

# Package ‘crossdes’

April 17, 2009

**Date** 2008-06-30

**Title** Design and Randomization in Crossover Studies

**Version** 1.0-9

**Author** Martin Oliver Sailer <sailer@statistik.uni-dortmund.de>

**Depends** AlgDesign, gtools, MASS

**Description** Contains functions for the construction and randomization of balanced carryover balanced designs. Contains functions to check given designs for balance. Also contains functions for simulation studies on the validity of two randomization procedures.

**Maintainer** Martin Oliver Sailer <sailer@statistik.uni-dortmund.de>

**License** GPL-2

**Repository** CRAN

**Date/Publication** 2008-07-05 16:13:53

## R topics documented:

all.combin . . . . .	2
analyze.rand . . . . .	3
balmin.RMD . . . . .	5
biertest.d4 . . . . .	6
biertest.dat . . . . .	7
biertest.f2 . . . . .	7
des.MOLS . . . . .	8
find.BIB . . . . .	9
get.plan . . . . .	10
isCbalanced . . . . .	12
isGYD . . . . .	14
MOLS . . . . .	15
rand.design.azais . . . . .	16
rand.design.RC . . . . .	17
williams . . . . .	19
williams.BIB . . . . .	20

---

`all.combin`*Balanced Row-Column Design with all Possible Treatment Orders*

---

### Description

The function constructs a row-column design with subjects as rows and periods as columns. Each subject gets each treatment at most once. All possible treatment orders are assigned to the subjects.

### Usage

```
all.combin(trt, k)
```

### Arguments

<code>trt</code>	An integer $> 1$ . Number of treatments (products) to be tested.
<code>k</code>	An integer $\leq trt$ . Number of periods for each subject.

### Details

The design is a carryover balanced generalized Youden design that is uniform on the columns. The treatments are numbered  $1, \dots, trt$ . The entry  $(i, j)$  of the design corresponds to the treatment the  $i$ -th subject gets in the  $j$ -th period.

### Value

A matrix with  $\frac{trt!}{(trt-k)!}$  rows and  $k$  columns representing the experimental design.

### Note

Requires the package `gtools`.

### Author(s)

Oliver Sailer ([sailer@statistik.uni-dortmund.de](mailto:sailer@statistik.uni-dortmund.de))

### References

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

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

### See Also

[get.plan](#)

**Examples**

```
# Design for 4 treatments assigned in 3 periods.
# All possible treatment orders occur.
all.combin(4,3)
```

---

analyze.rand	<i>Analysis of a Simulation Study to Asses the Validity of a Randomization Procedure</i>
--------------	------------------------------------------------------------------------------------------

---

**Description**

The function analyzes the results of simulation studies done by functions like `rand.design.azais` or `rand.design.RC`. A table of results and, optionally, graphs are given that help to assess whether a randomization procedure validates a certain model for a given design.

**Usage**

```
analyze.rand(where, fig = FALSE, ref = FALSE, refval = numeric(6),
             reftext = "Reference Value", pch1 = 1, pch2 = 46, col1 = "red",
             col2 = "black", ...)
```

**Arguments**

where	Path that gives the location of the simulation results
fig	Logical flag. Set to TRUE if you want to display the results in graphs
ref	Logical flag. TRUE if you want to use your own reference values for the estimated contrast in the simulation study. Otherwise, the true values of the contrast are used.
refval	A numerical vector of length 6. The values that the contrast estimates are compared to. This is set automatically to the true values of the contrast if <code>ref</code> is FALSE.
reftext	Character string that contains the legend text in a graph displaying the location of the estimated contrast and the corresponding reference value.
pch1	A scalar that corresponds to the plotting character in the Q-Q-Plot of the difference of variance estimates that is displayed if <code>fig</code> is TRUE.
pch2	A scalar that corresponds to the plotting character in the Q-Q-Plot of the estimated contrast that is displayed if <code>fig</code> is TRUE.
col1	The colour of the theoretical CDF in the CDF-Plots that are displayed if <code>fig</code> is TRUE Also the colour of the reference values described above.
col2	The colour of the empirical CDF in the CDF-Plots that are displayed if <code>fig</code> is TRUE
...	Additional parameters for the graphs.

## Details

The input to the function comes from a file that is generated by the functions `rand.design.azais` or `rand.design.RC`. This file contains simulated values for contrast estimates and corresponding variance estimates. It also contains information on the experimental design and the model used.

The output contains two tables. The first one displays empirical quantiles of the randomization t-statistics for the contrast as well as estimates of the location of the contrasts. The second table contains estimates of the variance of the contrast. In theory, the randomization validates the model for the design used, if the contrast estimate is unbiased and the variance estimate of the contrast is unbiased, too. The simulation study suggests that this is achieved, if two conditions hold: First, the absolute value of the Z-statistic of table 1 is less than 1.96 (Gauss test on the randomization contrasts). Second, the confidence interval for the difference of the model variance estimate and the empirical variance of the randomization contrasts should include zero. The boundaries of the interval are given in columns 4 and 6 of table 2. An additional information on the validity is given by the randomization t-statistics. If the model is valid, for most real data they will be approximately t-distributed. This can be checked by comparing the empirical 5% quantile to the 5% quantile of the true t-distribution. The plots show histograms of the randomization contrast estimates and cumulative distribution functions for the permutation t-statistics. Also, normal Q-Q-Plots of the estimated contrast and the estimate of the difference between the true variance and the model variance of the contrast are given.

## Value

- 1 A 6\*5 matrix of results. The rows correspond to the cases considered in `rand.design.azais`. The first column contains the number of randomization t-statistics smaller than the 5% quantile of the theoretical t-distribution. The second column has the fraction of t-statistics smaller than this quantile. The third column contains the mean of the estimated contrast values. The fourth has the reference value for this mean. In the fifth column the test statistic for the Gauss test of the contrast being equal to the reference value is displayed.
- 2 A 6\*5 matrix of results. The rows correspond to the cases considered in `rand.design.azais`. The first column contains the empirical variance of the contrast estimates, the second column has the average value of the variance estimate for the contrast under the applied model. The following columns contain the lower, center and upper value of the confidence intervals for the difference of the true variance and the estimated variance of the estimated contrast.

## Note

If `fig` is TRUE, 24 graphics windows are opened.

## Author(s)

Oliver Sailer <sailer@statistik.uni-dortmund.de>

## References

Bailey, R.A. and Rowley C.A. (1987): Valid randomization. Proceedings of the Royal Society London A 410, 105-124.

Kunert, J. and Sailer, O. (2006): On nearly balanced designs for sensory trials. Food Quality and Preference 17, 219-227.

Kunert, J. and Sailer, O. (2007): Randomization of neighbour balanced generalized Youden designs. Journal of Statistical Planning and Inference 137, 2045-2055.

### See Also

[rand.design.azais](#), [rand.design.RC](#)

### Examples

```
## Not run:
# First create a data set to analyze:
d <- matrix(c(1:4,2:4,1,4,1:3,3,4,1,2),ncol=4)
rand.design.RC( d, rnorm(16), -1, 1, 1000, "D:\mytest.txt" )
# Now do the analysis:
analyze.rand( "D:\mytest.txt" )
analyze.rand( "D:\mytest.txt", fig=TRUE, ref=TRUE,
  refval=c(0, -1, 0, -1, -.25, -1.25) )
## End(Not run)
```

---

balmin.RMD

*Function to construct the balanced minimal repeated measurements designs of Afsarinejad (1983)*

---

### Description

The function constructs a row-column design with subjects as rows and periods as columns. The design is incomplete, i.e. no subject gets all the treatments. The design is balanced for carryover effects but will in general not be a balanced block design.

### Usage

```
balmin.RMD(trt, n, p)
```

### Arguments

trt	An integer >1 giving the number of treatments (products) to be tested.
n	An integer >1 giving the number of subjects (assessors) in the study.
p	An integer >1 giving the number of periods for each subject.

### Details

A necessary and sufficient condition for the existence of such a design is that  $\frac{(trt-1)}{(p-1)}$  be a positive integer. In this case  $n = \frac{trt(trt-1)}{(p-1)}$ . In the resulting design the treatments are numbered  $1, \dots, trt$ . The entry  $(i, j)$  of the design corresponds to the treatment the  $i$ -th subject gets in the  $j$ -th period.

**Value**

A matrix with  $n$  rows and  $p$  columns representing the experimental design.

**Author(s)**

Oliver Sailer <sailer@statistik.uni-dortmund.de>

**References**

- Afsarinejad, K. (1983): Balanced repeated measurements designs. *Biometrika* 70, 199-204.
- Wakeling, I.N. and MacFie, H.J.H. (1995): Designing consumer trials balanced for first and higher orders of carry-over effect when only a subset of  $k$  samples from  $t$  may be tested. *Food Quality and Preference* 6, 299-308.

**See Also**

[get.plan](#)

**Examples**

```
balmin.RMD(10,30,4) # a balanced minimal RMD
balmin.RMD(11,22,6) # another balanced minimal RMD
```

---

biertest.d4

*Experimental Design for the Beer Testing Data in Kunert (1998)*

---

**Description**

The design is a balanced incomplete block design with rows as blocks. If one assumes that there is a preperiod, i.e. that each assessor is given the treatment of the fifth period before the experiment starts, then the design is carryover balanced.

**Usage**

```
data(biertest.d4)
```

**Format**

A matrix with 12 rows corresponding to the assessors, 5 columns corresponding to the five periods.

**Details**

The five treatments (products) to be tested are numbered 1, ..., 5.

**Source**

Kunert, J. (1998): Sensory experiments as crossover studies. *Food Quality and Preference* 9, 243-253 (design d4).

**Examples**

```
data(biertest.d4)
```

---

```
biertest.dat
```

*The Beer Testing Data in Kunert (1998)*

---

**Description**

The data comes from a beer testing experiment where assessors had to rate the bitterness of five beers.

**Usage**

```
data(biertest.dat)
```

**Format**

A matrix with 12 rows corresponding to the assessors, 5 columns corresponding to the five periods.

**Details**

The possible values for the ratings lie in the interval  $[0, 12]$ . Note that the data were reordered after the experiment.

**Source**

Kunert, J. (1998): Sensory experiments as crossover studies. *Food Quality and Preference* 9, 243-253 (table 1).

**Examples**

```
data(biertest.dat)
```

---

```
biertest.f2
```

*Experimental Design for the Beer Testing Data in Kunert (1998)*

---

**Description**

The design is a row-column design. It is actually a generalized latin square design that is also carryover balanced.

**Usage**

```
data(biertest.d4)
```

**Format**

A matrix with 10 rows corresponding to the first 10 assessors in a fictional uniformity trial, 5 columns corresponding to the five periods.

**Details**

There are five products to be tested, numbered 1, . . . , 5.

**Source**

Kunert, J. (1998): Sensory experiments as crossover studies. *Food Quality and Preference* 9, 243-253 (design f2).

**Examples**

```
data(biertest.f2)
```

---

 des.MOLS

---

*Construction of Designs Based on Mutually Orthogonal Latin Squares*


---

**Description**

The function constructs row-column designs based on complete sets of mutually orthogonal latin squares. Each subject may get each treatment at most once. The design is a generalized Youden design that is also balanced for carryover effects.

**Usage**

```
des.MOLS(trt, k = trt)
```

**Arguments**

trt	A prime power less than 100. The number of treatments (products) to be tested.
k	An integer $\leq$ trt. Number of periods for each subject.

**Details**

A complete set of mutually orthogonal latin squares is constructed using Galois Fields. The rows of the designs represent the treatment orders for the subjects. If an incomplete design with  $k$  columns is needed, only the first  $k$  columns of the designs are considered. The treatments are numbered 1, . . . , trt. The entry  $(i, j)$  of the design corresponds to the treatment the  $i$ -th subject gets in the  $j$ -th period.

**Value**

A matrix with  $trt(trt - 1)$  rows and  $k$  columns representing the experimental design.

**Author(s)**

Oliver Sailer <sailer@statistik.uni-dortmund.de>

**References**

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

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.

**See Also**

[get.plan](#), [MOLS](#)

**Examples**

```
des.MOLS(7, 7)
des.MOLS(8, 5)
```

---

find.BIB

*Generation of Balanced Incomplete Block Designs Using the Package AlgDesign*

---

**Description**

The function `optBlock` of the library `AlgDesign` is used to search for balanced incomplete block designs (BIBDs). The design is assigned to a matrix where rows represent blocks (subjects) and columns represent periods.

**Usage**

```
find.BIB(trt, b, k, iter = 30)
```

**Arguments**

<code>trt</code>	An integer > 1 giving the number of treatments of the design.
<code>b</code>	An integer > 1 giving the number of rows (subjects) of the design.
<code>k</code>	An integer > 1 giving the number of columns (periods) of the design.
<code>iter</code>	The number of iterations of the function <code>optBlock</code>

**Details**

The function `optBlock` tries to find a D-optimal block design for the specified parameters. The resulting design need not be a BIBD. The necessary conditions for the existence are that  $\frac{bk}{trt}$  and  $\frac{bk(k-1)}{trt(trt-1)}$  positive integers. They are NOT checked automatically. Even if they are fulfilled, there need not be a BIBD. If no BIBD is found, the function is iterated. If no BIBD is found after `iter` iterations, the search is terminated. The resulting design should be checked by the user applying `isGYD`.

**Value**

A matrix that represents the experimental design.

**Note**

As indicated above, the returned design is not necessarily a BIBD design.

**Author(s)**

Oliver Sailer <sailer@statistik.uni-dortmund.de>

**References**

Wheeler, R.E. (2004). `optBlock`. `AlgDesign`. The R project for statistical computing <http://www.r-project.org/>

**See Also**

[get.plan](#), [optBlock](#)

**Examples**

```
find.BIB(10, 30, 4)      # BIBD
find.BIB(3, 3, 3)      # BIBD
find.BIB(5, 5, 3, 100) # There is no BIBD for these parameters
# isGYD(find.BIB(5, 5, 3, 100))
```

---

`get.plan`

*Menu-Driven Construction of Carryover Balanced Experimental Designs*

---

**Description**

This menu based function constructs and randomizes simple experimental designs for repeated measurements with one or two block variables. It is assumed that each subject is assigned to each treatment at most once. A maximum number of subjects in the study is also requested. Five possible construction methods available. These construction methods and the characteristics of the resulting designs are described in Wakeling and MacFie (1995). See also Jones and Kenward (1989), Ch. 5, for a discussion of these designs. The function is demonstrated in more detail in Sailer (2005).

**Usage**

```
get.plan(trt, k = trt, maxsub = 1000, random = TRUE)
```

**Arguments**

trt	An integer > 1, giving the number of treatments.
k	An integer in $\{2, \dots, trt\}$ giving the number of periods.
maxsub	The maximum number of subjects available.
random	Logical flag. If TRUE, the design is randomized after construction.

**Details**

The five types of designs are: designs based on all possible treatment orders ("all.combin"), Williams designs ("williams"), designs based on mutually orthogonal latin squares ("des.MOLS"), a combination of balanced incomplete block designs (BIBDs) and Williams designs ("williams.BIB") by Patterson (1951) and the balanced minimal designs of Afsarinejad ("balmin.RMD"). Some designs are only available for special combinations of treatment number and number of periods. Other designs may require too many subjects. Therefore, the possible choices available for the submitted values of *trt*, *k* and *maxsub* are determined. If there is no design available, the parameters may be changed interactively. If more than one design type is available the user has to choose one. The minimum number of subjects required for the designs is given and may be a criterion for selecting a design. All types of designs are balanced for first-order carryover effects. All types except the balanced minimal RMDs are also balanced block designs. The user may want to construct a design for a multiple of the minimum number of subjects required to get closer to the preferred number of subjects. Once the design is chosen, the labels for the treatments and subjects should be randomized and the design is displayed. The treatments are numbered  $1, \dots, trt$ . The entry  $(i, j)$  of the design corresponds to the treatment the *i*-th subject gets in the *j*-th period.

**Value**

A matrix representing the experimental design.

**Warning**

There is a possible problem with this implementation of the "williams.BIB" approach:

For the construction of designs that combine BIBDs with Williams designs, the function `find.BIB` is called to search for a BIBD. If the necessary conditions for the existence of a BIBD are fulfilled, this approach always returns a design. This design will however not always be a BIBD! When using the patterson approach, please check the resulting design for balance using `isGYD` and `isCbalanced`.

It should be noted that this is a computational problem only, not a problem of the theoretical approach of Patterson (1951).

**Note**

The "All combinations" approach requires the package `gttools`.

**Author(s)**

Oliver Sailer (sailer@statistik.uni-dortmund.de)

**References**

- Afsarinejad, K. (1983): Balanced repeated measurements designs. *Biometrika* 70, 199-204.
- Jones, B. and Kenward, M.G. (1989): *Design and Analysis of Cross-Over Trials*. Chapman and Hall, London.
- Patterson, H.D. (1951): Change-over trials. *Journal of the Royal Statistical Society B* 13, 256-271.
- Patterson, H.D. (1952): The construction of balanced designs for experiments involving sequences of treatments. *Biometrika* 39, 32-48.
- Sailer, O. (2005): crossdes: A package for design and randomization in crossover studies. *Rnews* 5, 24-27.
- Wakeling, I.N. and MacFie, H.J.H. (1995): Designing consumer trials balanced for first and higher orders of carry-over effect when only a subset of k samples from t may be tested. *Food Quality and Preference* 6, 299-308.
- 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.

**See Also**

[all.combin](#), [balmin.RMD](#), [des.MOLS](#), [williams](#), [williams.BIB](#)

**Examples**

```
## Not run:
get.plan(10,4,60)
# "williams.BIB" or "balmin.RMD"
get.plan(7,7,7000)
# "all.combin", "williams" or "des.MOLS", "williams" requires
# only 14 subjects, "all.combin" requires 5040.
get.plan(5,5,5)
# Increase maxsub
## End(Not run)
```

---

isCbalanced

*Checking Block Designs for Carryover Balance*

---

**Description**

The function checks whether a block design is balanced for first order carryover effects (residual effects). The user specifies whether there is a preperiod. The design is checked and the left neighbour incidence matrix is given.

**Usage**

```
isCbalanced(d, preperiod = FALSE)
```

**Arguments**

d	A matrix with entries $1, \dots, trt$ representing the experimental design with rows as blocks (subjects). The columns represent periods.
preperiod	Logical flag. TRUE if there is a preperiod. In this case, each subject experiences in the first period the residual effect of the treatment of the last period (i.e. the last period precedes the first period, i.e. the plots in the last period are left neighbours of the plots in the first period). FALSE if there are no residual effects in the first period.

**Details**

The design is said to be carryover balanced (balanced for first order carryover effects), if each treatment is preceded by all other treatments equally often and if no treatment is preceded by itself. If the design is balanced, this is stated.

**Value**

1	Logical flag. TRUE if the design is carryover balanced. This is not displayed on the screen.
2	Left neighbour incidence matrix. The $(i, j)$ -th element is the number of times that treatment $i$ precedes treatment $j$ .

**Author(s)**

Oliver Sailer <sailer@statistik.uni-dortmund.de>

**See Also**

[isGYD](#)

**Examples**

```
d1 <- matrix( c(1,2,3,4,1,1,1,1), 4, 2)
d2 <- matrix( c(1:4,2:4,1,4,1:3,3,4,1,2), ncol=4)
d3 <- matrix( rep(1:3,each=2), ncol=2)
isCbalanced(d1)
isCbalanced(d1, TRUE)
isCbalanced(d2)
isCbalanced(d3, TRUE)
```

**Description**

A function to check a simple block or a row-column design for balance. The rows and columns of the design are blocking variables. It is checked which type of balance the design fulfills. Optionally, incidence and concurrence matrices are given.

**Usage**

```
isGYD(d, tables=FALSE, type=TRUE)
```

**Arguments**

d	A matrix representing the experimental design. The treatments must be numbered $1, \dots, trt$ .
tables	Logical flag. If TRUE, incidence matrices are displayed.
type	Logical flag. If TRUE, the type of design is displayed.

**Details**

A design is said to be a balanced block design if the following three conditions hold: i) Each treatment appears equally often in the design. ii) The design is binary in the sense that each treatment appears in each block either  $n$  or  $n+1$  times where  $n$  is an integer. iii) The number of concurrences of treatments  $i$  and  $j$  is the same for all pairs of distinct treatments  $(i, j)$ . Here the blocks are either rows or columns.

A design that has less columns (rows) than treatments is said to be incomplete with respect to rows (columns). A design that is balanced with respect to both rows and columns is called a generalized Youden design (GYD). A GYD for which each treatment occurs equally often in each row (column) is called uniform on the rows (columns). If both conditions hold, it is called a generalized latin square. A design where each treatment occurs exactly once in each row and column is called a latin square.

**Value**

A list containing information about balance in rows and columns as well as incidence and concurrence matrices for the design.

**Author(s)**

Oliver Sailer <sailer@statistik.uni-dortmund.de>

**See Also**

[isCbalanced](#)

**Examples**

```

d1 <- matrix( c(1,2,3,4,1,1,1,1), 4,2)
# d1 is not balanced
d2 <- matrix( c(1:4,2:4,1,4,1:3,3,4,1,2), ncol=4)
# d2 is a latin square
d3 <- matrix( rep(1:3,each=2), ncol=2)
# d3 is a balanced incomplete block design.
d1
isGYD(d1,tables=TRUE)
d2
isGYD(d2,tables=TRUE)
d3
isGYD(d3,tables=TRUE)

```

MOLS

*Construction of Complete Sets of Mutually Orthogonal Latin Squares***Description**

The function constructs sets of mutually orthogonal latin squares (MOLS) using Galois fields. The construction works for prime powers only.

**Usage**

```
MOLS(p, n, primpol = GF(p, n)[[2]][1, ])
```

**Arguments**

**p** A prime number less than 100.  
**n** A positive integer.  
**primpol** A primitive polynomial of the Galois Field  $GF(p^n)$ .

**Details**

If  $trt = p^n$  is a prime power, then  $trt-1$  latin squares of order  $trt$  are constructed. The elements of the squares are numbered  $1, \dots, trt$ . These squares are mutually orthogonal, i.e. if any two of them are superimposed, the resulting array will contain each ordered pair  $(i, j)$ ,  $i, j$  in  $\{1, \dots, trt\}$  exactly once. The squares are in standard order, i.e. the first row is always equal to  $(1, \dots, trt)$ . A primitive polynomial may be constructed automatically using the internal function `GF`.

**Value**

For  $trt = p^n$ , an array that contains  $trt-1$  latin squares is returned.

**Author(s)**

Oliver Sailer (sailer@statistik.uni-dortmund.de)

## References

Cherowitzo, W.: <http://www-math.cudenver.edu/~wcherowi/courses/finflds.html>

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

## See Also

[des.MOLS](#)

## Examples

```
MOLS(7,1) # 6 mutually orthogonal latin squares of order 7
MOLS(2,3) # 7 mutually orthogonal latin squares of order 8
```

---

rand.design.azais *Simulation Study to Asses the Validity of a Randomization Procedure*

---

## Description

The function performs a simulation study to assess whether a randomization procedure proposed by Azais (1987) validates the simple block model for a given design. The results are stored to a file.

## Usage

```
rand.design.azais(design, dat, tau1, rho, n, where)
```

## Arguments

design	A matrix with $b$ rows and $k$ columns representing the experimental design. Treatments are numbered $1, \dots, trt$ .
dat	A numerical vector with $bk$ elements giving the data to be used for the simulation study. The first $k$ values of <code>dat</code> correspond to the first row of the design, the next $k$ values correspond to the second row etc.
tau1	The value of the main effect of treatment 1.
rho	The value that is used for the carryover (residual) effects of treatments 1 and 2.
n	The number of permutations in the simulation study.
where	Path that gives the location of the simulation results.

## Details

The simulation study proceeds as follows: For every iteration, treatment labels and rows of the design are randomized. Then the treatment order in each row is permuted cyclically. Then the elementary contrast  $\tau_{u_1} - \tau_{u_{trt}}$  is estimated and the estimate of the variance of this contrast is computed. These computations are done for each of six situations: 1) There are no direct or residual effects of treatments. 2) There is a direct effect of treatment 1. In 3) and 4), a residual effect of treatment 2 is added while in 5) and 6), a residual effect of treatment 1 is added. The estimates are then stored to `where`.

**Value**

There is no value returned. The results are stored in a file.

**Note**

You need to call `analyze.rand` to display and interpret the results. `rand.design.azais` just performs the simulation study.

**Author(s)**

Oliver Sailer ([sailer@statistik.uni-dortmund.de](mailto:sailer@statistik.uni-dortmund.de))

**References**

- Azais, J.M. (1987): Design of experiments for studying intergenotypic competition. *Journal of the Royal Statistical Society B* 49, 334-345.
- Bailey, R.A. and Rowley, C.A. (1987): Valid randomization. *Proceedings of the Royal Society London A* 410, 105-124.
- Kunert, J. and Sailer, O. (2006): On nearly balanced designs for sensory trials. *Food Quality and Preference* 17, 219-227.

**See Also**

[analyze.rand](#), [rand.design.RC](#)

**Examples**

```
## Not run:

# First create a data set to analyze:
d <- matrix(c(1,1,1,2,2,3,4,4,3,4,2,3), ncol=3)
rand.design.azais( d, rnorm(12), -1, 1, 1000, "D:\mytest.txt" )
# Now do the analysis:
analyze.rand( "D:\mytest.txt", fig=TRUE, ref=TRUE,
  refval=c(0, -1, 0, -1, -.5, -1.5) )
## End(Not run)
```

---

rand.design.RC

*Simulation Study to Assess the Validity of a Randomization Procedure*

---

**Description**

The function performs a simulation study to assess whether randomization of treatment labels and rows (subjects) validates the row-column model for a given design. The results are stored to a file.

**Usage**

```
rand.design.RC(design, dat, taul, rho, n, where)
```

**Arguments**

design	A matrix with $b$ rows and $k$ columns representing the experimental design. Treatments are numbered $1, \dots, trt$ .
dat	A numerical vector with $bk$ elements giving the data to be used for the simulation study. The first $k$ values of <code>dat</code> correspond to the first row of the design, the next $k$ values correspond to the second row etc.
tau1	The value of the main effect of treatment 1.
rho	The value that is used for the carryover (residual) effects of treatments 1 and 2.
n	The number of permutations in the simulation study.
where	Path that gives the location of the simulation results.

**Details**

The simulation study proceeds as follows: For every iteration, treatment labels and rows of the design are randomized. Then the elementary contrast  $\tau_{u_1} - \tau_{u_{trt}}$  is estimated and the estimate of the variance of this contrast is computed. These computations are done for each of six situations: 1) There are no direct or residual effects of treatments. 2) There is a direct effect of treatment 1. In 3) and 4), a residual effect of treatment 2 is added while in 5) and 6), a residual effect of treatment 1 is added. The estimates are then stored to `where`.

**Value**

There is no value returned. The results are stored in a file.

**Note**

You need to call `analyze.rand` to display and interpret the results. `rand.design.RC` just performs the simulation study.

**Author(s)**

Oliver Sailer ([sailer@statistik.uni-dortmund.de](mailto:sailer@statistik.uni-dortmund.de))

**References**

- Bailey, R.A. and Rowley, C.A. (1985): Valid randomization. Proceedings of the Royal Society London A 410, 105-124.
- Kunert, J. and Sailer, O. (2006): On nearly balanced designs for sensory trials. Food Quality and Preference 17, 219-227.

**See Also**

[analyze.rand](#), [rand.design.azais](#)

## Examples

```
## Not run:

# First create a data set to analyze:
d <- matrix(c(1:4,2:4,1,4,1:3,3,4,1,2),ncol=4)
rand.design.RC( d, rnorm(16), -1, 1, 1000, "D:\mytest.txt" )
# Now do the analysis:
analyze.rand( "D:\mytest.txt", fig=TRUE, ref=TRUE,
  refval=c(0, -1, 0, -1, -.25, -1.25) )
## End(Not run)
```

---

williams

*Construction of Williams Designs*

---

## Description

The function constructs williams designs. Williams designs are row-column designs. They are used if each of the treatments in the study is given to each of the subjects. If the number of treatments to be tested is even, the design is a latin square, otherwise it consists of two latin squares.

## Usage

```
williams(trt)
```

## Arguments

`trt` An integer > 1, giving the number of treatments in the design.

## Details

The resulting design is a (generalized) latin square that is also balanced for first order carryover effects. Carryover balance is achieved with very few subjects. In the experimental design the treatments are numbered  $1, \dots, trt$ . The entry  $(i, j)$  of the design corresponds to the treatment the  $i$ -th subject gets in the  $j$ -th period.

## Value

A matrix representing the experimental design.

## Author(s)

Oliver Sailer (sailer@statistik.uni-dortmund.de)

**References**

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

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.

**See Also**

[get.plan](#)

**Examples**

```
williams(3)
williams(10)
```

---

williams.BIB

*Construction of Carryover Balanced Designs Based on Balanced Incomplete Block Designs*

---

**Description**

Patterson (1951) combined balanced incomplete block designs (BIBDs) with Williams designs to get carryover balanced generalized Youden designs.

**Usage**

```
williams.BIB(d)
```

**Arguments**

`d` A matrix representing a BIBD. Rows represent blocks (subjects).

**Details**

For each row of the design, a Williams design is constructed using the treatments of that row. The rows of the resulting designs are then combined. The treatments are numbered  $1, \dots, trt$ . The entry  $(i, j)$  of the design corresponds to the treatment the  $i$ -th subject gets in the  $j$ -th period.

**Value**

A matrix representing the experimental design.

**Warning**

The resulting design is only balanced properly if the input design actually is a BIBD. This is NOT checked automatically. You have to do this by yourself, e.g. by applying `isGYD` to your design.

**Note**

BIBDs may be generated using `find.BIB`.

**Author(s)**

Oliver Sailer <sailer@statistik.uni-dortmund.de>

**References**

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

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

**See Also**

`get.plan`, `isGYD`, `find.BIB`, `williams`

**Examples**

```
d <- matrix( rep(1:3,each=2), ncol=2)
# # check for balance
# isGYD(d)
williams.BIB(d)
```

# Index

## \*Topic **datasets**

biertest.d4, [6](#)  
biertest.dat, [7](#)  
biertest.f2, [7](#)

## \*Topic **design**

all.combin, [1](#)  
analyze.rand, [3](#)  
balmin.RMD, [5](#)  
des.MOLS, [8](#)  
find.BIB, [9](#)  
get.plan, [10](#)  
isCbalanced, [12](#)  
isGYD, [13](#)  
MOLS, [15](#)  
rand.design.azais, [16](#)  
rand.design.RC, [17](#)  
williams, [19](#)  
williams.BIB, [20](#)

rand.design.RC, [5](#), [17](#), [17](#)

williams, [12](#), [19](#), [21](#)

williams.BIB, [12](#), [20](#)

all.combin, [1](#), [12](#)

analyze.rand, [3](#), [17](#), [18](#)

balmin.RMD, [5](#), [12](#)

biertest.d4, [6](#)

biertest.dat, [7](#)

biertest.f2, [7](#)

des.MOLS, [8](#), [12](#), [15](#)

find.BIB, [9](#), [21](#)

get.plan, [2](#), [6](#), [9](#), [10](#), [10](#), [19](#), [21](#)

isCbalanced, [12](#), [14](#)

isGYD, [13](#), [13](#), [21](#)

MOLS, [9](#), [15](#)

optBlock, [10](#)

rand.design.azais, [4](#), [5](#), [16](#), [18](#)