crossdes

A Package for Design and Randomization in Crossover Studies

by Oliver Sailer

Introduction

Design of experiments for crossover studies requires dealing with possible order and carryover effects that may bias the results of the analysis. One approach to deal with those effects is to use designs balanced for the respective effects. Almost always there are effects unaccounted for in the statistical model that may or may not be of practical importance. An important way of addressing those effects is randomization.

crossdes constructs and randomizes balanced block designs for multiple treatments that are also balanced for carryover effects. Jones and Kenward (1989), Ch. 5 review different crossover designs and cite optimality results. Wakeling and MacFie (1995) promoted the use of balanced designs for sensory studies. Five construction methods described in this paper are implemented here. They include Williams designs (Williams, 1949) and designs based on complete sets of mutually orthogonal latin squares. If one is just interested in getting balanced incomplete block designs that are not balanced for carryover, the package **AlgDesign** may be used (Wheeler, 2004). **crossdes** also contains functions to check given designs for balance.

crossdes is available on CRAN. It requires three packages also available on CRAN: **AlgDesign**, **gtools**, located in package bundle **gregmisc**, and **MASS**, available in bundle **VR**.

Simple Crossover Studies for Multiple Treatments

When comparing the effects of different treatments on experimental units, economical and ethical considerations often limit the number of units to be included in the study. In crossover studies, this restraint is addressed by administering multiple treatments to each subject.

However, the design setup leads to a number of possible problems in such an analysis. In addition to the treatment effects, we have to consider subject effects, order effects and carryover effects. In case we are explicitly interested in estimating those effects, we may fit corresponding mixed effects models. If, however, we are only interested in differences between treatments, we may use balanced designs that average out the nuisance parameters as far as possible.

By far the most common crossover design is the AB/BA design with just two treatments applied in two periods, see e.g. Jones and Kenward (1989) or Senn (1993). Here we restrict our attention to balanced designs for the comparison of three or more treatments and to cases, where each unit receives each treatment at most once. In general these designs are no longer balanced for treatment or carryover effects if some observations are missing. Simulation studies suggest that these designs are fairly robust against small numbers of dropouts, see e.g. Kunert and Sailer (2005). In the next section we explain how to actually get balanced designs in R using **crossdes**.

Design Construction

There are three important parameters that we need to define before we can construct a design: The number of treatments t to compare, the number of periods $p \leq t$, i.e. the number of treatments each subject gets and the (maximum) number of subjects or experimental units n available for the study.

To ways to get balanced designs are implemented. The first approach is to specify one of the five construction methods described below. Unless there is no design for the specified parameters t, p and n, a matrix representing the experimental plan is given. Rows represent subjects and columns represent periods. The design should then be randomized. The function `random.RT` randomizes row and treatment labels.

The function `all.combin` constructs balanced block designs that are balanced for all orders of carryover effects up to $p-1$ based on all possible permutations of treatment orders. The user specifies the number of treatments and periods, t and p. While this approach works for any $t \geq 2$ and $p \leq t$, the fact that every treatment combination occurs leads to very large numbers of subjects required for the study as long as t and p aren't very small, see e.g. Patterson (1952).

As long as t is a prime power, there is a possibility to drastically reduce the number of subjects required and still retain the same properties as described above. The function `des.MOLS` constructs designs based on complete sets of mutually orthogonal latin squares (MOLS) where $t \leq 100$ has to be a prime power and $p \leq t$, see e.g. Williams (1949). The function `MOLS` gives a complete set of $t-1$ MOLS based on Galois Fields that is of use in other applications of combinatorics in the design of experiments as well (Street and Street, 1987). The necessary arguments are r and s, where r is prime and s is a positive integer such that $r^s = t$.

If the number of subjects is still too large or t is not a prime power, one may consider designs that are only balanced for first order carryover effects. If each subject gets each treatment once, Williams designs only use t or $2t$ subjects (Williams, 1949). `williams` gives carryover balanced latin squares for $t \geq 2$.

If the number of treatments is large, it may not be possible for the subjects to receive all treatments. Two construction methods for incomplete designs are implemented in the package. One is `williams.BIB`, which combines balanced incomplete block designs (BIBD) and Williams designs (Patterson, 1951). The user needs to specify a balanced incomplete block design. Such designs may be found using the package **AlgDesign** (Wheeler, 2004). The function `find.BIB` provides a convenient way to use that package to get a BIBD. The last construction function considered here is `balmin.RMD`, a function that constructs the balanced minimal repeated measurements designs of Afsarinejad (1983). These designs are balanced for first order carryover effects but in general, they are not balanced for subject and order effects.

A more convenient way is to use the menu driven function `get.plan` documented below. The user specifies t, p and the maximum number of subjects available. The function checks which of the above functions may work for the given parameters. If there is no design that fits to the parameters, the design parameters may be adapted and a new choice of methods is presented.

For example, we may want to get a design for the comparison of 7 products. Assume that a maximum of 100 test persons are available and they may test all products within one session, i.e. $t = p = 7$ and $n = 100$. We have

```
> design <- get.plan(7,7,100)

Possible constructions and minimum
numbers of subjects:
          1        2
Method:  williams des.MOLS
Number:  14       42

Please choose one of the following
constructions
1:williams
2:des.MOLS
3:Exit
Selection:
```

`get.plan` suggests to use either a Williams design (which requires 14 subjects only) or to use a set of MOLS, which requires 42 subjects. Assume that we are interested in keeping the number of subjects as low as possible. We choose a williams design and type in the corresponding number:

```
> Selection: 1
williams selected. How many 'replicates'
do you wish (1 - 7 )?
Selection:
```

We choose not to replicate the design since that would mean additional subjects and type in `1`. If we wanted to get closer to the maximum number of subjects available we could have selected a design based on MOLS (for higher order balance) and two replicates of the design. That would have meant 84 subjects.

```
> Selection: 1
1 replicate(s) chosen
Row and treatment labels have been
randomized.
Rows represent subjects, columns
represent periods.
```

As indicated, the design is already randomized. A possible realization of the treatment sequences for the subjects is as follows:

```
> design
      [,1] [,2] [,3] [,4] [,5] [,6] [,7]
 [1,]    1    3    4    5    7    6    2
 [2,]    3    5    1    6    4    2    7
 [3,]    5    6    3    2    1    7    4
 [4,]    7    2    4    6    1    5    3
 [5,]    3    1    5    4    6    7    2
 [6,]    1    4    3    7    5    2    6
 [7,]    7    4    2    1    6    3    5
 [8,]    5    3    6    1    2    4    7
 [9,]    2    7    6    4    5    1    3
[10,]    6    2    5    7    3    4    1
[11,]    6    5    2    3    7    1    4
[12,]    4    1    7    3    2    5    6
[13,]    2    6    7    5    4    3    1
[14,]    4    7    1    2    3    6    5
```

Checking for Balance

Experimental plans that fit into the row-column scheme may be checked for balance. The function `isGYD` checks whether a given design is a balanced block design with respect to rows, columns or both rows and columns. The user specifies the matrix that represents the design. The design is then checked for balance. If the argument `tables` is set to `TRUE`, matrices giving the number of occurences of each treatment in each row and column as well as other tables describing the design are given. Continuing the example of the previous section we have

```
>  isGYD(design, tables=TRUE)

[1] The design is a generalized latin
square.

$"Number of occurrences of treatments
in d"
```

```
 1  2  3  4  5  6  7
14 14 14 14 14 14 14

$"Row incidence matrix of d"
  1 2 3 4 5 6 7 8 9 10 11 12 13 14
1 1 1 1 1 1 1 1 1 1  1  1  1  1  1

...

7 1 1 1 1 1 1 1 1 1  1  1  1  1  1

$"Column incidence matrix of d"
  1 2 3 4 5 6 7
1 2 2 2 2 2 2 2

...

7 2 2 2 2 2 2 2

$"Concurrence w.r.t. rows"
   1  2  3  4  5  6  7
1 14 14 14 14 14 14 14

...

7 14 14 14 14 14 14 14

$"Concurrence w.r.t. columns"
   1  2  3  4  5  6  7
1 28 28 28 28 28 28 28

...

7 28 28 28 28 28 28 28
```

The design is a generalized latin square which means that it's balanced for subject and period effects. We see for example that each product occurs 14 times within design. Each subject gets each product exactly once and each product appears in each period exactly twice.

Applying the function isCbalanced we see that our design is also balanced for first order carryover effects. Each product is preceded by every other product exactly twice but never by itself.

```
> isCbalanced(design)

The design is (first order) carry-over
balanced.

Left neighbour incidence matrix M_ij
(i is left neighbour of j)

     [,1] [,2] [,3] [,4] [,5] [,6] [,7]
[1,]    0    2    2    2    2    2    2
[2,]    2    0    2    2    2    2    2
[3,]    2    2    0    2    2    2    2
[4,]    2    2    2    0    2    2    2
[5,]    2    2    2    2    0    2    2
[6,]    2    2    2    2    2    0    2
[7,]    2    2    2    2    2    2    0
```

Summary

The package **crossdes** implements methods for the construction of balanced crossover designs. Block designs may be randomized and checked for balance. For details on the functions described above the user is referred to the package manual.

Acknowledgements

Support by the Deutsche Forschungsgemeischaft (SFB 475, Komplexitätsreduktion in multivariaten Datenstrukturen) is gratefully acknowledged.

Bibliography

Afsarinejad, K. (1983) Balanced repeated measurements designs. *Biometrika* **70**, 199–204. 25

Jones, B. and Kenward, M.G. (1989) *Design and Analysis of Cross-Over Trials.* Chapman and Hall, London. 24

Kunert, J. and Sailer, O. (2005) On Nearly Balanced Designs for Sensory Trials. To appear in *Food Quality and Preference*. 24

Patterson, H.D. (1951) Change-over trials. *Journal of the Royal Statistical Society B* **13**, 256–271. 25

Patterson, H.D. (1952) The construction of balanced designs for experiments involving sequences of treatments. *Biometrika* **39**, 32–48. 24

Senn, S. (1993) *Cross-over Trials in Clinical Research.* Wiley, New York. 24

Street, A.P. and Street, D.J. (1987) *Combinatorics of experimental design.* Oxford University Press, Oxford. 24

Wakeling, I. N. and MacFie, H. J. H. (1995) Designing consumer trials for first and higher orders of carry-over effects when only a subset of k samples from p may be tested. *Food Quality and Preference* **6**, 299–308. 24

Wheeler, R.E. (2004). optBlock. **AlgDesign**. The R project for statistical computing http://www.r-project.org/ 24, 25

Williams, E. J. (1949) Experimental designs balanced for the estimation of residual effects of treatments. *Australian Journal of Scientific Research, Ser. A* **2**, 149–168.

24, 25

Oliver Sailer
Fachbereich Statistik, Universität Dortmund, Germany
sailer@statistik.uni-dortmund.de

R Help Desk

Make 'R CMD' Work under Windows – an Example

Uwe Ligges and Duncan Murdoch

Introduction

During the last couple of months there have been a number of questions on both mailing lists 'R-help' and 'R-devel' on how to install R packages from source under Windows. Even though the procedure is documented in detail in the "R Installation and Administration" (R Development Core Team, 2005a) manual, people still seem to be uncertain about some details. This is perhaps partly caused by some confusing online material published by others who tried to help, but there is also the fact that requirements change over time, and the fact that the procedure is quite different from what most Windows users are used to doing.

To install packages from source, you will need to use the 'R CMD' commands at a command shell (which Windows users sometimes call a "DOS prompt", even though DOS is long gone from modern versions of Windows). A well set up environment is required. For recent versions of R, all you need to do is described in the "R Installation and Administration" manual (R Development Core Team, 2005a), Appendix "The Windows toolset". It is immensely important to follow the instructions given therein exactly. While some variations will work, you will cause yourself a lot of grief if you happen on one that does not.

We will give *an example* and describe what we did on a Windows XP machine to get a working environment. We give the URLs of downloads in the footnotes. These URLs work at the time of this writing (November, 2005), but may change over time. Check the Rtools web page[1] for updates. All the software will be installed into the directory 'c:\devel', but the user can choose arbitrary directories. It is highly recommended to avoid special characters and blanks in the directory names, though.

First, we have installed a recent (binary) version of R from a local CRAN mirror[2] into 'c:\devel\R-2.2.0' containing the executable in its 'bin' directory. Of course, it is also possible to install/compile R from sources itself, but this is a bit more complicated than just installing packages. Now follow the sections of the mentioned Appendix:

1. "The command line tools" are downloaded[3] and unzipped so that the executables are located in 'c:\devel\tools\bin'.

2. "Perl" is downloaded[4] and installed so that the executables are located in 'c:\devel\perl\bin'.

3. The current release of the "The MinGW compilers" is downloaded[5] and installed so that the executables are located in 'c:\devel\MinGW\bin'.

   Alternatively, one may download the recent versions of the components separately from http://www.mingw.org: 'gcc-core-VERSION.tar.gz', 'gcc-g++-VERSION.tar.gz', 'gcc-g77-VERSION.tar.gz', 'binutils-VERSION.tar.gz', 'mingw-runtime-VERSION.tar.gz', and 'w32api-VERSION.tar.gz' (replace 'VERSION' with the current version numbers) and unpack them manually after the rest of the setup. More details below.

4. "The Microsoft HTML Help Workshop" is downloaded[6] and installed to 'c:\devel\HtmlHelp\'.

5. "LaTeX": We choose the 'MiKTeX' distribution[7] and install it to 'c:\devel\texmf\' and 'c:\devel\localtexmf\', respectively. Therefore, the executables are in 'c:\devel\texmf\miktex\bin'.

The paths of 'Perl' and 'MiKTeX' are set automatically by their installers. However, you need to put the other tools on your path yourself. You can do this in the Windows Control Panel under "System | Advanced | Environment variables", but this will affect all programs on your system, and it may end up being modified when you install other software in the future. A better way is to create a small file

[1] http://www.murdoch-sutherland.com/Rtools
[2] A list of CRAN mirrors is given at http://CRAN.R-Project.org/mirrors.html.
[3] http://www.murdoch-sutherland.com/Rtools/tools.zip
[4] http://www.activestate.com/Products/ActivePerl/Download.html
[5] 'MinGW-5.0.0.exe' from http://sourceforge.net/projects/mingw/
[6] http://www.microsoft.com/office/ork/xp/appndx/appa06.htm
[7] http://www.miktex.org/setup.html