	$FC < 0.5 < CC$	Yes	Yes	Yes	Yes
2	$FC > CC$	$FC < 0.5 < CC$	Yes	Yes	No	Yes
3	$FC \leq CC$	$FC \leq CC < 0.5$	Yes	No	Yes	Yes
4	$FC \leq CC$	$CC < FC < 0.5$	Yes	No	Yes	No
4	$FC > CC$	$FC \leq CC < 0.5$	Yes	No	No	Yes
6	$FC > CC$	$CC < FC < 0.5$	Yes	No	No	No
7	$FC \leq CC$	$CC < 0.5 < FC$	No	No	Yes	No
8	$FC > CC$	$CC < 0.5 < FC$	No	No	No	No

taken from Schneider and Rohlfing (manuscript)

Function `mmr()` first sorts each pair of typical and IIR cases into ranks 1-8 as defined in Table 2. Cases in smaller rank numbers are more adequate for the analytic goal of the comparative within-case analysis of these two case types. For case pairs in rank 1, for example, it holds that the difference-making quality can be attributed to the focal conjunct FC both on the typical and the IIR case, and that it is determinate.

Within each rank, Formula 4 maximizes the following criteria: between both cases, the difference in FC and in Y, respectively, should be small; both should have high membership in CC; and both should be close to the diagonal. Within each rank, case pairs with smaller formula values are more appropriate. Additionally, typical cases should be uniquely covered by the sufficient term under inverstigation, while IIR cases should be globally uncovered (not covered by any of the sufficient

terms).

$$
\begin{aligned}
TYP - IIR = \; & [1 - (FC_{TYP} - FC_{IIR})] && \text{large difference in focal condition} \\
& + [1 - (Y_{TYP} - Y_{IIR})] && \text{large difference in outcome} \\
& + |min(CC_{TYP}) - min(CC_{IIR})| && \text{small difference in complementary conditions} \\
& + 2 * |(Y_{TYP} - min\,(FC_{TYP}, CC_{TYP}))| && \text{typical case close to diagonal} \\
& + 2 * |(Y_{IIR} - min\,(FC_{IIR}, CC_{IIR}))| && \text{IIR case close to diagonal}
\end{aligned} \tag{4}
$$

where Y = outcome, FC = focal condition, and CC = complementary condition

For switching to comparative MMR and identifying pairs of cases, argument `match` must be set to `TRUE`. Additionally, for getting the best available pairs of typical and IIR cases we set `cases` to `2`. In the output for the first sufficient term (notice argument `term` set to `1`) we will get the best available pairs for each focal conjunct in turn as separate tables. The output lists the names of the typical and IIR case, their value on the above formula (*Distance*), which rank does the pair come from, whether the typical case is uniquely covered, and whether the IIR case is globally uncovered. Researchers should strive to pick cases that are uniquely covered and globally uncovered, have the smallest rank possible, and have the smallest *Distance* value. For example, for focal conjunct *OCCUP*, typical case `Denmark_99` and IIR case `New Zealand_90` are the best pair available as they are in Rank 1, they have the smallest formula value (Distance=1.24), and the typical case is uniquely covered by the term, while the IIR case is globally uncovered by the solution.

```
# Get matching pairs of typical and IIR cases for the first term
# of the parsimonious solution:

mmr (results = sol_yp, outcome = "EXPORT", neg.out = FALSE,
     sol = 1, match = TRUE, cases = 2, term = 1)

## Focal Conjunct emp :
## ----------
##           Typical        IIR Distance PairRank UniqCovTyp GlobUncovIIR
## 44 Switzerland_03  Norway_90     1.30        1       TRUE         TRUE
## 43     Denmark_03  Norway_90     1.38        1       TRUE         TRUE
## 88 Switzerland_03  Norway_95     1.38        1       TRUE         TRUE
## 80 Switzerland_03   Italy_95     1.40        1       TRUE         TRUE
## 60 Switzerland_03 Belgium_95     1.55        1       TRUE         TRUE
##
## Focal Conjunct OCCUP :
## ----------
##            Typical            IIR Distance PairRank UniqCovTyp
## 37      Denmark_99 New Zealand_90     1.24        1       TRUE
## 81      Denmark_99 New Zealand_95     1.30        1       TRUE
## 133     Denmark_99 New Zealand_03     1.31        1       TRUE
## 38  Switzerland_99 New Zealand_90     1.39        1       TRUE
## 82  Switzerland_99 New Zealand_95     1.45        1       TRUE
##     GlobUncovIIR
## 37          TRUE
## 81          TRUE
## 133         TRUE
## 38          TRUE
## 82          TRUE
```

Matching two typical cases

The matching of two typical cases follows a logic similar to the one between a typical and an IIR case. The goal is to probe the difference-making properties of each conjunct (FC) in sufficient term S to mechanism M. Table 3 defines the four ranks that can occur based on two typical cases' membership in FC and the complementary conditions CC. After sorting each possible pair of typical cases into one of these ranks, Formula 5 further ranks those pairs such that their difference in FC and the outcome, respectively, is minimized; that their membership in CC is maximized; and that both are close to the

diagonal (test severity principle). Additionally, the two typical cases should be uniquely covered by the sufficient term.

Table 3: Possible membership constellations between focal (*FC*) and complementary conjuncts (*CC*) in comparison of two typical cases

Rank	Typical 1	Typical 2	Attribution typical 1	Attribution typical 2
1	$FC \leq CC$	$FC \leq CC$	Yes	Yes
2	$FC \leq CC$	$FC > CC$	Yes	No
2	$FC > CC$	$FC \leq CC$	No	Yes
4	$FC > CC$	$CC > FC$	No	No

taken from Schneider and Rohlfing (manuscript)

$$
\begin{aligned}
TYP_1 - TYP_2 = \; & [0.5 - (FC_{TYP_1} - FC_{TYP_2})] && \text{large difference in focal condition} \\
& + [0.5 - (Y_{TYP_1} - Y_{TYP_2})] && \text{large difference in outcome} \\
& + \left|min(CC_{TYP_1}) - min(CC_{TYP_2})\right| && \text{small difference in complementary conditions} \\
& + 2 * \left|(Y_{TYP_1} - min\left(FC_{TYP_1}, CC_{TYP_1}\right))\right| && \text{typical case close to diagonal} \\
& + 2 * \left|(Y_{TYP_2} - min\left(FC_{TYP_2}, CC_{TYP_2}\right))\right| && \text{typical case close to diagonal}
\end{aligned} \tag{5}
$$

For getting the best available pairs of two typical cases argument `cases` in function `mmr()` must be set to 1. The output is similar to the one for typical-IIR pairs of cases, the best available pair of typical cases for each focal conjunct being situated in as low a rank as possible, having the smaller formula value (Distance), and being both uniquely covered. Looking at the first term of the parsimonious solution, we can see that pair `Switzerland_03-Denmark_03` is the best available for focal conjunct *emp*, while pair `Switzerland_99-Denmark_99` is the best available for focal conjunct *OCCUP*, both pairs containing uniquely covered typical cases, being in Rank 1, and having the smallest *Distance* value for their respective focal conjunct.

```
# Get matching pairs of typical and typical cases for the first term
# of the parsimonious solution:

mmr (results = sol_yp, outcome = "EXPORT", neg.out = FALSE,
     sol = 1, match = TRUE, cases = 1, term = 1)

## Focal Conjunct emp :
## ----------
##          Typical1       Typical2 Distance PairRank UniqCov1 UniqCov2
## 12 Switzerland_03     Denmark_03     1.72        1     TRUE     TRUE
## 4  Switzerland_03     Denmark_99     1.44        2     TRUE     TRUE
## 3      Denmark_03     Denmark_99     1.62        2     TRUE     TRUE
## 8  Switzerland_03 Switzerland_99     2.13        2     TRUE     TRUE
## 7      Denmark_03 Switzerland_99     2.31        2     TRUE     TRUE
##
## Focal Conjunct OCCUP :
## ----------
##         Typical1       Typical2 Distance PairRank UniqCov1 UniqCov2
## 2 Switzerland_99     Denmark_99     1.27        1     TRUE     TRUE
## 4 Switzerland_03     Denmark_99     1.44        3     TRUE     TRUE
## 3     Denmark_03     Denmark_99     1.62        3     TRUE     TRUE
## 8 Switzerland_03 Switzerland_99     2.13        3     TRUE     TRUE
## 7     Denmark_03 Switzerland_99     2.31        3     TRUE     TRUE
```

Matching typical and deviant cases consistency

The comparative within-case analysis of a typical and a deviant case consistency aims at identifying the INUS condition missing from the sufficient term *S* in question. The best available pair of cases

maximizes the following criteria: their membership in S should be as high and similar as possible and their membership in Y is different as possible. Formula 6 translates these matching criteria into practice.

$$\begin{aligned} TYP - DCON = & [(1 - S_{TYP}) + (1 - S_{DCON})] && \text{large membership in term} \\ & + [1 - (Y_{TYP} - Y_{DCON})] && \text{large difference in outcome} \\ & + |S_{TYP} - S_{DCON}| && \text{similar membership in term} \end{aligned} \tag{6}$$

Setting cases to `3` we get best available pair of typical and deviant consistency cases for each sufficient term in the parsimonious solution `sol_yp`. For identifying a missing INUS in sufficient term $emp * OCCUP$, the best available pair of cases that we could choose for process-tracing would be the one between typical case `Switzerland_03` and deviant consistency case `Australia_90`, as they have the smallest formula value (Distance=0.84).

```
# Get matching pairs of typical and deviant consistency cases for the
# parsimonious solution:

mmr (results = sol_yp, outcome = "EXPORT", neg.out = FALSE,
     sol = 1, match = TRUE, cases = 3)
```

```
## Term emp*OCCUP :
## ----------
##          typical deviant_consistency distance      term best_matching_pair
## 1 Switzerland_03        Australia_90     0.84 emp*OCCUP               TRUE
## 2 Switzerland_99        Australia_90     0.85 emp*OCCUP              FALSE
## 3 Switzerland_99      Switzerland_90     0.85 emp*OCCUP              FALSE
## 4 Switzerland_03      Switzerland_90     0.92 emp*OCCUP              FALSE
## 5 Switzerland_03        Australia_95     0.96 emp*OCCUP              FALSE
##
## Term BARGAIN*UNI*STOCK :
## ----------
##          typical deviant_consistency distance             term
## 1 Netherlands_03        Australia_95     0.55 BARGAIN*UNI*STOCK
## 2 Netherlands_03        Australia_03     0.59 BARGAIN*UNI*STOCK
## 3 Netherlands_99        Australia_95     0.65 BARGAIN*UNI*STOCK
## 4 Netherlands_99        Australia_03     0.69 BARGAIN*UNI*STOCK
## 5 Netherlands_99            Spain_99     0.73 BARGAIN*UNI*STOCK
##   best_matching_pair
## 1               TRUE
## 2              FALSE
## 3              FALSE
## 4              FALSE
## 5              FALSE
##
## Term occup*STOCK*ma :
## ----------
##    typical deviant_consistency distance          term best_matching_pair
## 1   USA_03           Spain_03     0.84 occup*STOCK*ma               TRUE
## 2 Japan_99           Spain_03     0.86 occup*STOCK*ma              FALSE
## 3 Japan_03           Spain_03     0.88 occup*STOCK*ma              FALSE
## 4   USA_90           Spain_03     0.89 occup*STOCK*ma              FALSE
## 5   USA_95           Spain_03     0.91 occup*STOCK*ma              FALSE
```

Matching deviant cases coverage and IIR cases

The comparative within-case analysis of a deviant case coverage and an IIR case aims at identifying the sufficient conjunction S_+ missing from the sufficient solution formula generated with QCA. The point of reference for matching cases is their membership in the truth table row TT to which they belong. Analogous to the within-case comparison of a typical and a deviant case consistency case, the goal is to maximize both cases' membership and their similarity in TT and their difference in Y.

Formula 7 achieves this.

$$\begin{aligned} DCOV - IIR = & [(1 - TT_{DCOV}) + (1 - TT_{IIR})] && \text{large membership in TT row} \\ & + [1 - (Y_{DCOV} - Y_{IIR})] && \text{large difference in outcome} \\ & + |TT_{DCOV} - TT_{IIR}| && \text{similar membership in TT row} \end{aligned} \tag{7}$$

Cases for this forth type of comparison can be identified by setting `cases` to `4`. Since for deviant coverage and IIR cases we are interested in identifying an entire missing sufficient term, the output for these pairs is focused on matching pairs in truth table rows, rather than in sufficient terms. Therefore, the output is sorted by truth table rows (the columns starting with *TT* showing the combination of conditions) and for each truth table row we can identify a best matching pair of cases according to formula values in column *distance*. For example, if we focus on truth table row $EMP * BARGAIN * uni * OCCUP * stock * ma$ (rows 6, 7, 8, 9, 10 in the output), the deviant case coverage `France_95` and the IIR case `Finland_90` constitute the best matching pair, having the smallest formula value for this specific truth table row (distance=1.43).

```
# Get matching pairs of deviant coverage and IIR cases for the
# parsimonious solution:

mmr (results = sol_yp, outcome = "EXPORT", neg.out = FALSE,
     sol = 1, match = TRUE, cases = 4)

## Matching Deviant Coverage-IIR Cases :
## ----------
##    deviant_coverage individually_irrelevant distance best_matching_pair
## 1            USA_99          New Zealand_95     0.58               TRUE
## 2             UK_03          New Zealand_95     0.59              FALSE
## 3             UK_99          New Zealand_95     0.59              FALSE
## 4         Ireland_03          New Zealand_95     0.73              FALSE
## 5         Ireland_99          New Zealand_95     0.73              FALSE
## 6          France_95              Finland_90     1.43               TRUE
## 7          France_95                Italy_95     1.44              FALSE
## 8         Germany_03              Finland_90     1.46              FALSE
## 9         Germany_03                Italy_95     1.47              FALSE
## 10         France_95              Austria_90     1.51              FALSE
## 11         Sweden_95               Sweden_90     1.41               TRUE
## 12         Sweden_95              Belgium_95     1.43              FALSE
## 13   Netherlands_95              Belgium_95     1.48              FALSE
## 14   Netherlands_95               Sweden_90     1.64              FALSE
##    TT_EMP TT_BARGAIN TT_UNI TT_OCCUP TT_STOCK TT_MA
## 1       0          0      1        0        1     1
## 2       0          0      1        0        1     1
## 3       0          0      1        0        1     1
## 4       0          0      1        0        1     1
## 5       0          0      1        0        1     1
## 6       1          1      0        1        0     0
## 7       1          1      0        1        0     0
## 8       1          1      0        1        0     0
## 9       1          1      0        1        0     0
## 10      1          1      0        1        0     0
## 11      1          1      0        1        1     1
## 12      1          1      0        1        1     1
## 13      1          1      0        1        1     1
## 14      1          1      0        1        1     1
```

Enhanced standard analysis (ESA)

Limited empirical diversity is an omnipresent feature in social science data. The treatment of logical remainders rows has been a major theme since the Ragin's path-breaking book (Ragin, 1987, esp. chapter 7). In Ragin (2008, chapters 8 and 9), three approaches towards remainders are proposed under the label of the Standard Analysis (SA). Researchers can decide not to include them into the

logical minimization (yielding the conservative or complex solution *CS*), to include all remainders that are simplifying (yielding the most parsimonious solution *PS*), or to include only those simplifying assumption that are easy based on so-called directional expectations (yielding the intermediate solution (*IS*)).

Schneider and Wagemann (2012, chapter 8) propose the Enhanced Standard Analysis (ESA), which argues that simplifying assumptions on specific remainders can be untenable. There are three sources of untenability. *Incoherent counterfactuals*, which are either logical remainders contradicting claims of necessity[6] or assumptions made for the negated outcome[7], and *implausible counterfactuals*, which consist of claims about impossible remainders[8]. ESA simply stipulates that no QCA solution formula can be based on untenable assumptions.

Figure 6 provides a graphical representation of the different types of assumptions as defined by SA and ESA. Both approaches only allow for simplifying assumptions[9] (i.e. those in the inner circle) and both distinguish between difficult and easy counterfactuals (i.e. the vertical line inside the circle)[10]. ESA but not SA does block any untenable assumption (i.e. the gray area on the lower part). A risk of making untenable assumption is given whenever a researcher is claiming the presence of a necessary condition, when statements of sufficiency for both the outcome and its negation are made, and/or when two or more conditions with mutually exclusive categories are used in a truth table. ESA requires that researchers identify those logical remainder rows whose inclusion into the logical minimization would amount to an untenable claim. As a result, one obtains the enhanced PS and the enhanced IS.[11]

Function `esa()` provides a straightforward tool for avoiding untenable assumptions and thus putting ESA into practice. First, function `esa()` can exclude remainders that contradict single necessary conditions, unions of necessary conditions, or more complicated expressions of necessity. For example, assuming that the disjunction $STOCK + MA$ is necessary for the outcome EXPORT, we ban all remainder rows implied by this necessity claim in the `nec_cond` argument. All the logical remainder rows that are subsets of $\neg STOCK \neg MA$ are subsequently set ot OUT = 0 in the truth table object `ttnew` and thus excluded from further logical minimization.

```
# Ban logical reminders contradicting necessity:
# Let's assume that "STOCK + MA" is necessary for "EXPORT":

newtt <- esa(oldtt = TT_y, nec_cond = "STOCK + MA")

##    EMP BARGAIN UNI OCCUP STOCK MA OUT n incl PRI cases
## 1    0       0   0     0     0  0   0 0    -   -
## 3    0       0   0     0     1  0   ? 0    -   -
## 4    0       0   0     0     1  1   ? 0    -   -
## 5    0       0   0     1     0  0   0 0    -   -
## 6    0       0   0     1     0  1   ? 0    -   -
## 7    0       0   0     1     1  0   ? 0    -   -
## 9    0       0   1     0     0  0   0 0    -   -
## 13   0       0   1     1     0  0   0 0    -   -
## 14   0       0   1     1     0  1   ? 0    -   -
## 15   0       0   1     1     1  0   ? 0    -   -
## 17   0       1   0     0     0  0   0 0    -   -
## 18   0       1   0     0     0  1   ? 0    -   -
## 20   0       1   0     0     1  1   ? 0    -   -
## 21   0       1   0     1     0  0   0 0    -   -
## 22   0       1   0     1     0  1   ? 0    -   -
## 23   0       1   0     1     1  0   ? 0    -   -
## 24   0       1   0     1     1  1   ? 0    -   -
## 25   0       1   1     0     0  0   0 0    -   -
## 26   0       1   1     0     0  1   ? 0    -   -
## 30   0       1   1     1     0  1   ? 0    -   -
```

[6] $X \leftarrow Y$ (implies $\neg X \rightarrow \neg Y$) & $\neg X \rightarrow Y$

[7] $X \rightarrow Y$ & $X \rightarrow \neg Y$

[8] For instance, X = 'rich-poor country' etc & $X \rightarrow Y$

[9] Schneider and Wagemann (2012) propose Theory-Guided ESA (TESA) as an approach in which parsimony is not the primary decision rule for choosing remainders.

[10] It can be noted that all assumptions that contribute to parsimony are either easy or difficult, they cannot be neither. This is because all assumptions not constrainted by directional expectations (if there are any or just some) are by default easy.

[11] For an application of ESA, see, for instance Thomann (2015).

```
## 31   0       1   1     1     1  0   ? 0    -   -
## 33   1       0   0     0     0  0   0 0    -   -
## 34   1       0   0     0     0  1   ? 0    -   -
## 35   1       0   0     0     1  0   ? 0    -   -
## 36   1       0   0     0     1  1   ? 0    -   -
## 37   1       0   0     1     0  0   0 0    -   -
## 38   1       0   0     1     0  1   ? 0    -   -
## 39   1       0   0     1     1  0   ? 0    -   -
## 40   1       0   0     1     1  1   ? 0    -   -
## 41   1       0   1     0     0  0   0 0    -   -
## 42   1       0   1     0     0  1   ? 0    -   -
## 44   1       0   1     0     1  1   ? 0    -   -
## 45   1       0   1     1     0  0   0 0    -   -
## 46   1       0   1     1     0  1   ? 0    -   -
## 47   1       0   1     1     1  0   ? 0    -   -
## 48   1       0   1     1     1  1   ? 0    -   -
## 50   1       1   0     0     0  1   ? 0    -   -
## 51   1       1   0     0     1  0   ? 0    -   -
## 52   1       1   0     0     1  1   ? 0    -   -
## 54   1       1   0     1     0  1   ? 0    -   -
## 58   1       1   1     0     0  1   ? 0    -   -
## 59   1       1   1     0     1  0   ? 0    -   -
```

Secondly, the `esa()` function can also ban implausible counterfactuals to produce truth tables in which specific logical remainders identifyied through conjunctions are excluded. For example, we can ban all remainder rows that have $BARGAIN+ \sim OCCUP$ by using the Boolean expression in argument `untenable_LR`. Finally, the function can exclude contradictory simplifying assumptions (which are another form of untenable assumptions) and empirically observed rows that are part of simultaneous subset relations[12] by just using the unique truth table row identifier in the argument `contrad_rows`. While argument `untenable_LR` accepts Boolean expression for excluding only logical remainders, argument `contrad_rows` can exclude both empirically observed rows and remainder rows through their unique identifier (row number).[13]

```
# Ban impossible logical remainders:
newtt <- esa(oldtt = TT_y, untenable_LR = "BARGAIN*~OCCUP")

# Ban contradictory rows:
newtt <- esa(oldtt = TT_y, contrad_rows = c("19", "14", "46", "51"))
```

Set-analytic theory evaluation

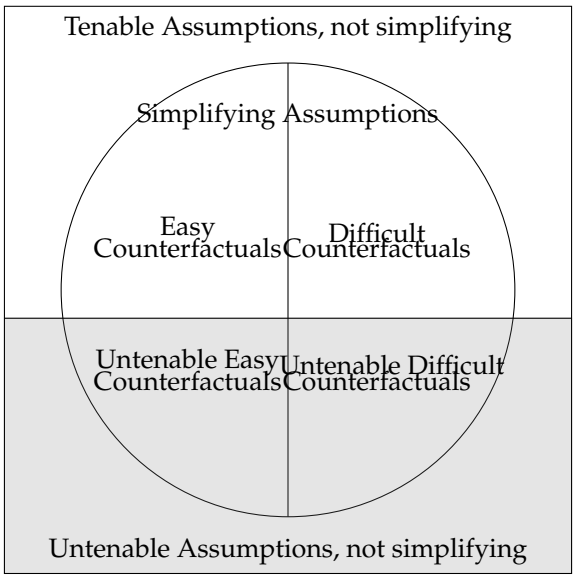
Ragin (1987, chapter 7) spells out the notion of theory evaluation. In essence, it consists of identifying the overlap between a researcher's theory (T) as formulated prior to the empirical analysis and the empirical results (S) obtained via QCA. With both T and S being represented in the form of Boolean expressions, all four logically possible combinations between T and S can be expressed in Boolean terms as well and each case's membership in each of these four expressions be calculated. Theory evaluation reveals which aspects of T are empirically corroborated by S and which ones are not. It also reveals how strong this emirical support is. Last but not least, theory evaluation can serve as a case selection device by identifying cases that display membership scores in the empirical solution and the outcome that are expected or utterly unexpected based on T.

Schneider and Wagemann (2012, chapter 11) refine Theory Evaluation by taking into account each case's membership score not only in T and S, but also outcome Y. This is necessary due to the development of parameters of fit based on which by now it has become widespread practice to allow for solution formulas with less than perfect consistency and coverage scores. This means that in applied QCA, there are cases of $S\&\neg Y$ and of $\neg S\&Y$. This gives rise to eight different areas. Each area can be defined as a Boolean expression, provides different analytic information, and defines

[12] Simultaneous subset relations happen when an empirically observed truth table row is a consistent enough subset of both the outcome and its negation.

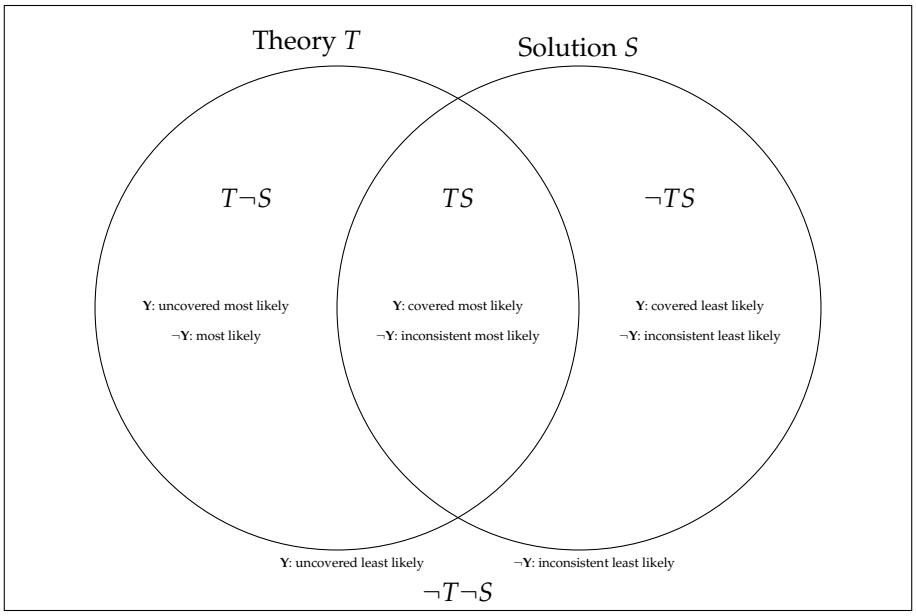
[13] Due to space restrictions, output for these functions is not shown here.

Figure 6: Graphical Representation of Different Types of Assumptions on Remainders



different types of cases. Each case has partial membership in all areas but only in one of higher than 0.5. Figure 7 provides a visualization of the areas and the kinds of cases in each area.[14]

Figure 7: Graphical Representation of Theory Evaluation



Function `theory.evaluation()` performs the theory evaluation procedure between a theory specified in Boolean terms and results obtained using the **QCA** package. Assuming that the theory can be summarized as `EMP*~MA + STOCK`, the example below shows how theory evaluation works using the second intermediate solution for outcome *EXPORT*. The first part of the output shows the names and proportion of cases in each of the intersections between theory and the empirical solution. The second part of the output shows parameters of fit for the solution, the theory, and their intersections, which indicate how much each of these areas are in line with a statement of sufficiency for *EXPORT*. Additionally, the function also stores the membership of each case in each intersection between theory and empirics, which can be accessed by setting argument `print.data` to `TRUE`.

[14] For applied examples of theory evaluation, see, for instance, Sager and Thomann (2017); Schneider and Maerz (2017).

```
# Assuming the theory can be summarized as "EMP*~MA + STOCK",
# perform theory evaluation using the second intermediate solution:

theory.evaluation(theory = "EMP*~MA + STOCK", empirics = sol_yi,
                  outcome = "EXPORT", sol = 2, print.data=FALSE)

##
## Cases:
## ----------
##
## Covered Most Likely (T*E and Y > 0.5) :
## ********************
##
## Cases in the intersection/Total number of cases: 23 / 76 = 30.26 %
##
## Case Names:
##  [1] "Ireland_90"     "Japan_90"       "USA_90"         "Ireland_95"
##  [5] "Japan_95"       "Switzerland_95" "USA_95"         "Denmark_99"
##  [9] "Finland_99"     "France_99"      "Japan_99"       "Netherlands_99"
## [13] "Sweden_99"      "Switzerland_99" "Belgium_03"     "Denmark_03"
## [17] "Finland_03"     "France_03"      "Japan_03"       "Netherlands_03"
## [21] "Sweden_03"      "Switzerland_03" "USA_03"
##
## Covered Least Likely (t*E and Y > 0.5) :
## ********************
##
## Cases in the intersection/Total number of cases: 0 / 76 = 0 %
##
## Case Names:
## [1] "No cases in this intersection"
##
## Uncovered Most Likely (T*e and Y > 0.5) :
## ********************
##
## Cases in the intersection/Total number of cases:  12 / 76 = 15.79 %
##
## Case Names:
##  [1] "UK_90"          "France_95"      "Netherlands_95" "Sweden_95"
##  [5] "UK_95"          "Germany_99"     "Ireland_99"     "UK_99"
##  [9] "USA_99"         "Germany_03"     "Ireland_03"     "UK_03"
##
## Uncovered Least Likely (t*e and Y > 0.5) :
## ********************
##
## Cases in the intersection/Total number of cases:  0 / 76 = 0 %
##
## Case Names:
## [1] "No cases in this intersection"
##
## Inconsistent Most Likely (T*E and Y < 0.5) :
## ********************
##
## Cases in the intersection/Total number of cases:  12 / 76 = 15.79 %
##
## Case Names:
##  [1] "Canada_90"      "Switzerland_90" "Australia_95"   "Canada_95"
##  [5] "Denmark_95"     "Finland_95"     "Australia_99"   "Belgium_99"
##  [9] "Spain_99"       "Australia_03"   "Norway_03"      "Spain_03"
##
## Inconsistent Least Likely (t*E and Y < 0.5) :
## ********************
##
## Cases in the intersection/Total number of cases:  1 / 76 = 1.32 %
##
```

```
## Case Names:
## [1] "Australia_90"
##
## Consistent Most Likely (T*e and Y < 0.5) :
## ******************
##
## Cases in the intersection/Total number of cases:  23 / 76 = 30.26 %
##
## Case Names:
##  [1] "Austria_90"     "Belgium_90"     "Denmark_90"     "Finland_90"
##  [5] "France_90"      "Germany_90"     "Italy_90"       "Netherlands_90"
##  [9] "Norway_90"      "Spain_90"       "Sweden_90"      "Austria_95"
## [13] "Belgium_95"     "Germany_95"     "Italy_95"       "New Zealand_95"
## [17] "Norway_95"      "Spain_95"       "Austria_99"     "Canada_99"
## [21] "Italy_99"       "Canada_03"      "Italy_03"
##
## Consistent Least Likely (t*e and Y < 0.5) :
## ******************
##
## Cases in the intersection/Total number of cases:  4 / 76 = 5.26 %
##
## Case Names:
## [1] "New Zealand_90" "New Zealand_99" "Norway_99"      "New Zealand_03"
##
##
## Fit:
## ----------
##
##                     Cons.Suf Cov.Suf   PRI Cons.Suf(H)
## emp*bargain*OCCUP      0.909   0.194 0.721       0.865
## BARGAIN*UNI*STOCK      0.796   0.497 0.665       0.704
## emp*UNI*OCCUP*ma       0.919   0.171 0.611       0.894
## emp*occup*STOCK*ma     0.904   0.298 0.802       0.859
## UNI*occup*STOCK*ma     0.894   0.341 0.795       0.853
## Sol.Formula            0.799   0.705 0.691       0.716
## Theory                 0.639   0.973 0.515       0.550
## T*E                    0.811   0.705 0.707       0.726
## t*E                    0.825   0.165 0.423       0.764
## T*e                    0.651   0.547 0.419       0.592
## t*e                    0.697   0.203 0.232       0.640
```

Diagnostic tools for clustered data structures

Most of the data analyzed in the social sciences and neighboring disciplines contains structures, or layers, that might be analytically relevant but are not captured by the models used to analyze that data. García-Castro and Arino (2016) discuss clustering along a *temporal* dimension. This can be days, years, decades, or substantively important periods (before - after a crisis). Clusters can also be of different origin. For instance, cases can be clustered along *geographic* units, such as world regions or subnational units. There can also be clusterings along *substantive* lines, such as e.g. economic sectors, parties, political regime types.

Whenever a researcher is not capturing these differences via a condition in her QCA model, she de facto assumes that the analytic difference does not matter. There are often good reasons to not include additional conditions into an analysis, with keeping limited diversity at bay being one of them. It should, however, been put to an empirical test whether it is ok to pool cases across different time periods, geographic units, and/or substantive areas.

Function `cluster()` provides the tools for performing such a test. It analyzes whether the QCA solution formula obtained from the pooled data also can be found in each of the sub-populations in the data. If it can, then pooling the data is fine. If it cannot, then pooling the data is not fine because it produces a solution that does not hold for all sub-populations. Rather, it is an artifact of having pooled cases that follow different causal logics. In this case, researchers might decide to drop from their analysis those sub-populations that do not follow the general pattern or to include a condition

into their model that captures the difference.

For using the `cluster()` function, researchers need data in the long format, with a column identifying the unit of analysis and a column identifying the clustering element. In our example, using the Schneider et al. (2010) data, the column identifyign the units is the `COUNTRY` column, while the cluster element is stored in the `YEAR` column. After having the data in the long format, we can get the diagnostic of how our solution holds throughout the different units and clusters by just imputing the solution (in this case `sol_yi`) in the `cluster()` function, while also specifying the data, the outcome, the unit identifier, and the cluster identifier. The first part of the output shows the consistency sufficiency for the overall, pooled sample for each sufficient term and the entire solution to be diagnosed. This first row of the output should be equivalent to the consistency measures obtained when producing the solution. The rows below show consistency values of the same terms and solution, but for each cluster and each unit in part. These are the values we would obtain if the analysis where to be run for each cluster subsample and each unit subsample separately. For example, the pooled consistency of sufficient term $emp * UNI * OCCUP * ma$ is 0.919 but only 0.733 for the year 1990. This might be an indication for the researcher that her sufficiency statement is not as consistent and might not work in the same way for the cluster of cases from 1990. In general, if consistency values between clusters differ greatly from pooled consistency for a term, we might want to rexamine the setup of the analysis to account for this. The `Distances:` section of the output reports on how much the parameters of fit differ from the clusters to the pooled data. Finally, the last part of the output displays a similar table for coverage, with pooled, between clusters, and within units measures.

```
# Perfom cluster diagnostic:
# First we need to load the Schneider et. al. (2010) data in the long format:

data(SCHLF)

# This data has a column identifying the unit (country)
# and the clustering element (year):

head(SCHLF)

##               EMP BARGAIN  UNI OCCUP STOCK   MA EXPORT   COUNTRY YEAR
## Australia_90 0.07    0.90 1.00  0.68  0.45 0.33   0.19 Australia 1990
## Austria_90   0.70    0.98 0.01  0.91  0.01 0.05   0.25   Austria 1990
## Belgium_90   0.94    0.95 0.14  0.37  0.26 0.14   0.14   Belgium 1990
## Canada_90    0.04    0.21 0.99  0.11  0.62 0.31   0.28    Canada 1990
## Denmark_90   0.59    0.78 0.10  0.55  0.53 0.10   0.34   Denmark 1990
## Finland_90   0.70    0.97 0.20  0.95  0.02 0.13   0.17   Finland 1990

# Get the intermediate solution:

sol_yi <- minimize(SCHLF, outcome = "EXPORT",
                   conditions = c("EMP","BARGAIN","UNI",
                                  "OCCUP","STOCK", "MA"),
                   incl.cut1 = .9,
                   include = "?",
                   details = TRUE, show.cases = TRUE,
                   dir.exp = c(0,0,0,0,0,0))

# Get pooled, within, and between consistencies for the intermediate solution:

cluster(data = SCHLF, results = sol_yi, outcome = "EXPORT", unit_id = "COUNTRY",
        cluster_id = "YEAR")

## Consistencies:
## ---------------
##                     emp*bargain*OCCUP BARGAIN*UNI*STOCK emp*UNI*OCCUP*ma
## Pooled                          0.909             0.796            0.919
## Between 1990                    0.839             0.873            0.733
## Between 1995                    0.903             0.727            0.953
## Between 1999                    0.928             0.802            1.000
## Between 2003                    0.951             0.818            1.000
## Within Australia                1.000             0.405            0.634
```

```
## Within Austria                1.000             1.000            1.000
## Within Belgium                1.000             0.803            1.000
## Within Canada                 1.000             1.000            1.000
## Within Denmark                1.000             0.757            1.000
## Within Finland                1.000             0.835            0.957
## Within France                 1.000             0.916            1.000
## Within Germany                1.000             1.000            1.000
## Within Ireland                1.000             1.000            1.000
## Within Italy                  1.000             0.800            1.000
## Within Japan                  1.000             1.000            1.000
## Within Netherlands            1.000             1.000            1.000
## Within NewZealand             0.414             0.875            0.727
## Within Norway                 0.965             0.486            0.930
## Within Spain                  1.000             0.524            1.000
## Within Sweden                 1.000             0.926            1.000
## Within Switzerland            0.880             1.000            1.000
## Within UK                     1.000             1.000            1.000
## Within USA                    1.000             1.000            1.000
##                    emp*occup*STOCK*ma bargain*occup*STOCK*ma
## Pooled                          0.904                  0.913
## Between 1990                    0.858                  0.903
## Between 1995                    0.847                  0.884
## Between 1999                    1.000                  1.000
## Between 2003                    0.995                  0.916
## Within Australia                0.865                  1.000
## Within Austria                  1.000                  1.000
## Within Belgium                  1.000                  1.000
## Within Canada                   0.587                  0.587
## Within Denmark                  0.732                  1.000
## Within Finland                  1.000                  1.000
## Within France                   1.000                  1.000
## Within Germany                  1.000                  1.000
## Within Ireland                  1.000                  1.000
## Within Italy                    1.000                  1.000
## Within Japan                    1.000                  0.997
## Within Netherlands              1.000                  1.000
## Within NewZealand               0.710                  0.710
## Within Norway                   0.930                  0.961
## Within Spain                    1.000                  0.628
## Within Sweden                   1.000                  1.000
## Within Switzerland              1.000                  1.000
## Within UK                       1.000                  1.000
## Within USA                      1.000                  1.000
##
##
## Distances:
## ----------
##                        emp*bargain*OCCUP BARGAIN*UNI*STOCK
## From Between to Pooled             0.023             0.032
## From Within to Pooled              0.031             0.050
##                        emp*UNI*OCCUP*ma emp*occup*STOCK*ma
## From Between to Pooled            0.060              0.039
## From Within to Pooled             0.024              0.030
##                        bargain*occup*STOCK*ma
## From Between to Pooled                  0.024
## From Within to Pooled                   0.032
##
##
## Coverages:
## ----------
##                    emp*bargain*OCCUP BARGAIN*UNI*STOCK emp*UNI*OCCUP*ma
## Pooled                         0.194             0.497            0.171
## Between 1990                   0.231             0.246            0.193
## Between 1995                   0.206             0.466            0.271
```

```
## Between 1999                  0.174            0.589           0.042
## Between 2003                  0.184            0.570           0.214
## Within Australia              0.415            1.000           0.675
## Within Austria                0.075            0.041           0.279
## Within Belgium                0.138            0.959           0.283
## Within Canada                 0.328            0.545           0.246
## Within Denmark                0.273            0.894           0.317
## Within Finland                0.059            0.937           0.282
## Within France                 0.070            0.805           0.118
## Within Germany                0.236            0.374           0.205
## Within Ireland                0.113            0.352           0.098
## Within Italy                  0.173            0.367           0.112
## Within Japan                  0.161            0.064           0.161
## Within Netherlands            0.150            0.748           0.183
## Within NewZealand             1.000            0.778           0.667
## Within Norway                 0.598            0.978           0.435
## Within Spain                  0.204            0.710           0.204
## Within Sweden                 0.061            0.761           0.054
## Within Switzerland            0.738            0.244           0.032
## Within UK                     0.075            0.282           0.052
## Within USA                    0.037            0.045           0.037
##                   emp*occup*STOCK*ma bargain*occup*STOCK*ma
## Pooled                         0.298                  0.278
## Between 1990                   0.452                  0.450
## Between 1995                   0.459                  0.345
## Between 1999                   0.069                  0.117
## Between 2003                   0.321                  0.290
## Within Australia               0.675                  0.317
## Within Austria                 0.041                  0.034
## Within Belgium                 0.228                  0.103
## Within Canada                  0.701                  0.701
## Within Denmark                 0.480                  0.238
## Within Finland                 0.218                  0.042
## Within France                  0.132                  0.048
## Within Germany                 0.205                  0.169
## Within Ireland                 0.291                  0.118
## Within Italy                   0.265                  0.184
## Within Japan                   0.456                  0.947
## Within Netherlands             0.196                  0.070
## Within NewZealand              0.611                  0.611
## Within Norway                  0.435                  0.533
## Within Spain                   0.204                  0.581
## Within Sweden                  0.054                  0.057
## Within Switzerland             0.093                  0.093
## Within UK                      0.072                  0.072
## Within USA                     0.717                  0.717
```

Function `cluster()` can be applyied in a similar fashion for necessary relationships by just setting argument `necessity` to `TRUE` and inputting the necessary condition to be diagnosed in the field `results`. Additionally, we can also diagnose Boolean expressions by just entering this into the `results` argument.

```
# Get pooled, within, and between consistencies for ~EMP
# as necessary for EXPORT:

cluster(data = SCHLF, results = "~EMP", outcome = "EXPORT",
        unit_id = "COUNTRY", cluster_id = "YEAR", necessity=TRUE)

# Get pooled, within, and between consistencies for EMP*~MA*STOCK
# as sufficient for EXPORT:

cluster(data = SCHLF, results = "EMP*~MA*STOCK", outcome = "EXPORT",
        unit_id = "COUNTRY", cluster_id = "YEAR")
```

Additional functions

Plotting sufficient terms and solutions, truth table rows, and necessity relations

Package **SetMethods** also includes a function `pimplot()` for plotting each sufficient term and the solution formula (obtained by using the `minimize()` function in package **QCA**). The function can also plot truth table rows against the outcome by using arguments `incl.tt` or `ttrows` as in the examples below. Additionally, the function can plot results obtained from necessity analyses using an object of class `"sS"` (obtained by using the `superSubset()` function in package **QCA**) by setting argument `necessity` to `TRUE`.[15]

```
# Plot the prime implicants of the parsimonious solution:

pimplot(data = SCHF, results = sol_yp, outcome = "EXPORT")

# Plot all truth table rows with a consistency higher than 0.9:

pimplot(data=SCHF, results = sol_yi, incl.tt=0.9, outcome = "EXPORT", sol = 1)

# Plot truth table rows "60" and "61":

pimplot(data=SCHF, results = sol_yi, ttrows =c("60","61"),
        outcome = "EXPORT", sol = 1)

# For plotting results of necessity analyses using superSubset,
# the first stept is to obtain an "sS" object:

SUPSUB <- superSubset(SCHF, outcome="EXPORT",
                      conditions = c("EMP","BARGAIN","UNI","OCCUP","STOCK", "MA"),
                      relation = "necessity", incl.cut = 0.8)
SUPSUB

# This can be imputed as result and necessity should be set to TRUE:

pimplot(data = SCHF, results = SUPSUB, outcome = "EXPORT",
        necessity = TRUE)
```

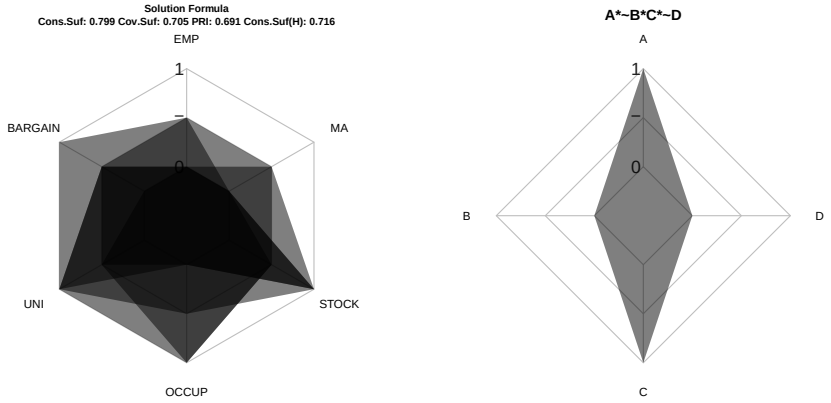
QCAradar

Another function included in the package is the `QCAradar()` function which allows visualization of QCA results or simple Boolean expressions in the form of a radar chart[16]. The function accepts in the argument `results` sufficient solutions obtained through the function `minimize()` in package **QCA**, or Boolean expressions involving more than three conditions, as in the second example below.

```
# Display radar chart for the second intermediate solution:

QCAradar(results = sol_yi, outcome = "EXPORT", fit=TRUE, sol = 2)
```

Figure 8a shows a radar chart for the second intermediate solution formula. The different sufficient terms are overlapping on the radar in different shades. For example, we can see the first term `emp*bargain*OCCUP`, as condition *EMP* is missing it is set to 0 for that respective corner, condition *BARGAIN* is missing and set to 0, and condition *OCCUP* is present and set to 1. Since the rest of the conditions are not specified in this term, they are all left at -.

[15] The plots resulting from these functions are not included in the paper due to length reasons.
[16] See Maerz (2017) for an applied example of radar charts.



(a) Radar Chart Intermediate Solution Formula 2

(b) Expression A* B*C* D

Figure 8: Radar Charts

```
# Show a radar chart for the following boolean expression "A*~B*C*~D"

QCAradar(results = "A*~B*C*~D")
```

Figure 8b shows a radar chart for the Boolean expression "$A* \sim B * C* \sim D$". Conditions A and C that are present are set to 1 for their respective corners, while conditions B and D that are missing and set to 0. There are not conditions left at - in this figure, as all conditions are specified.

Indirect calibration

SetMethods also includes a function for performing the indirect calibration procedure described by Ragin (2008)[17]. This procedure assumes that the cases included in the analysis have interval-scale raw scores which can be initially sorted broadly into different levels of fuzzy set membership. Subsequently, the raw scores are transformed into calibrated scores using a binomial or a beta regression. Assuming that vector `x` contains the initial raw scores, while vector `x_cal` contains the rough grouping of those values into set membership scores, function `indirectCalibration()` can produce a vector of fuzzy-set scores a by fitting the `x` to `x_cal` using a binomial regression if `binom` is set to `TRUE`.

```
# Generate fake data

set.seed(4)
x <- runif(20, 0, 1)

# Find quantiles

quant <- quantile(x, c(.2, .4, .5, .6, .8))

# Theoretical calibration

x_cal <- NA
x_cal[x <= quant[1]] <- 0
x_cal[x > quant[1] & x <= quant[2]] <- .2
x_cal[x > quant[2] & x <= quant[3]] <- .4
x_cal[x > quant[3] & x <= quant[4]] <- .6
x_cal[x > quant[4] & x <= quant[5]] <- .8
x_cal[x > quant[5]] <- 1
x_cal

# Indirect calibration (binomial)
```

[17]The **QCA** package can also perform indirect calibration through its `calibrate` function

```
a <- indirectCalibration(x, x_cal, binom = TRUE)
a
```

Conclusions

In this article, we have presented the main functionalities of the R package **Setmethods**. It is true that starting to perform QCA in R is more onerous than starting with a point-and-click software. Yet, the flexibility offered by R is also its strength, especially for a young method like QCA. As set methods continue to develop, software implementations need to be updated and improved at a fast rate. Package **SetMethods** is designed to do precisely this: providing a tool for implementing new ideas that enhance set-theoretic analyses for applied researchers.

Acknowledgements

We thank Juraj Medzihorsky and Mario Quaranta for their intput into previous versions of the **SetMethods** package. We also thank the participants of various ECPR Summer and Winter Schools in Methods and Techniques whose questions and testing are continuously improving the package.

Bibliography

L. Cronqvist. Tosmana: Tool for Small-n Analysis, Version 1.3.2.0 [Computer Program]. Record ID: 8880, 2011. [p507]

A. Dusa. User Manual for the QCA(GUI) Package in R. *Journal of Business Research*, 60(5):576–586, 2007. [p507]

R. García-Castro and M. A. Arino. A General Approach to Panel Data Set-Theoretic Research. *Journal of Advances in Management Sciences & Information Systems*, 2:63–76, 2016. [p526]

G. Goertz and J. Mahoney. *A Tale of Two Cultures: Contrasting Qualitative and Quantitative Paradigms*. Princeton University Press, Princeton, N.J, 2012. [p507]

S. Maerz. *The Many Faces of Authoritarian Persistence*. PhD thesis, Central European University, Doctoral Dissertation, Central European University, 2017. [p530]

J. Medzihorsky, I.-E. Oana, M. Quaranta, and C. Q. Schneider. SetMethods: Functions for Set-Theoretic Multi-Method Research and Advanced QCA, R Package, Version 2.1. *https://cran.r-project.org/web/%0Apackages/SetMethods/index.html*, 2016. [p507]

K. S. Mikkelsen. Fuzzy-Set Case Studies. *Sociological Methods & Research*, 46(3):422–455, 2017. ISSN 0049-1241. URL https://doi.org/10.1177/0049124115578032. [p508]

C. C. Ragin. *The Comparative Method: Moving Beyond Qualitative and Quantitative Strategies*. University of California Press, Berkeley, 1987. ISBN 0520058348. [p521, 523]

C. C. Ragin. *Redesigning Social Inquiry: Fuzzy Sets and Beyond*. University of Chicago Press, Chicago, 2008. [p508, 521]

C. C. Ragin, K. A. Drass, and S. Davey. Fuzzy-Set/Qualitative Comparative Analysis 2.0. http://www.u.arizona.edu/ cragin/fsQCA/, 2006. [p507]

B. Rihoux and B. Lobe. The Case for Qualitative Comparative Analysis (QCA): Adding Leverage for Thick Cross-Case Comparison. In D. Byrne and C. C. Ragin, editors, *Sage Handbook Of Case-Based Methods*, pages 222–242. Sage, 2009. [p508]

B. Rihoux, P. Alamos, D. Bol, A. Marx, and I. Rezsohazy. From Niche to Mainstream Method? A Comprehensive Mapping of QCA Applications in Journal Articles from 1984 to 2011. *Political Research Quarterly*, 66(1):175–184, 2013. [p507]

I. Rohlfing and C. Q. Schneider. Improving Research On Necessary Conditions: Formalized Case Selection for Process Tracing after QCA. *Political Research Quarterly*, 66(1):220–235, 2013. [p508]

I. Rohlfing and C. Q. Schneider. A Unifying Framework for Causal Analysis in Set-Theoretic Multi-Method Research. *Sociological Methods & Research*, 47(1):37–63, 2018. URL https://doi.org/10.1177/0049124115626170. [p508]

F. Sager and E. Thomann. Multiple Streams in Member State Implementation: Politics, Problem Construction and Policy Paths in Swiss Asylum Policy. *Journal of Public Policy*, 37(3):287–314, 2017. ISSN 0143-814X. URL https://doi.org/10.1017/s0143814x1600009x. [p524]

C. Q. Schneider and S. Maerz. Legitimation, Cooptation, and Repression and the Survival of Electoral Autocracies. *Zeitschrift für Vergleichende Politikwissenschaft*, 11:213–235, 2017. ISSN 1865-2646. URL https://doi.org/10.1007/s12286-017-0332-2. [p524]

C. Q. Schneider and I. Rohlfing. Combining QCA and Process Tracing in Set-Theoretic Multi-Method Research. *Sociological Methods and Research*, 42(4):559–597, 2013. [p508]

C. Q. Schneider and I. Rohlfing. Case Studies Nested in Fuzzy-Set QCA on Sufficiency: Formalizing Case Selection and Causal Inference. *Sociological Methods & Research*, 45(3):526–568, 2016. ISSN 0049-1241. URL https://doi.org/10.1177/0049124114532446. [p508]

C. Q. Schneider and C. Wagemann. *Set-Theoretic Methods for the Social Sciences: A Guide to Qualitative Comparative Analysis*. Cambridge University Press, Cambridge, 2012. [p522, 523]

M. R. Schneider, C. Schulze-Bentrop, and M. Paunescu. Mapping the Institutional Capital of High-Tech Firms: A Fuzzy-Set Analysis of Capitalist Variety and Export Performance. *Journal of International Business Studies*, 41(2):246–266, 2010. ISSN 0047-2506. URL https://doi.org/10.1057/jibs.2009.36. [p507, 513, 527]

E. Thomann. Customizing Europe: Transposition as Bottom-up Implementation. *Journal of European Public Policy*, 22(10):1368–1387, 2015. URL https://doi.org/10.1080/13501763.2015.1008554. [p522]

Ioana-Elena Oana
Central European University (CEU)
Nador utca 9, 1051 Budapest
Hungary
Oana_Ioana-Elena@phd.ceu.edu

Carsten Q. Schneider
Central European University (CEU)
Nador utca 9, 1051 Budapest
Hungary
schneiderc@ceu.edu

HRM: An R Package for Analysing High-dimensional Multi-factor Repeated Measures

by Martin Happ, Solomon W. Harrar, Arne C. Bathke

Abstract High-dimensional longitudinal data pose a serious challenge for statistical inference as many test statistics cannot be computed for high-dimensional data, or they do not maintain the nominal type-I error rate, or have very low power. Therefore, it is necessary to derive new inference methods capable of dealing with high dimensionality, and to make them available to statistics practitioners. One such method is implemented in the package **HRM** described in this article. This new method uses a similar approach as the Welch-Satterthwaite t-test approximation and works very well for high-dimensional data as long as the data distribution is not too skewed or heavy-tailed. The package also provides a GUI to offer an easy way to apply the methods.

Introduction

Repeated measures designs appear in many different situations. In clinical studies, physicians might be interested in the effect of different treatments over time. Typically, a univariate or multivariate measurement is observed over a period of time for each subject. The subjects can be separated into different disjoint groups and can often be assumed not to influence one another. That is, observations on different subjects are considered independent. We refer to factors whose levels contain different sets of subjects as whole-plot or between-subjects factors. On the other hand, the time factor is a typical sub-plot or within-subjects factor because it structures the observations within individual subjects. Observations on the same subject may be dependent, and for different level combinations of the whole-plot factors, different types of dependence may be assumed. However, within a group defined by a factor level combination or *cell*, the observation vectors are modeled as coming from the same multivariate distribution. In particular, the dependence structure is the same for all subjects within such a cell.

Let us first consider a simple, additive model for univariate repeated measures data with different groups. This model already allows for different variances in different cells, as described above, but it is not sufficiently general regarding the dependence structures that are encompassed, and therefore a more general model will be considered later. An additive model for repeated measures could be written as

$$Y_{ijk} = \mu_{ij} + A_{ik} + \epsilon_{ijk},$$

where Y_{ijk} is the observation on subject (k) in group (i) at time point (j). It is decomposed additively into the population mean μ_{ik} in group (i) for time point (k), and two random components. Namely, $A_{ik} \sim N(0, \tau_i^2)$ is the specific effect for subject (k) in group (i) and $\epsilon_{ijk} \sim N(0, \sigma_i^2)$ is an additive random error term whose variance may depend on the group. All random variables ϵ_{ijk} and A_{ik} shall be assumed independent. Such a model formulation results in a covariance structure where observations on different subjects, that is, with different index pair (i, k), are independent, and observations Y_{ij_1k} and Y_{ij_2k} on the same subject (k) in group (i) all have the same variance $\sigma_i^2 + \tau_i^2$, as well as the same covariance τ_i^2, no matter the value of their indices j_1 and j_2. In other words, the correlation $\rho_{ij_1k,ij_2k} = \tau_i^2/(\sigma_i^2 + \tau_i^2)$ between two observations does not depend on their time distance $|j_1 - j_2|$.

For other dependence structures the correlation may decrease if two observations are further apart. Such dependence models are certainly possible and sometimes justified. By inspecting the compound symmetry structure above more closely, we also observe that also differences of two observations have the same variance independent of the choice of time points. Indeed, this property is generally referred to as sphericity. Therefore, the compound symmetry covariance structure is a special case of sphericity (see e.g., Bathke et al. (2009)).

Box (1954a,b) (or in more detail Huynh and Feldt (1970)) showed that the simple F-test is no longer valid when sphericity of the covariance structure cannot be assumed. In lower-dimensional situations, classical corrections such as the Greenhouse-Geisser (Geisser and Greenhouse, 1958; Greenhouse and Geisser, 1959) or Huynh-Feldt (Huynh and Feldt, 1976; Lecoutre, 1991) methods for the repeated measures ANOVA can be used when the covariance matrix sphericity assumption is violated. However, the performance of tests using the Greenhouse-Geisser or Huynh-Feldt corrections heavily depends on the actual underlying covariance structures as both have been derived under the assumption of

homoscedasticity across the levels of the between-subjects factor, which is very often not justifiable in practice. In other words, they assume that the variance-covariance structure between the repeated observations is the same for each cell. This assumption is rather restrictive, in particular in the presence of high dimensionality of the data vectors, and therefore a more general model is needed. In a classical MANOVA-design or repeated measures design, this problem was already tackled by Harrar (2009); Chen et al. (2010); Konietschke et al. (2015); Pauly et al. (2015); Harrar and Kong (2016); and Bathke et al. (2018), among others.

Here, and in the following, high-dimensionality of repeated measures means that the number of repeated measurements (d) per subject is larger than the sample sizes (n_i) in group (i) $(d > n_i)$. Methods that rely on the inverse of an empirical variance-covariance matrix cannot be calculated in this situation. A well-known example is Hotelling's T^2 test for multivariate data which can be used to test for simple treatment effects, that is, for the hypothesis that expectation vectors are the same for all groups. In a high-dimensional setting, the empirical variance-covariance matrix is singular, rendering Hotelling's T^2 unusable. Alternative test statistics that can in principle still be calculated for high-dimensional data include the ANOVA type statistic (Brunner et al., 1997; Brunner and Puri, 2001). Also, the Greenhouse-Geisser and the Huynh-Feldt correction for the repeated measures ANOVA can technically be computed, but they do suffer from the limitations mentioned before, and the performance of all three approaches deteriorates with increasing dimensionality (see, e.g., the simulation results in Happ et al. (2016)). Another possible approach for high-dimensional data is presented by the procedure proposed by Secchi et al. (2013), however this requires strong assumptions on the covariance matrices. Specifically, when we denote with $\mathbf{\Sigma}_i$ the $d \times d$ dimensional variance-covariance matrix for the i-th group, then the assumption from Secchi et al. (2013) can be formulated as

$$0 < \lim_{d\to\infty} \frac{\operatorname{tr}\left(\mathbf{\Sigma}_i^k\right)}{d} < \infty$$

for $k = 1, 2$, and 4 and for all groups $i = 1, \ldots, a$. Similar assumptions have also been used by Srivastava (2007); Srivastava and Yanagihara (2010); Chen et al. (2010) or Harrar and Kong (2016) and these assumptions were discussed in Pauly et al. (2015). For practical applications, it may be difficult to verify this assumption.

The method implemented in the package **HRM** does not assume any particular covariance structures (Happ et al., 2016, 2017), neither homoscedasticity across the between-subjects factor levels, nor sphericity regarding the levels of the within-subjects factor. In particular, no compound symmetry is assumed. Also, it does not require a stringent covariance assumption as in Secchi et al. (2013) or Srivastava and Yanagihara (2010). It is therefore rather general. A limitation exists, however, for this method. The theory has been derived under the asumption of normal errors. While it performs well empirically for a wide range of simulated data distributions, including discrete data, we recommend caution when the data is very skewed or heavy-tailed. For details, see Section Limitations.

In the following subsection, we will describe the model that is underlying the **HRM** package. Here, it will become clear that this model allows for rather general covariance structures, rendering the associated R package applicable for a wide range of high-dimensional data sets.

Statistical model

Consider a setting with two whole- and two sub-plot factors. Models for other designs can be formulated analogously. We refer to the whole-plot factors as A and S (e.g., group and subgroup) and to the sub-plot factors as C and D, respectively. For example, in the EEG data in Section Examples, the sub-plot factors are *variable* and *region*. The observations are represented by independent random vectors

$$\mathbf{X}_{ijk} = \left(\mathbf{X}'_{ijk11}, \ldots, \mathbf{X}'_{ijkbc}\right)' \sim N\left(\boldsymbol{\mu}_{ij}, \mathbf{\Sigma}_{ij}\right).$$

Here, $\mathbf{X}_{ijk}$ is the $d = b \times c$ dimensional data vector of the k-th subject in the i-th group and j-th subgroup for $i = 1, \ldots, a$ and $j = 1, \ldots, s$. Note that we do not impose any special dependence structure on the covariance-matrices $\mathbf{\Sigma}_{ij}$.

Accordingly, the factors A, S, B, and C have a, s, b, and c factor levels, respectively. The sample size in the i-th group and j-th subgroup is denoted by n_{ij}. Overall, there are $N = \sum_{i,j} n_{ij}$ experimental units. Our hypotheses of interest concern interaction effects between two or more factors, and main effects of a single factor. They can all be formulated as quadratic forms using appropriate projection matrices $\boldsymbol{K}_\phi$ which depend on the respective hypothesis of interest ϕ, namely

$$H_0(\phi) : \boldsymbol{\mu}' \boldsymbol{K}_\phi \boldsymbol{\mu} = 0.$$

In a setting with only one whole- and one sub-plot factor with a and d factor levels (*group* and *time*), this projection matrix is simply given by a Kronecker product of two matrices, that is $\boldsymbol{K}_\phi = \boldsymbol{W}_\phi \otimes \boldsymbol{S}_\phi$. If we want to test for the main group effect, we choose $\boldsymbol{W}_\phi = \boldsymbol{P}_a$ and $\boldsymbol{S}_\phi = 1/d\ \boldsymbol{J}_d$ where $\boldsymbol{P}_a$ is the $a \times a$ dimensional centering matrix and $\boldsymbol{J}_d$ is the $d \times d$ dimensional matrix whose entries are all equal to 1. With the latter matrix, we average over the sub-plot factor, and the former matrix centers a vector by subtracting its mean from all components of the vector. If we want to test for an interaction effect, we use for both $\boldsymbol{W}_\phi$ and $\boldsymbol{S}_\phi$ a centering matrix and if we want to test only for the main time effect we simply use $\boldsymbol{W}_\phi = 1/a\ \boldsymbol{J}_a$ and $\boldsymbol{S}_\phi = \boldsymbol{P}_d$. The corresponding test statistic is then constructed as

$$T(\phi) := \frac{\bar{\boldsymbol{X}}' \boldsymbol{K}_\phi \bar{\boldsymbol{X}}}{\operatorname{tr}\left(\widehat{\boldsymbol{\Sigma}}_\phi\right)},$$

where

$$\widehat{\boldsymbol{\Sigma}}_\phi = \boldsymbol{K}_\phi \left(\bigoplus_{i=1}^{a} \bigoplus_{j=1}^{s} \tfrac{1}{n_{ij}} \widehat{\boldsymbol{\Sigma}}_\phi \right),$$

$\bar{\boldsymbol{X}}$ is the $as \times d$-dimensional vector containing the means of all groups at each time point, and $\widehat{\boldsymbol{\Sigma}}_\phi$ is the empirical variance-covariance matrix for the i-th group. In general, we do not know the exact distribution of $T(\phi)$, but we can approximate the sampling distribution of $T(\phi)$ under null hypothesis by an F-distribution with estimated degrees of freedom $\hat{f}$ and $\hat{f}_0$. We refer to Brunner et al. (2012); Happ et al. (2016, 2017) for details regarding this so-called Box approximation and in particular the estimation of the degrees of freedom.

Note that this method only provides an approximation to the sampling distribution. Although it generally performs well, it is not asymptotically exact. There are asymptotic tests such as those proposed by Pauly et al. (2015) or Sattler and Pauly (2017), converging under some assumptions such as $\min\{n_i, i = 1, \ldots, a\} \to \infty$ and they also provide a small sample size approximation. Similar to the aforementioned papers, an advantage of the approximative method in **HRM** is that it is working very well even for rather small sample sizes.

The R package HRM

In the R package **HRM**, the test statistic is implemented for designs described in Section Statistical model. It is possible to use tests with up to four factors (at most two whole-plot or three sub-plot factors). Two S3 methods are provided to facilitate different data input formats. Both S3 methods can be called by the generic function `hrm_test`. The data can either be provided in the wide table format where each row represents the observations from one subject. This means that for each group, all observations have to be stored in a matrix. Then the matrices from all groups need to be elements of a `list`. This method only supports one whole- or one sub-plot factor, that is, a maximum of two factors can be used.

The other way is to provide the data in the long table format as a data frame. Here, all observations are in stored in one column. The other columns of the data frame specify the factor levels to which an observation belongs. For this type of data, at least one whole- and one sub-plot factor have to be used and they support in general up to two whole- and two sub-plot factors. There is also the special case of one whole- and three sub-plot factors implemented.

Depending on the type of data that is given as an argument to `hrm_test`, either the S3 method `hrm_test.list` or `hrm_test.data.frame` is internally called. We will give now a short description of these two S3 methods.

Methods

The S3 method `hrm_test.list` has two parameters, `data` and `alpha`. The second parameter has its default value set to 0.05 and specifies the type-I error level of the procedure which is used for calculating the critical value for the test procedure. The first parameter is a list containing the data in the wide table format for all groups. That is, the list has the structure

```
data <- list(group_1, group_2, ..., group_a),
```

where `group_i` is the data for the i-th group in the wide table format (the repeated measurements are the columns). Clearly, this method only works with one whole- and one sub-plot factor.

The method `hrm_test.data.frame` can be used for designs with up to two whole- and three sub-plot factors, but it is limited to a maximum number of four factors. The parameter `data` needs to be a data frame containing the data with the aforementioned columns. Similar to the method

`hrm_test.list` we can also specify the nominal type-I error rate by using the parameter `alpha`. The method `hrm_test.data.frame` needs a formula object. These are special R objects that allow to write a statistical model in a compact way and is used by many R packages. Let us assume the model

$$Y_{ijk} = \mu_{ij} + \epsilon_{ijk} = \alpha_i + \beta_j + \gamma_{ij} + \epsilon_{ijk},$$

where Y_{ijk} is the observation at time j for subject k in the i-th group and ϵ_{ijk} is the measurement error. The influence of the whole-plot factor is described by α and of the sub-plot factor by β. We denote by γ the interaction between these two factors. To write this model in a compact way in R we can use the formula

```
response ~ whole-plot factor * sub-plot factor.
```

Here the expression on the left-hand side of $\sim$ is the response variable. It is explained by the variables on the right-hand side. The symbol $*$ means that the interaction term γ_{ij} is also included in the model, that is $\mu_{ij} = \alpha_i + \beta_j + \gamma_{ij}$. Therefore main effects and the interaction effect are included in the model. This formula is equivalent to

```
response ~ whole-plot factor + sub-plot factor
+ whole-plot factor : sub-plot factor,
```

where : specifically denotes for the interaction effect. Hence the $*$ notation is an abbreviation for adding main and interaction effects. If we are only interested in the main effects then we could use the formula

```
response ~ whole-plot factor + sub-plot factor,
```

or alternatively if we are only interested in the interaction effect, our formula has the form

```
response ~ whole-plot factor : sub-plot factor.
```

For the method `hrm_test.data.fram` we also need to specify the column name for the subjects. This column name needs to be a `character`. Otherwise an error message is returned by the method.

All these methods previously described return an HRM object which contains a list consisting of the results in a `data.frame`, the formula, the type-I error, the subject column name and the column names of the whole- and sub-plot factors which are used, and the data used for testing.

In the special case when only one whole- and one sub-plot factor is used, it is possible to plot the so-called profile curves. In each group $i = 1, \ldots, a$, the mean $\bar{x}_{it}$ is calculated at each time point $t = 1, \ldots, d$. Then, the points $(1, \bar{x}_{i1}), \ldots, (d, \bar{x}_{id})$ are plotted for all groups $i = 1, \ldots, a$. An example can be seen in Figure 3. For plotting these curves, the S3 method `plot.HRM` is available. It needs an object of class `"HRM"` which is returned by the function `hrm_test`. Additionally we can change the labels for the x-axis and the y-axis with the parameters `xlab` and `ylab` respectively. Furthermore, it is possible to disable the legend by setting the argument `legend = FALSE`, the title for the legend can be changed with the argument `legend.title`. By setting `legend.title = NULL`, no legend title is displayed. Internally, this method is only a **ggplot2** wrapper (Wickham, 2009). If there is only one sub-plot but no whole-plot factor used, it is still possible to plot profile curves (see for example Figure 5).

For objects of class `"HRM"`, there are also the methods `print.HRM` and `summary.HRM` available. The first method just reproduces the standard output from `hrm_test` whereas `summary.HRM` lists additionally the whole- and sub-plot factors which were used in the model.

To improve the performance in case of a large dimension, this package uses mainly the dual empirical variance-covariance matrices (Brunner et al., 2012; Happ et al., 2017). Because the trace function is invariant under cyclic permutations, the empirical $d \times d$ variance-covariance matrix can be reduced to an $n_i \times n_i$ matrix. This is especially useful if $d >> n_i$, where n_i is the sample size in the i-th group. Another improvement is to use

$$\operatorname{tr}\left(\widehat{\blacksquare}_i \widehat{\blacksquare}_j\right) = 1_d' \left(\widehat{\blacksquare}_i * \widehat{\blacksquare}_j'\right) 1_d$$

for calculating the trace of a matrix product, where $*$ is the Hadamard-Schur product. The effects of these two improvements are only noticeable for calculating a single test statistic if d is very large ($d > 1000$).

An additional enhancement for high-dimensional data is achieved by using the method `hrm_test.list` instead of `hrm_test.data.frame`. Although it can sometimes be convenient to use the long table format in a data frame, for computing the degrees of freedom and the test statistic, the data has to be separated internally by the whole-plot factors. This procedure can be quite time consuming for very large datasets (dimension $d > 10^5$ and number of subjects > 100) if the data structure `"data.frame"` is

dimension d	`data.frame`	`data.table`
10^5	224.17	0.64
2×10^5	1411.93	1.07

Table 1: Running times in seconds for converting a wide table into a long table for 100 subjects.

used. The problem in this case is that the data structure `"data.frame"` is not suitable for large data sets. But most functions that import data into R return a `"data.frame"` object. In such a case, it is beneficial to provide the data already separated by the whole-plot factors in a list. This can be done with the method `hrm_test.list`. One disadvantage by using this method is, that it is currently not possible to specify more than one whole- or sub-plot factor. Another way to solve this problem is to convert the `"data.frame"` object into a `"data.table"` object (from package **data.table** (Dowle and Srinivasan, 2017)) which is specifically developed for large data sets. But the syntax for a data.table is completely different from a data.frame and their syntax is also not compatible with each other. In order to demonstrate the effect, we ran a small simulation with $d = 10^5$ and $n = 100$. Here, it takes about 4 minutes to convert a `data.frame` from the wide to the long table format and when we used a `data.table` it takes under 1 second to do the same (see Table 1). By increasing the dimension, the time it takes for the conversion increases exponentially if you use a `data.frame`. For this simulation we used the the function `Sys.time` to measure the calculation time and repeated each calculation five times. Note that the time needed for rearranging the data heavily depends on the computer on which the R script runs. In our case, we employed a computer with an AMD Ryzen 5 1600 CPU (6× 3.2 GHz) and 16GB DDR-4 RAM (2800 MHz). Therefore the results from this simulation cannot be generalized but it can be an indication that a data frame should not be rearranged if the dimension is large. Therefore, using the method `hrm_test.list` may help to avoid this additional step of converting your data frame into a `"data.table"` object.

Graphical user interface

For more convenient access to the package's capabilities, a graphical user interface (GUI) can be used by typing the command `hrm_GUI()` into the R console. A window opens where the data file (long table format) can be loaded and viewed. This window is shown in Figure 1. After typing the formula as described before and selecting the column identifying the subjects, the results can be displayed in a separate window. Optionally, the results can be saved as a LaTeX-table or just as plain text. For each hypothesis test, the degrees of freedom of the F approximation, the value of the test statistic, the critical value for the test, and the p-value are displayed. An example of the results window for the EEG data is shown in Figure 2. This data set contains of 160 subjects who have been divided into one of four diagnostic groups (whole-plot factor). For each subject 40 measurements from an EEG are available (sub-plot factor). This example is explained in more detail in Section EEG Data Example. In the results window are the test statistics along with the degrees of freedom and p-values for main and interaction effects displayed.

If only one whole- and one sub-plot factor is being used, the profile curves of the groups are displayed as a graphic in a separate window. The plot can be saved in the pdf format.

The GUI relies on the function `hrm_test` for doing the analysis and `plot.HRM` for creating the plot as shown in Figure 3. That is, there is no difference between using the GUI or using both generic functions directly. For creating the GUI, the **RGtk2** package is used (Lawrence and Temple Lang, 2010). This package allows us to utilise the cross-platform widget-toolkit GTK+ with R. The current version 2.20.34 of this package cannot be installed on macOS, therefore the GUI for the package **HRM** does not work on macOS.

For opening data files, we rely on the functions provided in the package **tcltk** to ensure that this procedure works on multiple computer environments. For viewing the data we utilise the package **RGtk2Extras** (Taverner et al., 2012) which provides a very easy way do display and manipulate data in a GUI. The packages **cairoDevice** and **ggplot2** are used for plotting (Lawrence, 2017; Wickham, 2016) profile curves. Similar as before, we have to avoid functions that only work for a specific computer environment. Therefore we work with the package **cairoDevice** which is capable of displaying a graphic in a window and does not depend on a specific computer environment. For saving the results of the function `hrm_test` as a LaTeX-table, we are using the package **xtable** (Dahl, 2016) which converts a `data.frame` into the corresponding LaTeX code for a table.

Because of the limitation of **RGtk2**, the GUI does not work on macOS (OS X 10.11.6). But the package **HRM** itself can be installed on macOS and the function `hrm_test` can be run regardless. Apart from that, the GUI works on all other newer operating systems (for example Debian 7.3.0-12 or

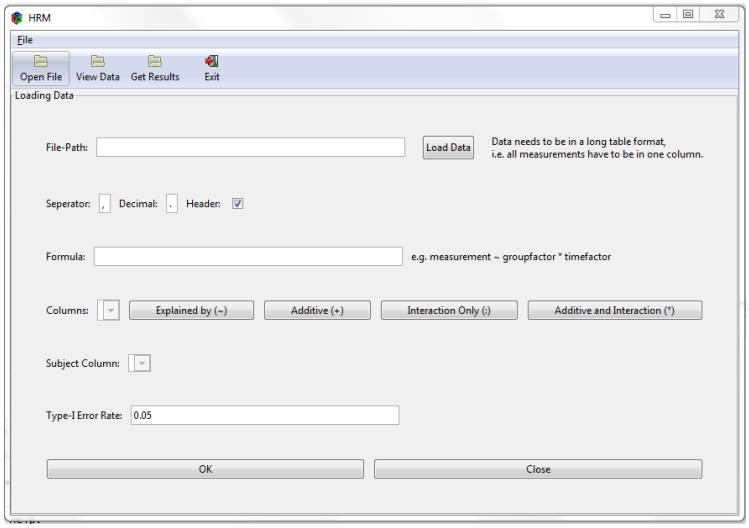


Figure 1: GUI for the **HRM** package. The data can be loaded either by clicking 'Open File' in the toolbar or by clicking on the button 'Load Data' next to the textfield for the file path. The formula has to be specified in the appropriate textfield and the subject column needs to be chosen in the dropdown list.

Microsoft Windows 7, 8, 10).

Examples

EEG Data Example

In order to demonstrate how to work with the package, we use the EEG data from Staffen et al. (2014) which is included in the **HRM** package. There are 160 subjects who have been diagnosed with Alzheimer's disease (AD), mild cognitive impairments (MCI), or subjective cognitive complaints (SCC+ or SCC–). For each subject, four variables (activity, complexity, mobility and brain rate) are measured at 10 different brain regions (frontal left/right, parietal left/right, central left/right, temporal left/right, and occipital left/right). We are using the diagnostic groups and sex as whole-plot factors, and the region and variable as sub-plot factors. Sample sizes are given in Table 2. In all combinations of diagnostic group and sex the dimension is larger than the sample size, therefore this is considered a high-dimensional setting.

Sex	AD	MCI	SCC+	SCC-
male	12	27	14	6
female	24	30	31	16
sum	36	57	45	22

Table 2: Sample sizes for the EEG data.

To perform the analysis we are using the graphical user interface and the R console. The commands

```
# Save EEG data from package HRM
write.table(EEG,
        file = "EEG.csv",
        sep = ",",
        dec = ".",
        col.names = TRUE)
# Open the GUI
hrm_GUI()
```

will load the package and save the EEG data in a '.csv' file in the current working directory. The last line launches the GUI (see Figure 1). By clicking on the 'Open File' button we can select the 'EEG.csv' file we previously saved. Depending on the data, we may have to change the separator mark, decimal mark, or deselect that the file contains header. This can be done in the fields below the textfield for

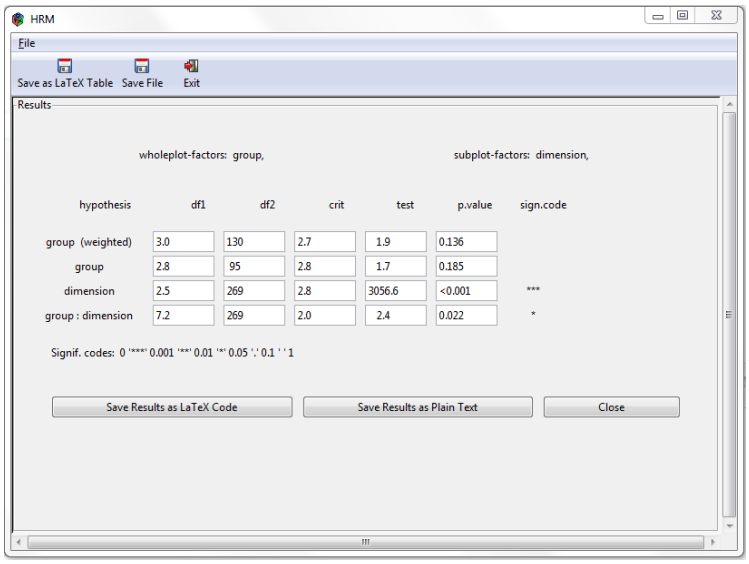


Figure 2: Results for the EEG data. The results can be saved as a LaTeX table by clicking on the button on the bottom or in the toolbar.

the file path. We can look at the data by clicking on the button 'View Data' in the toolbar. A new window is created displaying the data we have just loaded (seen in Figure 4). The data consists of columns specifying the whole-plot (`group`, `sex`) and sub-plot factors (`variable`, `region`). Additionally, one column contains the response variable from each person (`value`) and the subject column identifies the subjects. A unique identification of the subjects is necessary to determine automatically within the function `hrm_test` which factors are whole- and which are sub-plot factors. By combining the factors region and variable into just one factor, we obtain a new factor which we will simply call dimension. It is provided in the column with the same name.

Another way to look at the data is by using the command

```
View(EEG)
```

which will also display the complete data set. After we have loaded and looked at the data we need to specify the formula in the GUI. In this example, the formula is given by

```
value ~ group * sex * region * variable
```

which means that for the four factors tests for main and interaction effects will be performed. If we were only interested in testing the main effects we could use the formula

```
value ~ group + sex + region + variable
```

where we are using + instead of *. If we simply want to test interaction effects we could use : instead. In addition to specifying the formula, we need to select which column identifies the subjects. This is done by selecting the 'subject' column in the dropdown list. To perform the tests, we simply click on the 'Ok' button on the bottom of the window or the 'Get Results' button in the toolbar. A new window (see Figure 2) will open which contains the results of the performed tests. For each tested hypothesis, the degrees of freedom, the value of the test statistic, and the p-value are displayed. Above the results, a summary about the factors used in the analysis is provided. To perform these tests manually, we may use the commands

```
r <- hrm_test(formula = value ~ group*sex*region*variable,
              data = EEG,
              subject = "subject")
library(xtable)
print(xtable(r$result), include.rownames = FALSE)

% latex table generated in R 3.4.1 by xtable 1.8-2 package
\begin{table}[ht]
        \centering
        \begin{tabular}{lrrrrrl}
                \hline
                hypothesis & df1 & df2 & crit & test & p.value & sign.code \\
```

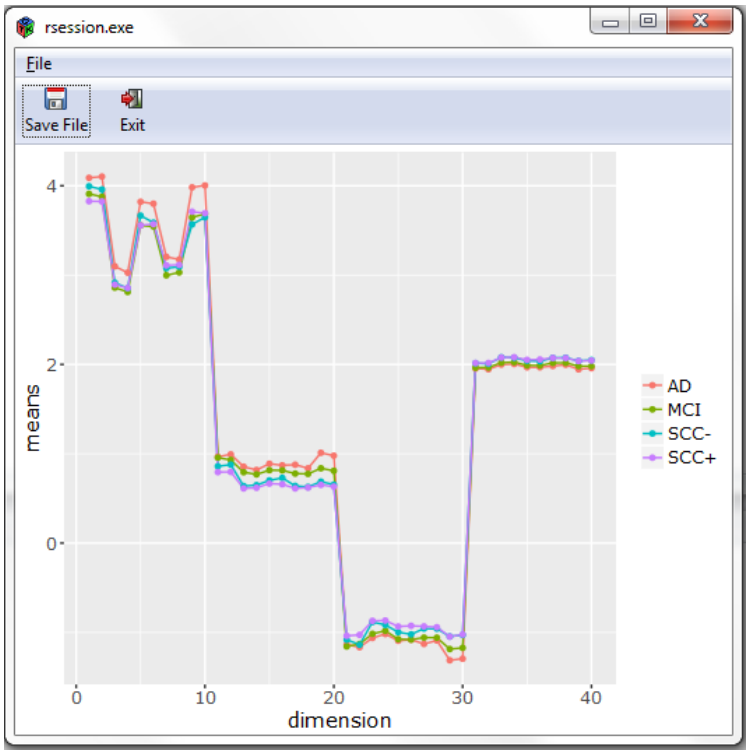
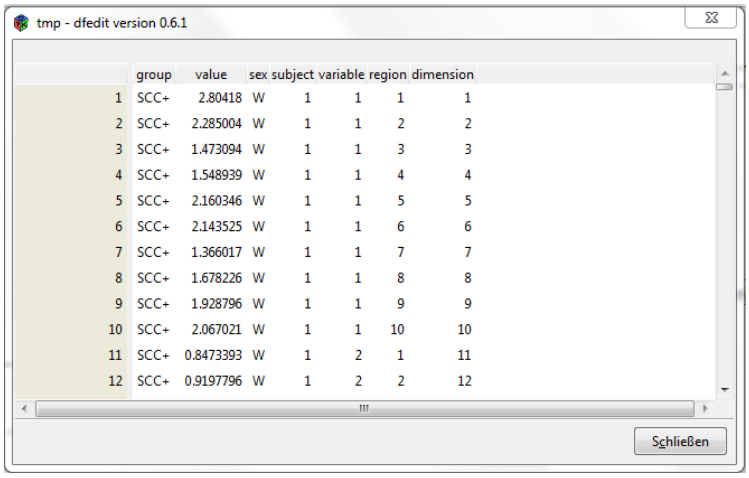


Figure 3: Profile Plot for the EEG data. The image can be saved by clicking on 'Save File'.

```
\hline
group & 3.04 & 116.29 & 2.67 & 1.66 & 0.18 &  \\
sex & 1.00 & 116.29 & 3.92 & 6.05 & 0.02 & * \\
region & 5.59 & 246.82 & 2.18 & 196.60 & 0.00 & *** \\
variable & 1.51 & 178.72 & 3.36 & 4930.69 & 0.00 & *** \\
group:sex & 3.04 & 116.29 & 2.67 & 0.18 & 0.91 &  \\
group:region & 14.09 & 246.82 & 1.73 & 0.80 & 0.67 &  \\
group:variable & 4.52 & 178.72 & 2.33 & 3.34 & 0.01 & ** \\
sex:region & 5.59 & 246.82 & 2.18 & 0.99 & 0.43 &  \\
sex:variable & 1.51 & 178.72 & 3.36 & 4.89 & 0.02 & * \\
region:variable & 7.76 & 299.89 & 1.99 & 143.58 & 0.00 & *** \\
group:sex:region & 14.09 & 246.82 & 1.73 & 0.58 & 0.88 &  \\
group:sex:variable & 4.52 & 178.72 & 2.33 & 0.26 & 0.92 &  \\
group:region:variable & 18.75 & 299.89 & 1.63 & 1.04 & 0.41 &  \\
sex:region:variable & 7.76 & 299.89 & 1.99 & 1.26 & 0.27 &  \\
```



	group	value	sex	subject	variable	region	dimension
1	SCC+	2.80418	W	1	1	1	1
2	SCC+	2.285004	W	1	1	2	2
3	SCC+	1.473094	W	1	1	3	3
4	SCC+	1.548939	W	1	1	4	4
5	SCC+	2.160346	W	1	1	5	5
6	SCC+	2.143525	W	1	1	6	6
7	SCC+	1.366017	W	1	1	7	7
8	SCC+	1.678226	W	1	1	8	8
9	SCC+	1.928796	W	1	1	9	9
10	SCC+	2.067021	W	1	1	10	10
11	SCC+	0.8473393	W	1	2	1	11
12	SCC+	0.9197796	W	1	2	2	12

Figure 4: EEG data used in the example.

```
                group:sex:region:variable & 18.75 & 299.89 & 1.63 & 0.74 & 0.77 & \\
                \hline
        \end{tabular}
\end{table}
```

to obtain the results stated here. The command `xtable` is used to convert a `"data.frame"` object into a LaTeX table. This is also possible with the GUI. In the toolbar of the results window there is a button labeled with 'Save as LaTeX table'. By clicking on it we can select a path to save a '.tex' file containing the results.

As an alternative formula, we may use the following.

```
value ~ group  * dimension
```

Here, only one whole- and one sub-plot factor are specified. After clicking on the 'Ok' button, two windows will open. In the first window, the results of the test statistics are displayed. The other window shows a plot of the profiles (see Figure 3). We can also create these plots by using the function

```
plot(hrm_test(value ~ group*dimension, data = EEG, subject = "subject"))
```

Here the function `hrm_test` returns an object of class `"HRM"` and the function `plot` is applied to an object of such a class. If we calculate the test statistics in this two-factor example and save the results again as a table, we obtain the following:

```
hrm_test(formula = value ~ group*dimension,
                data = EEG,
                subject = "subject")

Call:
value ~ group * dimension

hypothesis              df1       df2     crit        test    p.value  sign.code
group  (weighted)   2.993584 130.10764 2.676095    1.881486 0.13600798
group               2.779981  95.00304 2.768161    1.657329 0.18482432
dimension           2.504068 268.94339 2.803050 3056.569162 0.00000000  ***
group : dimension   7.155131 268.94339 2.031723    2.376905 0.02164946   *

Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

T-cell Activation Example

It is also possible to use the function `hrm_test` with only one sub-plot factor and no whole-plot factor. A data example with such a design is included in the package **longitudinal** (Opgen-Rhein and Strimmer., 2015) and is originally taken from Rangel et al. (2004). The data are from a microarray time series for T-lymphocytes activation. Here, the expression levels of $n = 58$ genes are measured at $d = 100$ time points ($d > n$), thus presenting a high-dimensional situation. In order to analyse the data, we need to load the necessary packages first and then convert the data into the long table format. At the end, we can use the function `hrm_test` which detects a significant time effect. Alternatively, we could have used the GUI. But for this approach we would have needed to save the data as a '.csv' file first because the GUI only supports loading data sets from external files.

This data example is just used as a demonstration for the package. From a statistical point of view, the gene expressions do not necessarily have to be independent, as different genes may have an influence on each other. For example in eukaryotes, gene expression is highly regulated. Some genes encode for transcription factors (for example zinc-fingers) which can up- or down-regulate the expression of other genes. With regard to this data example the gene FYB seems to influence the expression of several interleukin receptor genes and GATA-3 (a zinc-finger transcription factor) (Rangel et al., 2004). Therefore, it is very plausible that the assumption of independent genes is violated for this data example. However, for the illustration of the method we are making the simplifying assumption that observations on different 'subjects' (genes, in this case) are independent from each other.

```
library(longitudinal)
library(tidyr)
library(HRM)

# transforming the data into the long table format
data(tcell)
```

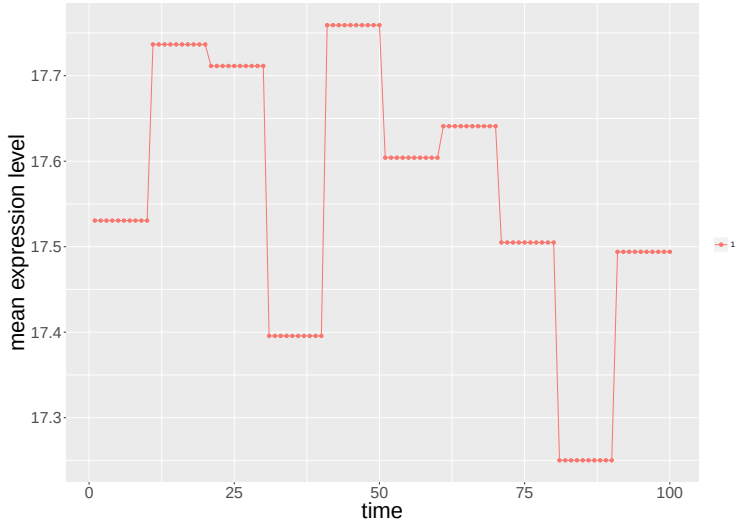


Figure 5: Profile curve.

```
data <- t(print(tcell.10))
data <- as.data.frame(data)
data$subject = 1:58
data_long <- gather(data, time, measurement, 1:100, factor_key = TRUE)

hrm_test(data_long, measurement ~ time, subject = "subject")

Call:
measurement ~ time

hypothesis      df1      df2     crit     test      p.value    sign.code
time         5.514332 314.3169 2.179376 6.268063 6.538363e-06     ***

Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

To plot the profile curve for this data set, we simply apply the `plot` function to an `"HRM"` object returned by the function `hrm_test`. By setting the optional argument `legend` to `FALSE`, no legend is added to the plot (see Figure 5).

```
object_hrm <- hrm_test(measurement ~ time, data = data_long, subject = "subject")
plot(object_hrm, legend = FALSE, xlab = "time", ylab = "mean expression level")
```

For the `plot.HRM` method it is also possible to use additional arguments which will be added to the `"ggplot2"` object. As an example we want to use the `theme_bw` style with no legend. Here we need to take into consideration that `theme_bw` overwrites everything related to the legend already specified in `plot.HRM`. Therefore, we need to specify separately that a legend should not be displayed. The code

```
plot(object_hrm, ... = theme_bw() +
        theme(legend.title = element_blank(), legend.position = "none") +
        theme(axis.text = element_text(size=18), axis.title = element_text(size = 25)) )
```

produces Figure 6 with a white background and no legend. Because `plot.HRM` simply returns an `"ggplot2"` object, we also could have used the following code.

```
plot(object_hrm) + theme_bw() +
        theme(legend.title = element_blank(), legend.position = "none") +
        theme(axis.text = element_text(size = 18), axis.title = element_text(size = 25))
```

That is we can pass additional arguments through `...` in the method `hrm.plot` or, even easier, just add the additional arguments directly to the object to manipulate the graphic.

Limitations

In Section Statistical model, we have already stated the mathematical assumptions for the methods implemented in the **HRM** package. We are assuming normally distributed observations from independent subjects. As long as the data distribution is not 'too far' from a normal distribution, the test works

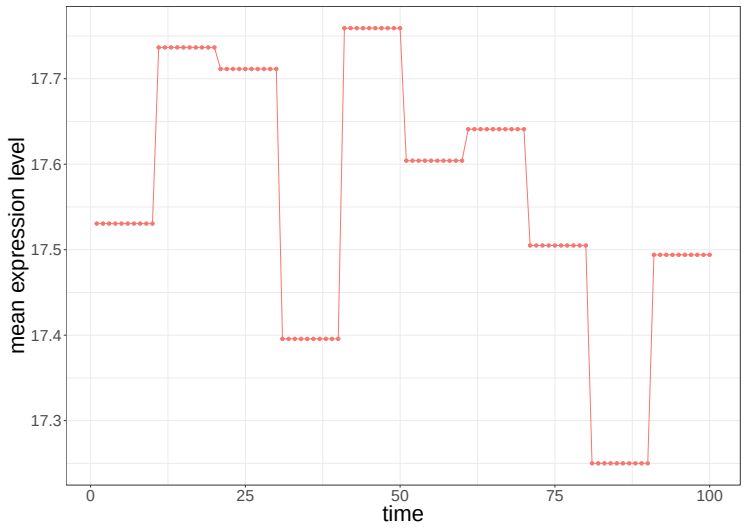


Figure 6: Profile curve generated with additional arguments for **ggplot2**.

fine. However, for very skewed or heavy-tailed distributions, the test may in general not maintain the type-I error rate. This could especially be a problem in combination with heteroscedastic data and unbalanced groups. Note that this limitation still holds for large sample sizes because the described method is only an approximation and not an asymptotically exact test. This limitation also applies in particular to the case of multivariate normally distributed observations. Therefore it is important to check the assumptions or do some simulations to see whether the test is robust against some violations of the assumptions. As we see in the following simulations, the test statistic is mainly affected by skewed or heavy-tailed distributions but works well for data generated by a binomial distribution. For our two examples used in Section Examples they do not satisfy the assumptions. For the genetic data, the gene expressions (subjects) are in general correlated. The EEG data seems to violate the normality assumption as some QQ-plots from marginal distributions show larger deviations from a straight line. But it is very difficult to check these distributional assumptions in general for high-dimensional data sets. In our case we only use those data sets for demonstration purposes but for real analyses one should be careful about the assumptions, as the type-I error can be quite inflated which we can see in the subsequent simulations.

In order to illustrate the behaviour of the proposed method when its formal assumptions are violated, we have performed several simulations for testing for the main group effect. For all simulations we used the dimension $d = 20$ and performed 10^4 simulation runs. We have already observed that the test statistic does not perform well for skewed data. Therefore we concentrated in this simulation study on the exponential and log-normal distribution. But we have also considered a discrete distribution, as in such a case the test seems to perform very well.

For the first simulation, we defined the random variables $X_{1jk} = \epsilon_{1j} + \delta_1$ for $j = 1, \ldots, 20$ and $k = 1, \ldots, n_1 = 20$. In the second group we chose $X_{2jk} = \epsilon_{2j} + \delta_2$ for $j = 1, \ldots, 20$ and $k = 1, \ldots, n_1 = 30$. Here, $\delta_1 \sim \mathcal{E}(1)$, $\delta_2 \sim \mathcal{E}(2)$, $\epsilon_{1j} \sim \mathcal{E}(1)$ and $\epsilon_{2j} \sim \mathcal{E}(2)$, where $\mathcal{E}$ refers to the exponential distribution. Then we shifted the second group equally for all 20 time points such that both groups have the same mean which means that the null hypothesis of no main group effect holds. For this setting, the simulated type-I error rate was 0.0688. In the second simulation we used for the second group $\mathcal{E}(4)$ distributions. In this case, the simulated type-I error rate was 0.0748. Additionally, we used different sample sizes to see the impact of negative or positive pairing. The results are displayed in Table 3. For all these settings, the simulated type-I error was slightly inflated. We performed similar simulations using log-normal distributions. Here, we considered $\delta_1 \sim LN(0,1)$, $\delta_2 \sim LN(0,\theta)$, $\epsilon_{1j} \sim LN(0,1)$ and $\epsilon_{2j} \sim LN(0,\theta)$ with $\theta = \frac{1}{2}, 2$. Both groups have been shifted such that they have the same mean. The results in Table 4 show heavily inflated type-I error rates (up to over 30%).

In a third simulation, we considered binomial distributions as an example for discrete data. Here, we defined $\delta_1, \epsilon_{2j} \sim Bin(m, 0.5)$ and $\delta_1, \epsilon_{2j} \sim Bin(m, 0.9)$ for $m = 5, 10, 15$ and for sample sizes $n_1 = 20, 15$ and $n_2 = 15, 20$. The other simulation parameters were the same as for the other two simulations. That is, the dimension was 20 and all random variables have been shifted to mean zero such hat the null hypothesis holds. The results are shown in Table 5. For binomial data, the method seemed to work very well even for heteroscedastic and unbalanced groups. Furthermore, the value of m did not seem to influence the result very much and even in the case of Bernoulli distributions ($m = 1$) the simulated type-I error rate was close to the nominal level of 5%.

Therefore the package HRM should not be used with distributions similar to log-normal or exponential distributions, that is very skewed or heavy-tailed distributions. Instead, in those cases, we recommend using a nonparametric test such as the rank-based ANOVA type statistic or a similar method.

θ	Sample sizes	Type-I error rate
2	20, 30	0.0688
4	20, 30	0.0748
2	30, 20	0.0543
4	30, 20	0.0685
2	25, 25	0.0616
4	25, 25	0.0702

Table 3: Type-I error rate simulation with different settings for testing for the main group effect. We used centred exponential distributions with parameter $\theta = 1$ in the first group and $\theta = 2, 4$ for the second group, and the dimension was $d = 20$.

θ	Sample sizes	Type-I error rate
0.5	20, 30	0.1157
2	20, 30	0.2927
0.5	30, 20	0.0917
2	30, 20	0.3126
0.5	25, 25	0.1032
2	25, 25	0.2974

Table 4: Type-I error rate simulation using log-normal distributions for testing for the main group effect. For the first group we used centred $LN(0, 1)$ and for the second group centred $LN(0, \theta)$ distributions, and the dimension was $d = 20$.

m	Sample sizes	Type-I error rate
1	20, 15	0.0488
5	20, 15	0.0476
10	20, 15	0.0503
15	20, 15	0.0467
1	15, 20	0.0515
5	15, 20	0.0485
10	15, 20	0.0479
15	15, 20	0.0489

Table 5: Type-I error rate simulation using binomial distributions for testing for the main group effect. For the first group we used centred $Bin(m, 0.5)$ and for the second group we used centred $Bin(m, 0.9)$ random variables, and the dimension was $d = 20$.

Another drawback of our method is that it is not invariant under scale transformations of individual variables. This means that changing the scale in one dimension (using m/s instead of km/h) may lead to a different result because the observations are not standardized to avoid the problem of singular empirical covariance matrices. Therefore the same scale should be used for all dimensions.

The package **MANOVA.RM** by Friedrich et al. (2017b) provides several asymptotically exact tests that do not have these limitations (see for example Friedrich and Pauly (2018)). However, not all have been extended to high-dimensional settings (Friedrich et al., 2017a) and some may also require a resampling step which may take a while to compute for large data sets. But this might only become a real problem if you want to simulate the power of the test for multiple situations. The ANOVA type statistic mentioned before has the problem that it gets quite conservative when the dimension is large (Brunner et al., 2002; Bathke et al., 2009; Friedrich et al., 2017a), and additionally, the test is also not invariant under scale transformations. Friedrich et al. (2017a) proposed a potential solution for this problem in the low-dimensional case.

Conclusions

The aim of the **HRM** package is to provide an easy to use way to do inference on high-dimensional repeated measures. To that end, a graphical user interface has also been implemented in this package. With the GUI, it is possible to load and view the data, plot the profile curves and save the results of hypothesis tests as a table which can be inserted very easily into a LaTeX document. The GUI is optional, that is, all of its functions can be used directly. There are different functions to allow for data in the long or wide table format. Although both formats can be transformed easily into each other, not having to do it is more convenient for users. This applies especially to statistics practitioners who are not expert R users.

The package currently supports up to four factors. If data in the wide table format are used, the maximum number of factors that can be used is two. For the long table format, it is possible to use up to two whole-plot factors and up to three sub-plot factors, but the maximum number of factors is four. Table 6 summarizes the minimum and maximum number of factors that can be used.

Table Format	Whole-plot	Sub-plot	Total
long table	0–2	0–3	1–4
wide table	1	1	2

Table 6: Minimum and maximum number of factors which can be used with the package **HRM**.

There are still a few limitations to the method implemented in the package. Improvements to the test statistic to get rid of these limitations are part of future work and will also be implemented in the package. For example, a nonparametric version of the test statistic will be provided in the future by setting an additional argument `nonparametric = TRUE` for the function `hrm_test`. This new test will use (pseudo)-ranks and can be applied to metric or non-metric data. Additionally, the package will be further improved to increase the performance for large data.

Funding

The research was supported by Austrian Science Fund (FWF) I 2697-N31.

Bibliography

A. C. Bathke, O. Schabenberger, R. D. Tobias, and L. V. Madden. Greenhouse-Geisser adjustment and the ANOVA-type statistic: Cousins or twins? *The American Statistician*, 63(3):239–246, 2009. URL https://doi.org/10.1198/tast.2009.08187. [p534, 545]

A. C. Bathke, S. Friedrich, M. Pauly, F. Konietschke, W. Staffen, N. Strobl, and Y. Höller. Testing mean differences among groups: Multivariate and repeated measures analysis with minimal assumptions. *Multivariate Behavioral Research*, 0(0):0–0, 2018. URL https://doi.org/10.1080/00273171.2018.1446320. PMID: 29565679. [p535]

G. Box. Some theorems on quadratic forms applied in the study of analysis of variance problems, ii. effects of inequality of variance and of correlation between errors in the two-way classification. *The Annals of Mathematical Statistics*, 25(3):484–498, 1954a. URL https://doi.org/10.1214/aoms/1177728717. [p534]

G. E. Box. Some theorems on quadratic forms applied in the study of analysis of variance problems, i. effect of inequality of variance in the one-way classification. *The Annals of Mathematical Statistics*, 25 (2):290–302, 1954b. URL https://doi.org/10.1214/aoms/1177728786. [p534]

E. Brunner and M. L. Puri. Nonparametric methods in factorial designs. *Statistical Papers*, 42(1):1–52, 2001. URL https://doi.org/10.1007/s003620000039. [p535]

E. Brunner, H. Dette, and A. Munk. Box-type approximations in nonparametric factorial designs. *Journal of the American Statistical Association*, 92(440):1494–1502, 1997. URL https://doi.org/10.1080/01621459.1997.10473671. [p535]

E. Brunner, S. Domhof, and F. Langer. *Nonparametric Analysis of Longitudinal Data in Factorial Designs*. John Wiley & Sons, 2002. [p545]

E. Brunner, A. C. Bathke, and M. Placzek. Estimation of Box's ε for low-and high-dimensional repeated measures designs with unequal covariance matrices. *Biometrical Journal*, 54(3):301–316, 2012. URL https://doi.org/10.1002/bimj.201100160. [p536, 537]

S. X. Chen, Y.-L. Qin, and others. A two-sample test for high-dimensional data with applications to gene-set testing. *The Annals of Statistics*, 38(2):808–835, 2010. URL https://dx.doi.org/10.1214/09-AOS716. [p535]

D. B. Dahl. *xtable: Export Tables to LaTeX or HTML*, 2016. URL https://CRAN.R-project.org/package=xtable. R package version 1.8-2. [p538]

M. Dowle and A. Srinivasan. *data.table: Extension of 'data.frame'*, 2017. URL https://CRAN.R-project.org/package=data.table. R package version 1.10.4-3. [p538]

S. Friedrich and M. Pauly. MATS: Inference for potentially singular and heteroscedastic MANOVA. *Journal of Multivariate Analysis*, 165:166 – 179, 2018. ISSN 0047-259X. URL https://doi.org/10.1016/j.jmva.2017.12.008. [p545]

S. Friedrich, E. Brunner, and M. Pauly. Permuting longitudinal data in spite of the dependencies. *Journal of Multivariate Analysis*, 153:255 – 265, 2017a. ISSN 0047-259X. URL https://doi.org/10.1016/j.jmva.2016.10.004. [p545]

S. Friedrich, F. Konietschke, and M. Pauly. *MANOVA.RM: Analysis of Multivariate Data and Repeated Measures Designs*, 2017b. URL https://CRAN.R-project.org/package=MANOVA.RM. R package version 0.1.2. [p545]

S. Geisser and S. W. Greenhouse. An extension of Box's results on the use of the F distribution in multivariate analysis. *The Annals of Mathematical Statistics*, 29(3):885–891, 1958. URL https://doi.org/10.1214/aoms/1177706545. [p534]

S. W. Greenhouse and S. Geisser. On methods in the analysis of profile data. *Psychometrika*, 24(2):95–112, 1959. URL https://doi.org/10.1007/BF02289823. [p534]

M. Happ, S. W. Harrar, and A. C. Bathke. Inference for low-and high-dimensional multigroup repeated measures designs with unequal covariance matrices. *Biometrical Journal*, 58(4):810–830, 2016. URL https://doi.org/10.1002/bimj.201500064. [p535, 536]

M. Happ, S. W. Harrar, and A. C. Bathke. High-dimensional repeated measures. *Journal of Statistical Theory and Practice*, 11(3):468–477, 2017. URL https://doi.org/10.1080/15598608.2017.1307792. [p535, 536, 537]

S. W. Harrar. Asymptotics for tests on mean profiles, additional information and dimensionality under non-normality. *Journal of Statistical Planning and Inference*, 139(8):2685 – 2705, 2009. ISSN 0378-3758. URL https://doi.org/10.1016/j.jspi.2008.12.008. [p535]

S. W. Harrar and X. Kong. High-dimensional multivariate repeated measures analysis with unequal covariance matrices. *Journal of Multivariate Analysis*, 145:1 – 21, 2016. ISSN 0047-259X. URL https://doi.org/10.1016/j.jmva.2015.11.012. [p535]

H. Huynh and L. S. Feldt. Conditions under which mean square ratios in repeated measurements designs have exact F-distributions. *Journal of the American Statistical Association*, 65(332):1582–1589, 1970. URL https://doi.org/10.1080/01621459.1970.10481187. [p534]

H. Huynh and L. S. Feldt. Estimation of the Box correction for degrees of freedom from sample data in randomized block and split-plot designs. *Journal of Educational Statistics*, 1(1):69–82, 1976. URL https://doi.org/10.3102/10769986001001069. [p534]

F. Konietschke, A. C. Bathke, S. W. Harrar, and M. Pauly. Parametric and nonparametric bootstrap methods for general MANOVA. *Journal of Multivariate Analysis*, 140:291 – 301, 2015. ISSN 0047-259X. URL https://doi.org/10.1016/j.jmva.2015.05.001. [p535]

M. Lawrence. *cairoDevice: Embeddable Cairo Graphics Device Driver*, 2017. URL https://CRAN.R-project.org/package=cairoDevice. R package version 2.24. [p538]

M. Lawrence and D. Temple Lang. RGtk2: A graphical user interface toolkit for R. *Journal of Statistical Software*, 37(8):1–52, 2010. URL http://www.jstatsoft.org/v37/i08/. [p538]

B. Lecoutre. A correction for the $\tilde{\varepsilon}$ approximate test in repeated measures designs with two or more independent groups. *Journal of Educational Statistics*, 16(4):371–372, 1991. URL https://doi.org/10.3102/10769986016004371. [p534]

R. Opgen-Rhein and K. Strimmer. *longitudinal: Analysis of Multiple Time Course Data*, 2015. URL https://CRAN.R-project.org/package=longitudinal. R package version 1.1.12. [p542]

M. Pauly, D. Ellenberger, and E. Brunner. Analysis of high-dimensional one group repeated measures designs. *Statistics*, 49(6):1243–1261, 2015. URL https://doi.org/10.1080/02331888.2015.1050022. [p535, 536]

C. Rangel, J. Angus, Z. Ghahramani, M. Lioumi, E. Sotheran, A. Gaiba, D. L. Wild, and F. Falciani. Modeling t-cell activation using gene expression profiling and state-space models. *Bioinformatics*, 20 (9):1361–1372, 2004. URL https://doi.org/10.1093/bioinformatics/bth093. [p542]

P. Sattler and M. Pauly. Inference for high-dimensional split-plot-designs: A unified approach for small to large numbers of factor levels. *arXiv preprint arXiv:1706.02592*, 2017. [p536]

P. Secchi, A. Stamm, and S. Vantini. Inference for the mean of large p small n data: A finite-sample high-dimensional generalization of Hotelling's theorem. *Electronic Journal of Statistics*, 7:2005–2031, 2013. URL https://doi.org/10.1214/13-EJS833. [p535]

M. S. Srivastava. Multivariate theory for analyzing high dimensional data. *JOURNAL OF THE JAPAN STATISTICAL SOCIETY*, 37(1):53–86, 2007. URL https://doi.org/10.14490/jjss.37.53. [p535]

M. S. Srivastava and H. Yanagihara. Testing the equality of several covariance matrices with fewer observations than the dimension. *Journal of Multivariate Analysis*, 101(6):1319 – 1329, 2010. ISSN 0047-259X. URL https://doi.org/10.1016/j.jmva.2009.12.010. [p535]

W. Staffen, N. Strobl, H. Zauner, Y. Höller, J. Dobesberger, and E. Trinka. Combining spect and eeg analysis for assessment of disorders with amnestic symptoms to enhance accuracy in early diagnostics. *Poster A19 Presented at the 11th Annual Meeting of the Austrian Society of Neurology. 26th-29th March 2014, Salzburg, Austria.*, 2014. [p539]

T. Taverner, with contributions from John Verzani, I. Conde, and L. Andronic. *RGtk2Extras: Data Frame Editor and Dialog Making Wrapper for RGtk2*, 2012. URL https://CRAN.R-project.org/package=RGtk2Extras. R package version 0.6.1. [p538]

H. Wickham. *Ggplot2: Elegant Graphics for Data Analysis*. Springer-Verlag, 2009. ISBN 978-0-387-98140-6. URL http://ggplot2.org. [p537]

Martin Happ
University of Salzburg
Department of Mathematics
5020 Salzburg
Austria
martin.happ@sbg.ac.at

Solomon W. Harrar
University of Kentucky
Department of Statistics
Lexington, KY 40536
USA
solomon.harrar@uky.edu

Arnce C. Bathke
University of Salzburg
Department of Mathematics
5020 Salzburg
Austria
arne.bathke@sbg.ac.at

Conference Report: eRum 2018

by Gergely Daróczi

Conference Summary

The European R Users Meeting (eRum) is an international conference that aims at bringing together users of the R language living in Europe – in the years when the useR! conference is hosted outside of the continent.

The first eRum conference was held in 2016 in Poznan, Poland with around 250 attendees and 20 sessions spanning over 3 days, including more than 80 speakers. Around that time, we also held a smaller conference in Budapest: the first satRday event happened with 25 speakers and almost 200 attendees from 19 countries in 2016.

The eRum 2018 conference is heritage of these two successful conferences: originally planned for 400-500 attendees from all around Europe, we ended up having 450 registrations from 39 countries – mostly from Germany and Hungary (15%), Poland, Austria, Netherlands and United Kingdom (10%), many other European countries, and for example Argentina, South Africa and New Zealand as well.

Tutorials

The conference started on May 14, 2018 with 3-6 hours workshops in the morning and in the afternoon as well on 7 parallel tracks – including topics from machine learning, writing and improving R packages, data visualization, handling spatial data and text mining. This day was hosted at the Central European University.

Scientific Program

The great success of the conference was mainly due to the high quality lineup of speakers: Achim Zeileis, Martin Mächler, Nathalie Villa-Vialaneix, Stefano Maria Iacus and Roger Bivand were our keynotes speakers; Arthur Charpentier, Barbara Borges Ribeiro, Colin Gillespie, Erin LeDell, Henrik Bengtsson, Jeroen Ooms, Mark van der Loo, Matthias Templ, Olga Mierzwa-Sulima, Przemyslaw Biecek and Szilard Pafka presented invited talks on the two main days of the conference on May 15-16, 2018 in the Akvarium Klub that is a cultural center located under a pool, in the center of Budapest.

The contributed presentations were picked from the more than 150 submissions that we have received for the Call for Papers, and the final conference program included 30 regular talks (18 mins), 24 lightning talks (5 mins), 8 Shiny demos and 22 posters.

The oral presentations were split into two parallel sessions (except for the keynotes), and all the talks were live-streamed and the video recordings are still available on the Hungarian RUG's YouTube channel along with the presented slides on the conference homepage.

Social Programs

The Welcome Reception offered networking opportunity for the more than 400 attendees with a light dinner, also combined with a poster session and a Shiny demo session in a concert room with a huge LED wall. The Conference Dinner was hosted on a boat cruise showing the riverside of the Danube both at daytime and on the way back at night as well, and some traditional Hungarian dishes were served with a nice selection of wines. The local R-Ladies chapter also hosted a meetup with almost 100 attendees featuring 6 talks and a networking opportunity with pizza and drinks to all interested parties.

Sponsors

As the conference ticket prices were kept as low as possible to make it affordable to all attendees (eg early-bird student tickets were priced at $50 for 3 days including catering), we are grateful to all our Platinum (RStudio), Gold (Mango Solutions, Microsoft, Quantide), Silver (H2O.ai, Emarsys, Open Analytics, Budapest BI Forum) and Bronze (WLOG Solutions, Jumping Rivers, R Consosritum, Upshift R Kft, R-bloggers) sponsors for contributing almost 1/4 of the overall conference budget.

Organizers

The Program Committee included 15 members form the Hungarian R User Group, organizers of eRum 2016, R Forwards, R Ladies and other European R User Groups: Adolfo Alvarez (Poland), Ágnes Salánki (Hungary), Andrew Lowe (Hungary), Bence Arató (Hungary), Branko Kovač (Serbia), Eszter Windhager-Pokol (Hungary), Gergely Daróczi (Hungary), Heather Turner (UK), Kevin O'Brien (Ireland), Imre Kocsis (Hungary), László Gönczy (Hungary), Maciej Beresewicz (Poland), Mariachiara Fortuna (Italy), Przemyslaw Biecek (Poland) and Szilárd Pafka (USA). The local Organizing Committee was lead by Gergely Daróczi, and the legal and accounting background was provided by Upshift R Kft for this nonprofit conference.

Further Information

- Conference homepage: http://2018.erum.io
- Twitter account: https://twitter.com/erum2018
- Talk abstracts, video recordings and slides: http://2018.erum.io/#talk-abstracts
- YouTube playlist of all eRum 2018 talks: https://www.youtube.com/watch?list=PLUBl0DoLa5SAo_XRnkQA5GtEORg9K7kMh

Gergely Daróczi
Budapest Users of R Network, Hungary
ORCiD: 0000-0003-3149-8537
daroczig@rapporter.net

R Day report

by Fernando P. Mayer, Walmes M. Zeviani, Wagner H. Bonat, Elias T. Krainski, Paulo J. Ribeiro Jr

About

R Day[1] - National Meeting of R Users, took place on May, 22, 2018 at Federal University of Paraná (UFPR), Curitiba, Brazil. It was the first event in Brazil endorsed by The R Foundation.

The event

R Day was planned as a satellite event of the 63rd RBRAS Annual Meeting[2] of the Brazilian Region of The International Biometric Society[3]. Paraná Federal University (UFPR) was selected on 2016 to host the meeting.

The R Day was considered as a UFPR *extension event*, and as such it was non-profitable and with no registration fees. There were 295 subscribed participants. Figure 1 shows how they split between students, professional and others.

Figure 2 shows the spread of the participants' home locations, for 88 of those who have opted for make geocoding information available on the subscription form. There was quite a spread, with all of the five country regions represented, an encouraging achievement considering the continental distances within the country. The fact the event was an RBRAS satellite helped to gather a wider audience.

An important feature was that the event was meant to be guided by the community, opened for proposals for oral presentations and tutorials for which there were 25 and 14 proposals, respectively. Time and space restrictions allowed for the selections of 18 oral presentations and 8 tutorials contemplating a diversity of topics related to R.

Three speakers were invited by the organising committee. Paulo Justiniano Ribeiro Jr, senior lecture at the Department of Statistics at UFPR and LEG member gave an historical review mixed with his own experiences which are highly connected with the history of R in Brazil. The full video of his talk was made available at Youtube[4]. Figure 3 captures a moment of Professor Paulo's talk, reckon to be influential on the dissemination of R in Brazil.

Rondon de Andrade Porto, data scientist at the Conselho Nacional de Justiça - CNJ (National Justice Council), showed how R is used to generate a large volume of visualizations and dynamic reports on the CNJ.

The R Day final talk was made by Gabriela de Queiroz (via webconference) who can be seen in Figure 4. Gabriela is the founder of *R Ladies* and told us how it all started and how it became an important community for the diffusion of R in Brazil and worldwide. It was such an inspiring talk that a few days later, the R Ladies Curitiba chapter was founded[5].

The number of participants, well above our expectations, and all the positive and enthusiastic feedbacks shows a clear demand for events gathering the community of R users in Brazil. The existing community is large, diverse and spread around the country. The objective is to promote regular R Day events for sharing expert knowledge and experience, and stimulate and strength relationships and networks. Figure 5 shows some of the participants who were there after the closing session.

Photos of the event are available at LEG's page on Facebook[6]. Slides, scripts and other

[1] http://rday.leg.ufpr.br
[2] http://www.rbras.org.br/rbras63/
[3] https://www.biometricsociety.org/
[4] https://www.youtube.com/watch?v=fnGvDEkjZy0
[5] https://www.meetup.com/pt-BR/rladies-curitiba
[6] https://www.facebook.com/leg.ufpr

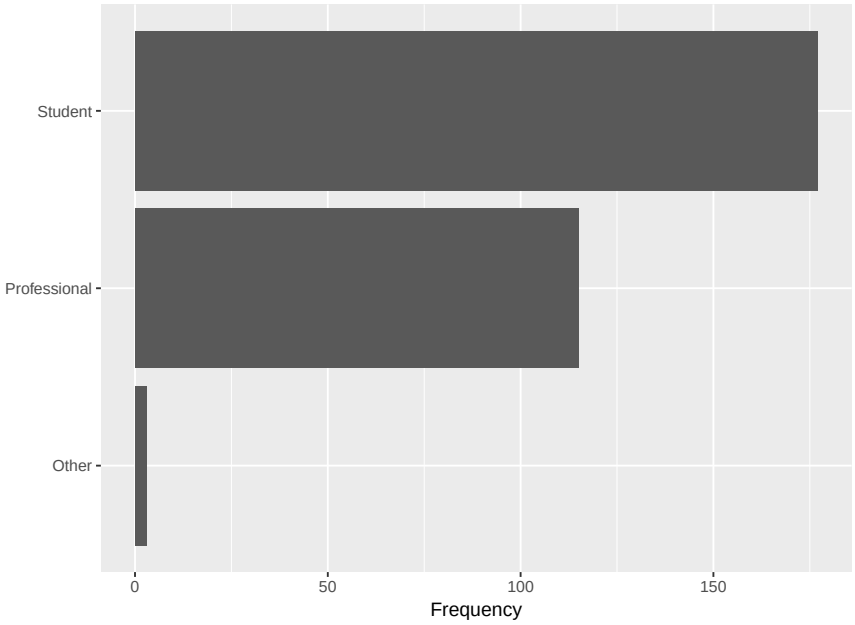


Figure 1: Number of subscribers by category.

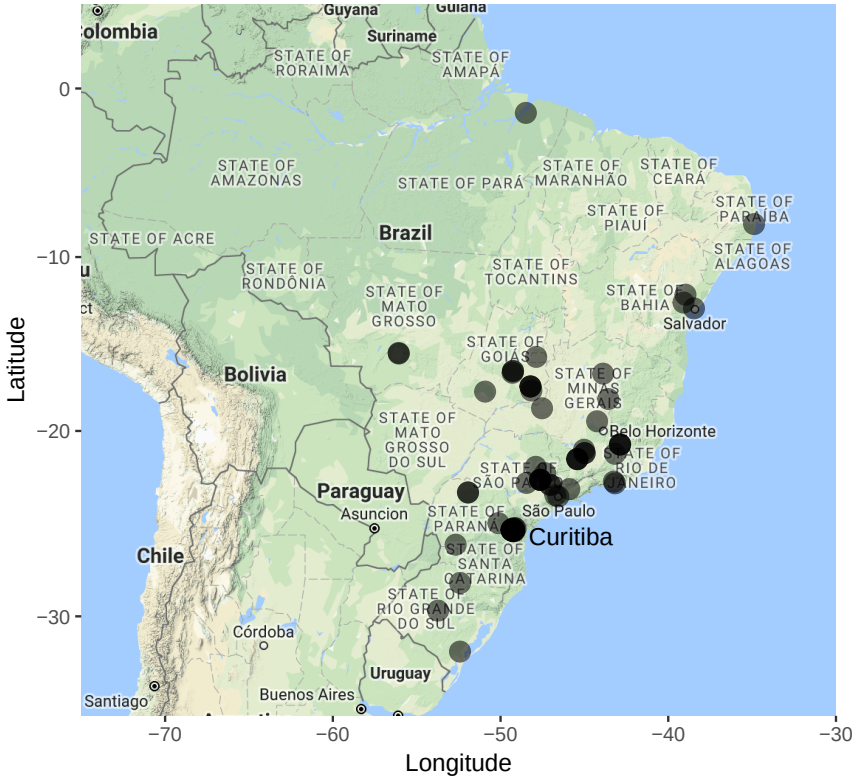


Figure 2: Home locations of the participants.

Figure 3: Prof. Paulo Justiniano (LEG) on the opening session.

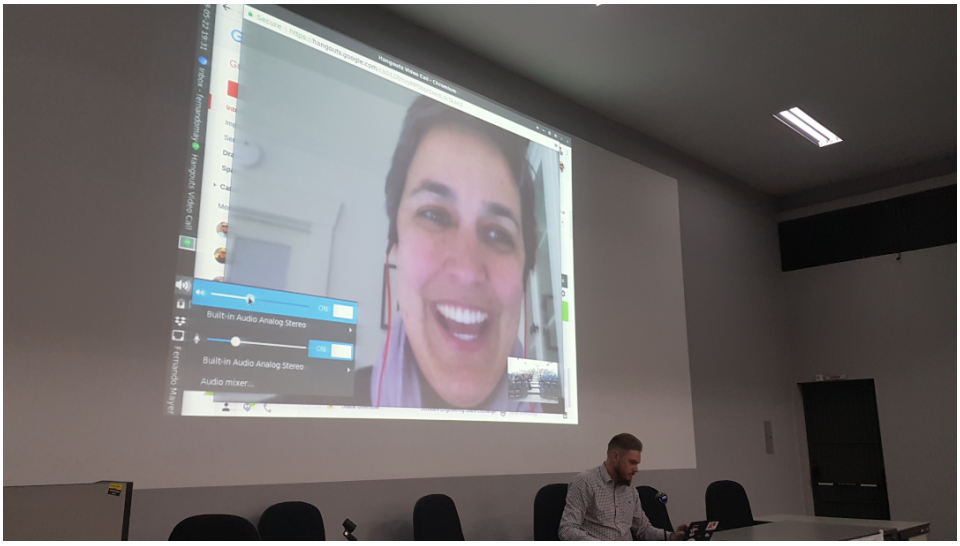


Figure 4: Gabriela de Queiroz (R Ladies) on the closing session.

Figure 5: Some participants of R Day after the closing session.

materials used by the presenters are available at R Day[7] web page.

Next steps

As R Day was so well received by the general community and by RBRAS meeting participants, the initial idea is to make R Day a regular satellite event. RBRAS meetings lasts for 3 days in even years, and for 5 days in odd years, and each year occurs in a different city of the country. That way, in even years, R Day could be organized in the city hosting the event. In odd years, we plan to make R Day in Curitiba, in a date that does not overlap with the RBRAS meeting, and maybe with 2 days duration.

Acknowledgements

We especially thank "The R Foundation" for endorsing R Day, particularly to Heather Turner, who contacted us and was always very kind and helpful to make this happen.

We thank all participants and the presenters, who offered to come voluntarily to make R Day a great event. Also, we would like to thank our sponsors[8], who provided the financial support for coffee breaks.

We also thank RBRAS for the partnership and support in making R Day a satellite event of its annual meeting.

Fernando P. Mayer
Laboratório de Estatística e Geoinformação - LEG/DEST/UFPR
fernando.mayer@ufpr.br

Walmes M. Zeviani
Laboratório de Estatística e Geoinformação - LEG/DEST/UFPR

Wagner H. Bonat
Laboratório de Estatística e Geoinformação - LEG/DEST/UFPR

Elias T. Krainski
Laboratório de Estatística e Geoinformação - LEG/DEST/UFPR

Paulo J. Ribeiro Jr
Laboratório de Estatística e Geoinformação - LEG/DEST/UFPR

[7] http://rday.leg.ufpr.br/materiais.html
[8] http://rday.leg.ufpr.br/apoio.html

R Foundation News

by Torsten Hothorn

Donations and members

Membership fees and donations received between 2017-12-01 and 2018-06-23.

Donations

Shaban Demirel (United States) Yves Deville (France) Danilo Godone (Italy) Reigo Hendrikson (Estonia) J. Brian Loria (United States) Joerg Maeder (Switzerland) Joni Oksanen (Finland) Ravinderpal Vaid (United States) Merck Research Laboratories, Kenilwort (United States) Zurich R Courses, Zurich (Switzerland)

Supporting benefactors

b-data GmbH, Zurich (Switzerland) Brigham Young University, Provo (United States) Kansai University, Fac of Commerce, Suita (Japan) Mirai Solutions GmbH, Zürich (Switzerland)

Supporting institutions

General Counsel Metrics, LLC, Princeton (United States)

Supporting members

Douglas Adamoski (Brazil) Ayala S. Allon (Israel) Justin Bem (Gabon) Florian Brezina (Germany) Jiddu Broersma (Netherlands) Michael Cantrall (United States) Robin Crockett (United Kingdom) Jorge de la Vega G (Mexico) Arturo Erdely (Mexico) Samuel Frame (Canada) Jan Marvin Garbuszus (Germany) J. Antonio García (Mexico) Susan Gruber (United States) Eugene Horber (Switzerland) Landon Jensen (United States) Stephen Kaluzny (United States) Sebastian Koehler (Germany) Luca La Rocca (Italy) Thomas Levine (United States) Fabio Marroni (Italy) Hans Mielke (Germany) Steffen Moritz (Germany) Alfredo Páez Jiménez (Spain) Elgin Perry (United States) Casper Petersen (Denmark) Laurent Schüpbach (Switzerland) András Tajti (Hungary) Marius Teodosiu (Romania) Pari Viswanathan (India) Chatchavan Wacharamanotham (Switzerland) Brandon Weinberg (United States) Douglas Zickuhr (Ireland) Joachim Zuckarelli (Germany)

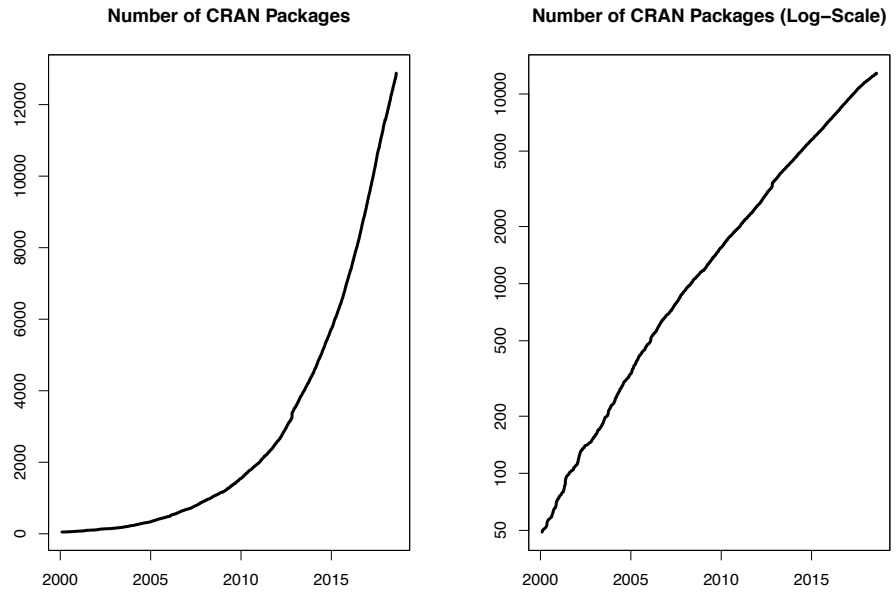
Torsten Hothorn
Universität Zürich, Switzerland Torsten.Hothorn@R-project.org

Changes on CRAN

2017-12-01 to 2018-06-30

by Kurt Hornik, Uwe Ligges and Achim Zeileis

In the past 7 months, 1178 new packages were added to the CRAN package repository. 18 packages were unarchived, 493 archived and none removed. The following shows the growth of the number of active packages in the CRAN package repository:



On 2018-06-30, the number of active packages was around 12582.

Changes in the CRAN checks

The package check pages now also show issues found by checks with alternative BLAS/LAPACK implementations (ATLAS, MKL and OpenBLAS), provided by Brian D. Ripley (see https://www.stats.ox.ac.uk/pub/bdr/Rblas/README.txt for more information).

Changes in the CRAN submission pipeline

Package maintainers who submitted packages this year found the automated submission system has been extended again. Incoming packages are automatically checked under both Linux and Windows. Based on these checks, the auto-check service (and in later steps possibly a CRAN team member) will take one of several actions:

archive reject the package, if the package does not pass the checks cleanly and the problem are not likely to be false positives.

inspect triggers a manual inspection of the package, this always happens for first time submissions and also for packages that show possible problems that are likely to be false positives. In both cases some human action is required.

pending if a decision is pending a response from the package maintainer: If an additional issue was present in the package that we cannot check for in the incoming checks (such

as the BLAS issues mentioned in the section above), the maintainer is automatically asked whether these issues have been fixed. Same is true for change of maintainer (or maintainer's mail address) where the old maintainer (old address) is automatically asked to confirm the maintainer change. The answers have to be processed manually.

pretest during a manual inspection, a human may trigger a new auto-check of the package for various reasons, e.g., after problems in the initial check or after updates of dependencies.

publish publish the package, if the package is already well established on CRAN, passes the checks cleanly, and does not have any reverse dependencies.

recheck if the package cleanly passes the checks and has at least one reverse dependency, this action moves the package into a queue for auto-checking the package's reverse dependencies. If the check status of at least one of the package's reverse dependencies changes to a worse state, the maintainer is asked whether this is expected and the other maintainers of affected packages have been informed and hence action *pending* is triggered. If no change to a worse state is discovered, the next action is *publish*.

All these actions include an informative e-mail message to the package maintainer. The package is also moved to a corresponding subdirectory of the incoming directory on CRAN. Once an action is *inspect* or *pending*, a CRAN team member will trigger the next action. The additional directory *pretest* is the one that contains the yet unprocessed packages.

During June 2018, CRAN received 2122 package submissions. For these, 3571 actions took place of which 2433 (68.1%) were auto processed actions and 1138 (31.9%) manual actions.

Minus some special cases, a summary of the auto-processed and manually triggered actions follows:

action	archive	inspect	pending	pretest	publish	recheck
auto	530	890	118	0	664	231
manual	412	36	81	50	449	110

These include the final decisions for the submissions which were as follows:

action	archive	publish
auto	470 (23.6%)	578 (29.0%)
manual	410 (20.6%)	535 (26.8%)

where we only count those as *auto* processed whose publication happened automatically in all steps.

The large number of 1467 manual action items (not counting additional mail communication) shows that even more automation is needed and will follow.

As the CRAN team is no longer able to respond to individual help requests or being involved in lengthy discussions for exceptions, please really use the corresponding mailing lists such as R-package-devel (see https://www.r-project.org/mail.html).

Changes in the CRAN Repository Policy

The Policy now says the following:

- CRAN hosts packages in publication quality and is not a development platform. A package's contribution has to be non-trivial.
- Packages should not write in the user's home filespace (including clipboards), nor anywhere else on the file system apart from the R session's temporary directory (or during installation in the location pointed to by `TMPDIR`: and such usage should be cleaned up). Installing into the system's R installation (e.g., scripts to its 'bin' directory) is not allowed.

- Packages should not attempt to disable compiler diagnostics.
- Uploads must be source tarballs created by `R CMD build` and following the '*PACKAGE_VERSION*.tar.gz' naming scheme. *This should be done with current R-patched or the current release of R.*
- For packages which have recently been archived, a snapshot of the CRAN results page at the time of archival may be available under https://cran-archive.r-project.org/web/checks/. (Note that only a few of the links from the snapshot will work: normally those to listed 'Alternative issues' will.)

CRAN package repository archive

As of 2018-07, a full CRAN mirror takes about 176 G, which is quite a lot, in particular taking into account that a considerable part is not needed for current versions of R and contributed packages. CRAN mirrors already only provide Mac and Windows binaries for R versions not older than 5 years (currently, from R 3.0 onwards), with disk usages of 65 G (about 39%) and 46 G (about 26%), respectively, for Mac and Windows. Older versions are available from the CRAN archive service available at https://CRAN-archive.R-project.org (but not the CRAN mirrors). For the CRAN package repository, sources currently use 52 G, with 44 G used for archived (non-current) versions. Given that the source archive area takes up about 25% of the whole CRAN mirror area, with material most likely needed only very occasionally, we will thus move the source archive area to the CRAN archive server during 2018-08. Archived source packages can then be obtained directly via https://CRAN-archive.R-project.org/src/contrib/Archive, and of course via the "Old sources" download links on the package web pages on every CRAN mirror.

CRAN mirror security

Currently, there are 100 official CRAN mirrors, 66 of which provide both secure downloads via '`https`' *and* use secure mirroring from the CRAN master (via rsync through ssh tunnels). Since the R 3.4.0 release, `chooseCRANmirror()` offers these mirrors in preference to the others which are not fully secured (yet).

New packages in CRAN task views

Bayesian **openEBGM**, **tRophicPosition**.

ClinicalTrials **InformativeCensoring**, **Mediana**, **ThreeArmedTrials**, **clusterPower**, **crmPack**, **dfped**, **dfpk**, **ewoc**, **gsbDesign**.

DifferentialEquations **QPot**, **cOde**, **dMod**, **phaseR**, **rODE**, **rodeo**, **rpgm**.

Distributions **MittagLeffleR**, **coga**, **hyper2**.

Econometrics **OrthoPanels**, **dlsem**, **pder**, **wooldridge**, **zTree**.

ExperimentalDesign **DoE.MIParray**, **FMC**, **MBHdesign**, **PBIBD**, **bioOED**, **edesign**, **idefix**, **minimalRSD**, **odr**, **optbdmaeAT**, **optrcdmaeAT**, **rsurface**, **sFFLHD**, **skpr***, **soptdmaeA**, **unrepx**.

ExtremeValue **POT**.

FunctionalData **covsep**, **denseFLMM**, **freqdom.fda**, **ftsspec**.

HighPerformanceComputing **Sim.DiffProc**, **drake**, **parSim**.

MachineLearning **ICEbox**, **effects**, **ggRandomForests**, **pdp**, **plotmo**, **tensorflow**.

MetaAnalysis **CIAAWconsensus**, **ConfoundedMeta**, **MetaSubtract**, **RandMeta**, **TFisher**, **clubSandwich**, **effsize**, **forestmodel**, **getmstatistic**, **metaBMA**, **metacart**, **metaforest**, **nmaINLA**, **psychmeta**, **ratesci**, **rma.exact**.

NaturalLanguageProcessing **alineR**, **ore**, **rel**, **stm**, **stringdist**.

NumericalMathematics **PythonInR**, **SnakeCharmR**, **XR**, **XRJulia**, **XRPython**, **expint**, **feather**, **findpython**, **fourierin**, **interp**, **logOfGamma**, **reticulate**, **tripack**.

Optimization **ABCoptim**, **CVXR**, **ManifoldOptim**, **Rtnmin**, **SACOBRA**, **colf**, **coneproj**, **ecr**, **flacco**, **metaheuristicOpt**, **mize**, **n1qn1**, **ompr**, **optimr**, **optimsimplex**, **quadprogXT**, **sdpt3r**.

Pharmacokinetics **RxODE**.

Phylogenetics **treeplyr**.

Psychometrics **CTTShiny**, **EFAutilities**, **MIIVsem**, **PLmixed**, **dexter**, **umx**.

Spatial **spm**, **spsann**.

SpatioTemporal **FLightR**, **sf**, **sigloc**.

TimeSeries **dLagM**, **fpp2**, **freqdom**, **freqdom.fda**, **ftsa**, **funtimes**, **influxdbr**, **odpc**, **sweep**, **timetk**, **tscount**, **wktmo**.

WebTechnologies **gtrendsR**.

Kurt Hornik
WU Wirtschaftsuniversität Wien, Austria
Kurt.Hornik@R-project.org

Uwe Ligges
TU Dortmund, Germany
Uwe.Ligges@R-project.org

Achim Zeileis
Universität Innsbruck, Austria
Achim.Zeileis@R-project.org

News from the Bioconductor Project

by Bioconductor Core Team

The *Bioconductor* project provides tools for the analysis and comprehension of high-throughput genomic data. *Bioconductor* 3.7 was released on 1 May, 2018. It is compatible with R 3.5.1 and consists of 1560 software packages, 342 experiment data packages, and 919 up-to-date annotation packages. The release announcement includes descriptions of 98 new software packages and updated NEWS files for many additional packages. Start using *Bioconductor* by installing the most recent version of R and evaluating the commands

```
source("https://bioconductor.org/biocLite.R")
biocLite()
```

Install additional packages and dependencies, e.g., **SingleCellExperiment**, with

```
BiocInstaller::biocLite("SingleCellExperiment")
```

Docker and Amazon images provide an effective on-ramp for power users to rapidly obtain access to standardized and scalable computing environments. Key resources include:

- The bioconductor.org web site to install, learn, use, and develop *Bioconductor* packages.
- A listing of available software, linking to pages describing each package.
- A question-and-answer style user support site and developer-oriented mailing list.
- The F1000Research *Bioconductor* channel for peer-reviewed *Bioconductor* work flows.
- Our package submission repository for open technical review of new packages.

Our annual conference will be on July 25 ('Developer Day'), 26, and 27, 2018, in Toronto, Canada, and features an exceptional line-up of morning scientific talks and afternoon hands-on workshops.

Bioconductor *Core Team*
Biostatistics and Bioinformatics
Roswell Park Comprehensive Cancer Center, Buffalo, NY
USA maintainer@bioconductor.org

Changes in R

From version 3.5.0 to version 3.5.0 patched

by R Core Team

CHANGES IN R 3.5.0 patched

BUG FIXES

- `file("stdin")` is no longer considered seekable.
- `dput()` and `dump()` are no longer truncating when `options(deparse.max.lines = *)` is set.
- Calls with an S3 class are no longer evaluated when printed, fixing part of PR#17398, thanks to a patch from Lionel Henry.
- Allow `file` argument of `Rscript` to include space even when it is first on the command line.
- `callNextMethod()` uses the generic from the environment of the calling method. Reported by Hervé Pagès with well documented examples.
- Compressed file connections are marked as blocking.
- `optim(*,lower = c(-Inf,-Inf))` no longer warns (and switches the method), thanks to a suggestion by John Nash.
- `predict(fm,newdata)` is now correct also for models where the formula has terms such as `splines::ns(..)` or `stats::poly(..)`, fixing PR#17414, based on a patch from Duncan Murdoch.
- `simulate.lm(glm(*,gaussian(link = <non-default>)))` has been corrected, fixing PR#17415 thanks to Alex Courtiol.
- `unlist(x)` no longer fails in some cases of nested empty lists. Reported by Steven Nydick.
- `qr.coef(qr(<all 0,w/ colnames>))` now works. Reported by Kun Ren.
- The radix sort is robust to vectors with >1 billion elements (but long vectors are still unsupported). Thanks to Matt Dowle for the fix.
- Terminal connections (e.g., stdin) are no longer buffered. Fixes PR#17432.
- `deparse(x)`, `dput(x)` and `dump()` now respect `c()`'s argument names `recursive` and `use.names`, e.g., for `x <-setNames(0,"recursive")`, thanks to Suharto Anggono's PR#17427.
- Unbuffered connections now work with encoding conversion. Reported by Stephen Berman.
- '.Renviron' on Windows with `Rgui` is again by default searched for in user documents directory when invoked *via* the launcher icon. Reported by Jeroen Ooms.
- `printCoefmat()` now also works with explicit `right=TRUE`.
- `print.noquote()` now also works with explicit `quote=FALSE`.
- The default method for `pairs(..,horInd=*,verInd=*)` now gets the correct order, thanks to reports by Chris Andrews and Gerrit Eichner. Additionally, when `horInd` or `verInd` contain only a subset of variables, all the axes are labeled correctly now.

- `agrep("..|..",...,fixed=FALSE)` now matches when it should, thanks to a reminder by Andreas Kolter.
- `str(ch)` now works for more invalid multibyte strings.

CHANGES IN R 3.5.0

SIGNIFICANT USER-VISIBLE CHANGES

- All packages are by default byte-compiled on installation. This makes the installed packages larger (usually marginally so) and may affect the format of messages and tracebacks (which often exclude `.Call` and similar).

NEW FEATURES

- `factor()` now uses `order()` to sort its levels, rather than `sort.list()`. This allows `factor()` to support custom vector-like objects if methods for the appropriate generics are defined. It has the side effect of making `factor()` succeed on empty or length-one non-atomic vector(-like) types (e.g., `"list"`), where it failed before.
- `diag()` gets an optional `names` argument: this may require updates to packages defining S4 methods for it.
- `chooseCRANmirror()` and `chooseBioCmirror()` no longer have a `useHTTPS` argument, not needed now all R builds support '`https://`' downloads.
- New `summary()` method for `warnings()` with a (somewhat experimental) `print()` method.
- (**methods** package.) `.self` is now automatically registered as a global variable when registering a reference class method.
- `tempdir(check = TRUE)` recreates the `tempdir()` directory if it is no longer valid (e.g. because some other process has cleaned up the '/tmp' directory).
- New `askYesNo()` function and `"askYesNo"` option to ask the user binary response questions in a customizable but consistent way. (Suggestion of PR#17242.)
- New low level utilities `...elt(n)` and `...length()` for working with `...` parts inside a function.
- `isTRUE()` is more tolerant and now true in

```
x <- rlnorm(99)
isTRUE(median(x) == quantile(x)["50%"])
```

  New function `isFALSE()` defined analogously to `isTRUE()`.
- The default symbol table size has been increased from 4119 to 49157; this may improve the performance of symbol resolution when many packages are loaded. (Suggested by Jim Hester.)
- `line()` gets a new option `iter = 1`.
- Reading from connections in text mode is buffered, significantly improving the performance of `readLines()`, as well as `scan()` and `read.table()`, at least when specifying `colClasses`.
- `order()` is smarter about picking a default sort `method` when its arguments are objects.

- `available.packages()` has two new arguments which control if the values from the per-session repository cache are used (default true, as before) and if so how old cached values can be to be used (default one hour).

  These arguments can be passed from `install.packages()`, `update.packages()` and functions calling that: to enable this `available.packages()`, `packageStatus()` and `download.file()` gain a `...` argument.

- `packageStatus()`'s `upgrade()` method no longer ignores its `...` argument but passes it to `install.packages()`.

- `installed.packages()` gains a `...` argument to allow arguments (including `noCache`) to be passed from `new.packages()`, `old.packages()`, `update.packages()` and `packageStatus()`.

- `factor(x,levels,labels)` now allows duplicated `labels` (not duplicated `levels`!). Hence you can map different values of `x` to the same level directly.

- Attempting to use `names<-()` on an S4 derivative of a basic type no longer emits a warning.

- The `list` method of `within()` gains an option `keepAttrs = FALSE` for some speed-up.

- `system()` and `system2()` now allow the specification of a maximum elapsed time ('timeout').

- `debug()` supports debugging of methods on any object of S4 class `"genericFunction"`, including group generics.

- Attempting to increase the length of a variable containing `NULL` using `length()<-` still has no effect on the target variable, but now triggers a warning.

- `type.convert()` becomes a generic function, with additional methods that operate recursively over list and `data.frame` objects. Courtesy of Arni Magnusson (PR#17269).

- `lower.tri(x)` and `upper.tri(x)` only needing `dim(x)` now work via new functions `.row()` and `.col()`, so no longer call `as.matrix()` by default in order to work efficiently for all kind of matrix-like objects.

- `print()` methods for `"xgettext"` and `"xngettext"` now use `encodeString()` which keeps, e.g. `"\n"`, visible. (Wish of PR#17298.)

- `package.skeleton()` gains an optional `encoding` argument.

- `approx()`, `spline()`, `splinefun()` and `approxfun()` also work for long vectors.

- `deparse()` and `dump()` are more useful for S4 objects, `dput()` now using the same internal C code instead of its previous imperfect workaround R code. S4 objects now typically deparse perfectly, i.e., can be recreated identically from deparsed code.

  `dput()`, `deparse()` and `dump()` now print the `names()` information only once, using the more readable `(tag = value)` syntax, notably for `list()`s, i.e., including data frames.

  These functions gain a new control option `"niceNames"` (see `.deparseOpts()`), which when set (as by default) also uses the `(tag = value)` syntax for atomic vectors. On the other hand, without deparse options `"showAttributes"` and `"niceNames"`, names are no longer shown also for lists. `as.character(list( c (one = 1)))` now includes the name, as `as.character(list(list(one = 1)))` has always done.

  `m:n` now also deparses nicely when $m > n$.

  The `"quoteExpressions"` option, also part of `"all"`, no longer `quote()`s formulas as that may not re-parse identically. (PR#17378)

- If the option `setWidthOnResize` is set and `TRUE`, R run in a terminal using a recent `readline` library will set the `width` option when the terminal is resized. Suggested by Ralf Goertz.
- If multiple `on.exit()` expressions are set using `add = TRUE` then all expressions will now be run even if one signals an error.
- `mclapply()` gets an option `affinity.list` which allows more efficient execution with heterogeneous processors, thanks to Helena Kotthaus.
- The `character` methods for `as.Date()` and `as.POSIXlt()` are more flexible *via* new arguments `tryFormats` and `optional`: see their help pages.
- `on.exit()` gains an optional argument `after` with default `TRUE`. Using `after = FALSE` with `add = TRUE` adds an exit expression before any existing ones. This way the expressions are run in a first-in last-out fashion. (From Lionel Henry.)
- On Windows, `file.rename()` internally retries the operation in case of error to attempt to recover from possible anti-virus interference.
- Command line completion on '`::`' now also includes lazy-loaded data.
- If the `TZ` environment variable is set when date-time functions are first used, it is recorded as the session default and so will be used rather than the default deduced from the OS if `TZ` is subsequently unset.
- There is now a `[` method for class `"DLLInfoList"`.
- `glm()` and `glm.fit` get the same `singular.ok = TRUE` argument that `lm()` has had forever. As a consequence, in `glm(*,method = <your_own>)`, user specified methods need to accept a `singular.ok` argument as well.
- `aspell()` gains a filter for Markdown ('.md' and '.Rmd') files.
- `intToUtf8(multiple = FALSE)` gains an argument to allow surrogate pairs to be interpreted.
- The maximum number of DLLs that can be loaded into R e.g. *via* `dyn.load()` has been increased up to 614 when the OS limit on the number of open files allows.
- `Sys.timezone()` on a Unix-alike caches the value at first use in a session: *inter alia* this means that setting `TZ` later in the session affects only the *current* time zone and not the *system* one.

  `Sys.timezone()` is now used to find the system timezone to pass to the code used when R is configured with '`--with-internal-tzcode`'.
- When `tar()` is used with an external command which is detected to be GNU `tar` or libarchive `tar` (aka `bsdtar`), a different command-line is generated to circumvent line-length limits in the shell.
- `system(*,intern = FALSE)`, `system2()` (when not capturing output), `file.edit()` and `file.show()` now issue a warning when the external command cannot be executed.
- The "default" (`"lm"` etc) methods of `vcov()` have gained new optional argument `complete = TRUE` which makes the `vcov()` methods more consistent with the `coef()` methods in the case of singular designs. The former (back-compatible) behavior is given by `vcov(*,complete = FALSE)`.
- `coef()` methods (for `lm` etc) also gain a `complete = TRUE` optional argument for consistency with `vcov()`.
  For `"aov"`, both `coef()` and `vcov()` methods remain back-compatibly consistent, using the *other* default, `complete = FALSE`.

- `attach(*,pos = 1)` is now an error instead of a warning.
- New function `getDefaultCluster()` in package **parallel** to get the default cluster set via `setDefaultCluster()`.
- `str(x)` for atomic objects `x` now treats both cases of `is.vector(x)` similarly, and hence much less often prints `"atomic"`. This is a slight non-back-compatible change producing typically both more informative and shorter output.
- `write.dcf()` gets optional argument `useBytes`.
- New, partly experimental `packageDate()` which tries to get a valid `"Date"` object from a package 'DESCRIPTION' file, thanks to suggestions in PR#17324.
- `tools::resaveRdaFiles()` gains a `version` argument, for use when packages should remain compatible with earlier versions of R.
- `ar.yw(x)` and hence by default `ar(x)` now work when `x` has `NA`s, mostly thanks to a patch by Pavel Krivitsky in PR#17366. The `ar.yw.default()`'s AIC computations have become more efficient by using `determinant()`.
- New `warnErrList()` utility (from package **nlme**, improved).
- By default the (arbitrary) signs of the loadings from `princomp()` are chosen so the first element is non-negative.
- If '`--default-packages`' is not used, then `Rscript` now checks the environment variable `R_SCRIPT_DEFAULT_PACKAGES`. If this is set, then it takes precedence over `R_DEFAULT_PACKAGES`. If default packages are not specified on the command line or by one of these environment variables, then `Rscript` now uses the same default packages as `R`. For now, the previous behavior of not including **methods** can be restored by setting the environment variable `R_SCRIPT_LEGACY` to '`yes`'.
- When a package is found more than once, the warning from `find.package(*,verbose=TRUE)` lists all library locations.
- POSIXt objects can now also be rounded or truncated to month or year.
- `stopifnot()` can be used alternatively via new argument `exprs` which is nicer and useful when testing several expressions in one call.
- The environment variable `R_MAX_VSIZE` can now be used to specify the maximal vector heap size. On macOS, unless specified by this environment variable, the maximal vector heap size is set to the maximum of 16GB and the available physical memory. This is to avoid having the `R` process killed when macOS over-commits memory.
- `sum(x)` and `sum(x1,x2,..,x<N>)` with many or long logical or integer vectors no longer overflows (and returns `NA` with a warning), but returns `double` numbers in such cases.
- Single components of `"POSIXlt"` objects can now be extracted and replaced via `[` indexing with 2 indices.
- S3 method lookup now searches the namespace registry after the top level environment of the calling environment.
- Arithmetic sequences created by `1:n`, `seq_along`, and the like now use compact internal representations via the `ALTREP` framework. Coercing integer and numeric vectors to character also now uses the `ALTREP` framework to defer the actual conversion until first use.
- Finalizers are now run with interrupts suspended.

- `merge()` gains new option `no.dups` and by default suffixes the second of two duplicated column names, thanks to a proposal by Scott Ritchie (and Gabe Becker).
- `scale.default(x,center,scale)` now also allows `center` or `scale` to be "numeric-alike", i.e., such that `as.numeric(.)` coerces them correctly. This also eliminates a wrong error message in such cases.
- `par*apply` and `par*applyLB` gain an optional argument `chunk.size` which allows to specify the granularity of scheduling.
- Some `as.data.frame()` methods, notably the `matrix` one, are now more careful in not accepting duplicated or `NA` row names, and by default produce unique non-NA row names. This is based on new function `.rowNamesDF(x,make.names = *) <-rNms` where the logical argument `make.names` allows to specify *how* invalid row names `rNms` are handled. `.rowNamesDF()` is a "workaround" compatible default.
- R has new serialization format (version 3) which supports custom serialization of `ALTREP` framework objects. These objects can still be serialized in format 2, but less efficiently. Serialization format 3 also records the current native encoding of unflagged strings and converts them when de-serialized in R running under different native encoding. Format 3 comes with new serialization magic numbers (RDA3, RDB3, RDX3). Format 3 can be selected by `version = 3` in `save()`, `serialize()` and `saveRDS()`, but format 2 remains the default for all serialization and saving of the workspace. Serialized data in format 3 cannot be read by versions of R prior to version 3.5.0.
- The `"Date"` and "date-time" classes `"POSIXlt"` and `"POSIXct"` now have a working `length<-()` method, as wished in PR#17387.
- `optim(*,control = list(warn.1d.NelderMead = FALSE))` allows to turn off the warning when applying the default `"Nelder-Mead"` method to 1-dimensional problems.
- `matplot(..,panel.first = .)` etc now work, as `log` becomes explicit argument and `...` is passed to `plot()` unevaluated, as suggested by Sebastian Meyer in PR#17386.
- Interrupts can be suspended while evaluating an expression using `suspendInterrupts`. Subexpression can be evaluated with interrupts enabled using `allowInterrupts`. These functions can be used to make sure cleanup handlers cannot be interrupted.
- R 3.5.0 includes a framework that allows packages to provide alternate representations of basic R objects (`ALTREP`). The framework is still experimental and may undergo changes in future R releases as more experience is gained. For now, documentation is provided in `https://svn.r-project.org/R/branches/ALTREP/ALTREP.html`.

UTILITIES

- `install.packages()` for source packages now has the possibility to set a 'timeout' (elapsed-time limit). For serial installs this uses the `timeout` argument of `system2()`: for parallel installs it requires the `timeout` utility command from GNU **coreutils**.
- It is now possible to set 'timeouts' (elapsed-time limits) for most parts of `R CMD check` *via* environment variables documented in the 'R Internals' manual.
- The 'BioC extra' repository which was dropped from Bioconductor 3.6 and later has been removed from `setRepositories()`. This changes the mapping for 6–8 used by `setRepositories(ind=)`.
- `R CMD check` now also applies the settings of environment variables `_R_CHECK_SUGGESTS_ONLY_` and `_R_CHECK_DEPENDS_ONLY_` to the re-building of vignettes.
- `R CMD check` with environment variable `_R_CHECK_DEPENDS_ONLY_` set to a true value makes test-suite-management packages available and (for the time being) works around a common omission of **rmarkdown** from the '`VignetteBuilder`' field.

INSTALLATION on a UNIX-ALIKE

- Support for a system Java on macOS has been removed — install a fairly recent Oracle Java (see 'R Installation and Administration' §C.3.2).
- `configure` works harder to set additional flags in `'SAFE_FFLAGS'` only where necessary, and to use flags which have little or no effect on performance.

  In rare circumstances it may be necessary to override the setting of `'SAFE_FFLAGS'`.
- C99 functions `expm1`, `hypot`, `log1p` and `nearbyint` are now required.
- `configure` sets a `'-std'` flag for the C++ compiler for all supported C++ standards (e.g., `'-std=gnu++11'` for the C++11 compiler). Previously this was not done in a few cases where the default standard passed the tests made (e.g. `clang 6.0.0` for C++11).

C-LEVEL FACILITIES

- 'Writing R Extensions' documents macros `MAYBE_REFERENCED`, `MAYBE_SHARED` and `MARK_NOT_MUTABLE` that should be used by package `C` code instead `NAMED` or `SET_NAMED`.
- The object header layout has been changed to support merging the `ALTREP` branch. This requires re-installing packages that use compiled code.
- 'Writing R Extensions' now documents the `R_tryCatch`, `R_tryCatchError`, and `R_UnwindProtect` functions.
- `NAMEDMAX` has been raised to 3 to allow protection of intermediate results from (usually ill-advised) assignments in arguments to `BUILTIN` functions. Package `C` code using `SET_NAMED` may need to be revised.

DEPRECATED AND DEFUNCT

- `Sys.timezone(location = FALSE)` is defunct, and is ignored (with a warning).
- `methods:::bind_activation()` is defunct now; it typically has been unneeded for years.

  The undocumented 'hidden' objects `.__H__.cbind` and `.__H__.rbind` in package **base** are deprecated (in favour of `cbind` and `rbind`).
- The declaration of `pythag()` in 'Rmath.h' has been removed — the entry point has not been provided since R 2.14.0.

BUG FIXES

- `printCoefmat()` now also works without column names.
- The S4 methods on `Ops()` for the `"structure"` class no longer cause infinite recursion when the structure is not an S4 object.
- `nlm(f,..)` for the case where `f()` has a `"hessian"` attribute now computes $LL' = H + \mu I$ correctly. (PR#17249).
- An S4 method that "rematches" to its generic and overrides the default value of a generic formal argument to `NULL` no longer drops the argument from its formals.
- `Rscript` can now accept more than one argument given on the `'#!'` line of a script. Previously, one could only pass a single argument on the `'#!'` line in Linux.
- Connections are now written correctly with encoding `"UTF-16LE"`. (PR#16737).

- Evaluation of `..0` now signals an error. When `..1` is used and `...` is empty, the error message is more appropriate.
- (Windows mainly.) Unicode code points which require surrogate pairs in UTF-16 are now handled. All systems should properly handle surrogate pairs, even those systems that do not need to make use of them. (PR#16098)
- `stopifnot(e,e2,...)` now evaluates the expressions sequentially and in case of an error or warning shows the relevant expression instead of the full `stopifnot(..)` call.
- `path.expand()` on Windows now accepts paths specified as UTF-8-encoded character strings even if not representable in the current locale. (PR#17120)
- `line(x,y)` now correctly computes the medians of the left and right group's x-values and in all cases reproduces straight lines.
- Extending S4 classes with slots corresponding to special attributes like `dim` and `dimnames` now works.
- Fix for `legend()` when `fill` has multiple values the first of which is `NA` (all colours used to default to `par(fg)`). (PR#17288)
- `installed.packages()` did not remove the cached value for a library tree that had been emptied (but would not use the old value, just waste time checking it).
- The documentation for `installed.packages(noCache = TRUE)` incorrectly claimed it would refresh the cache.
- `aggregate(<data.frame>)` no longer uses spurious names in some cases. (PR#17283)
- `object.size()` now also works for long vectors.
- `packageDescription()` tries harder to solve re-encoding issues, notably seen in some Windows locales. This fixes the `citation()` issue in PR#17291.
- `poly(<matrix>,3)` now works, thanks to prompting by Marc Schwartz.
- `readLines()` no longer segfaults on very large files with embedded `'\0'` (aka 'nul') characters. (PR#17311)
- `ns()` (package **splines**) now also works for a single observation. `interpSpline()` gives a more friendly error message when the number of points is less than four.
- `dist(x,method = "canberra")` now uses the correct definition; the result may only differ when `x` contains values of differing signs, e.g. not for 0-1 data.
- `methods:::cbind()` and `methods:::rbind()` avoid deep recursion, thanks to Suharto Anggono via PR#17300.
- Arithmetic with zero-column data frames now works more consistently; issue raised by Bill Dunlap.

  Arithmetic with data frames gives a data frame for `^` (which previously gave a numeric matrix).
- `pretty(x,n)` for large `n` or large `diff(range(x))` now works better (though it was never meant for large `n`); internally it uses the same rounding fuzz (1e-10) as `seq.default()` — as it did up to 2010-02-03 when both were 1e-7.
- Internal C-level `R_check_class_and_super()` and hence `R_check_class_etc()` now also consider non-direct super classes and hence return a match in more cases. This e.g., fixes behaviour of derived classes in package **Matrix**.
- Reverted unintended change in behavior of `return` calls in `on.exit` expressions introduced by stack unwinding changes in R 3.3.0.

- Attributes on symbols are now detected and prevented; attempt to add an attribute to a symbol results in an error.
- `fisher.test(*,workspace = <n>)` now may also increase the internal stack size which allows larger problem to be solved, fixing PR#1662.
- The **methods** package no longer directly copies slots (attributes) into a prototype that is of an "abnormal" (reference) type, like a symbol.
- The **methods** package no longer attempts to call `length<-()` on `NULL` (during the bootstrap process).
- The **methods** package correctly shows methods when there are multiple methods with the same signature for the same generic (still not fully supported, but at least the user can see them).
- `sys.on.exit()` is now always evaluated in the right frame. (From Lionel Henry.)
- `seq.POSIXt(*,by = "<n>DSTdays")` now should work correctly in all cases and is faster. (PR#17342)
- `.C()` when returning a logical vector now always maps values other than FALSE and NA to TRUE (as documented).
- Subassignment with zero length vectors now coerces as documented (PR#17344). Further, `x <-numeric(); x[1] <-character()` now signals an error 'replacement has length zero' (or a translation of that) instead of doing nothing.
- (Package **parallel**.) `mclapply()`, `pvec()` and `mcparallel()` (when `mccollect()` is used to collect results) no longer leave zombie processes behind.
- `R CMD INSTALL <pkg>` now produces the intended error message when, e.g., the `LazyData` field is invalid.
- `as.matrix(dd)` now works when the data frame `dd` contains a column which is a data frame or matrix, including a 0-column matrix/d.f. .
- `mclapply(X,mc.cores)` now follows its documentation and calls `lapply()` in case `mc.cores = 1` also in the case `mc.preschedule` is false. (PR#17373)
- `aggregate(<data.frame>,drop=FALSE)` no longer calls the function on <empty> parts but sets corresponding results to `NA`. (Thanks to Suharto Anggono's patches in PR#17280).
- The `duplicated()` method for data frames is now based on the `list` method (instead of string coercion). Consequently `unique()` is better distinguishing data frame rows, fixing PR#17369 and PR#17381. The methods for matrices and arrays are changed accordingly.
- Calling `names()` on an S4 object derived from `"environment"` behaves (by default) like calling `names()` on an ordinary environment.
- `read.table()` with a non-default separator now supports quotes following a non-whitespace character, matching the behavior of `scan()`.
- `parLapplyLB` and `parSapplyLB` have been fixed to do load balancing (dynamic scheduling). This also means that results of computations depending on random number generators will now really be non-reproducible, as documented.
- Indexing a list using dollar and empty string (`l$""`) returns NULL.
- Using `\usage{ data(<name>,package="<pkg>") }` no longer produces `R CMD check` warnings.

- `match.arg()` more carefully chooses the environment for constructing default `choices`, fixing PR#17401 as proposed by Duncan Murdoch.
- Deparsing of consecutive `!` calls is now consistent with deparsing unary `-` and `+` calls and creates code that can be reparsed exactly; thanks to a patch by Lionel Henry in PR#17397. (As a side effect, this uses fewer parentheses in some other deparsing involving `!` calls.)