Package 'blocksdesign'

October 18, 2018

Type Package

Title Nested and Crossed Block Designs for Factorial, Fractional Factorial and Unstructured Treatment Sets

Version 3.1

Date 2018-10-18"

Author R. N. Edmondson.

Maintainer Rodney Edmondson <rodney.edmondson@gmail.com>

Depends R (>= 3.1.0)

Description Constructs D-optimal or near D-optimal nested and crossed block designs for unstructured or general factorial treatment designs. Where a structured treatment design is required, a D-optimal or near D-optimal treatment design is found based on a suitable model matrix design formula. The required block design is then found for the required treatment design based on a defined set of block factors. The block factors are added in sequence and each added block factor is optimized conditional on all previously added block factors. The block design can have repeated nesting down to any required depth of nesting with either simple nested blocks or a crossed blocks design at each level of nesting. Outputs include a table showing the allocation of treatments to blocks and tables showing the achieved D-efficiency factors for each block and treatment design.

License GPL (>= 2)

Imports lme4, crossdes

LazyData true

RoxygenNote 6.1.0

Encoding UTF-8

NeedsCompilation no

Repository CRAN

Date/Publication 2018-10-18 11:10:06 UTC

R topics documented:

`blocksdesign-package` *Blocks design package*

Description

The `blocksdesign` package provides functionality for the construction of multi-level nested or crossed block designs for any feasible combination of qualitative or quantitative level treatment factors.

Details

Block designs aim to group experimental units into homogeneous blocks to provide maximum precision of estimation of treatment effects within blocks. The most basic type of block design is a complete randomized blocks design where every block contains one or more complete replicate sets of treatments. Complete randomized block designs estimate all treatment effects fully within individual blocks and are usually the best choice for small experiments. However, for large experiments, the average variability within complete replicate blocks can be large and then it may be beneficial to sub-divide each complete replicate block into smaller incomplete blocks which give improved precision of comparison on inter-block treatment effects.

Block designs with a single level of nesting are widely used in practical research but sometimes a single set of nested blocks may still be too large to give good control of intra-block variability. In this situation, a second set of incomplete blocks can be nested within the first set to reduce the intra-block variability still further. This process of recursive nesting can be repeated as often as required until the bottom set of blocks is sufficiently small to give good control of intra-block variability.

Sometimes it can be advantageous to use a double blocking system in which one set of blocks, usually called row blocks, is crossed with a second set of blocks, usually called column blocks. Double blocking systems can be valuable for controlling block effects in two dimensions simultaneously.

The `blocksdesign` package provides functionality for the construction of general multi-level block designs with nested or crossed blocks for any feasible depth of nesting. The design algorithm proceeds recursively with each nested set of blocks optimized conditionally within each preceding set of blocks. The analysis of incomplete block designs is complex but the availability of modern computers and modern software, for example the R mixed model software package `lme4` (Bates et. al. 2014), makes the analysis of any feasible nested block designs with any depth of nesting practicable.

The `blocksdesign` function has two main design functions:

i) design

This function provides general block designs for unstructured treatment sets or for factorial treatment sets with qualitative or quantitative level treatment factors. The function finds a D-optimal or near D-optimal treatment design of the required size, possibly a simple unstructured treatment set, and then finds a D-optimal or near D-optimal block design for that treatment design. The design algorithm builds the blocks design by sequentially adding blocks factors where each added block factor is optimized conditional on all the previously added block factors. Sequential optimization allows the blocking factors to be fitted in order of importance with the largest and most important blocks fitted first and the smaller and less important blocks fitted subsequently. If there are no defined bock factors, the algorithm assumes a completely randomised treatment design The output from design includes a data frame of the block and treatment factors for each plot and a table showing the achieved D-efficiency factors for each set of nested or crossed blocks. Fractional factorial efficiency factors based on the generalized variance of the complete factorial design are also shown (see the design documentation for details)

ii) blocks

This function generates arbitrary nested block designs for unstructured treatment sets where it can be assumed that all block sizes at any particular level of nesting are all as near equal as possible. Special block designs such as lattice designs or latin or Trojan square designs are constructed algebraically. The outputs from the blocks function include a data frame showing the allocation of treatments to blocks for each plot of the design and a table showing the achieved D- and A-efficiency factors for each set of nested blocks together with A-efficiency upper bounds, where available. A plan showing the allocation of treatments to blocks in the bottom level of the design is also included in the output.

References

BATES D., MAECHLER M., BOLKER B., WALKER S. (2015). Fitting Linear Mixed-Effects Models Using lme4. Journal of Statistical Software, 67(1), 1-48. doi:10.18637/jss.v067.i01.

---

blockEfficiencies *Efficiency factors*

Description

Finds efficiency bounds for block designs.

Usage

```
blockEfficiencies(TF, BF, treatments_model = NULL)
```

Arguments

TF	the treatments factor data frame
BF	the block factors data frame
treatments_model	the treatments model formula for the treatments data frame. Default is a full factorial model.

Details

efficiency factors of regular block designs

`blocks` *Block designs for unstructured treatment sets*

Description

Constructs randomized nested block designs for unstructured treatment sets with any feasible depth of nesting.

Usage

```
blocks(treatments, replicates, blocks = NULL, searches = NULL,
  seed = NULL, jumps = 1)
```

Arguments

`treatments`	a partition of the total required number of treatments into equally replicated treatment sets.
`replicates`	a set of treatment replication numbers with one replication number for each partitioned treatment set.
`blocks`	the number of blocks nested in each preceding block for each level of nesting from the top-level block downwards. The top-level block is a single super-block which need not be defined explicitly.
`searches`	the maximum number of local optima searched for a design optimization. The default number decreases as the design size increases.
`seed`	an integer initializing the random number generator. The default is a random seed.
`jumps`	the number of pairwise random treatment swaps used to escape a local maxima. The default is a single swap.

Details

Constructs randomized nested block designs with arbitrary depth of nesting for arbitrary unstructured treatment sets.

The `treatments` parameter is a set of numbers that partitions the total number of treatments into equally replicated treatment sets while the `replicates` parameter is a matching set of numbers that defines the replication of each equally replicated treatment set.

The `blocks` parameter, if any, defines the number of blocks for each level of nesting from the highest to the lowest. The first number, if any, is the number of nested row blocks in the first-level of nesting, the second number, if any, is the number of nested row blocks in the second-level of nesting and so on down to any required feasible depth of nesting.

Block sizes are as nearly equal as possible and will never differ by more than a single plot for any particular block classification.

Unreplicated treatments are allowed and any simple nested block design can be augmented by any number of single unreplicated treatments to give augmented blocks that never differ in size by more than a single plot. General crossed block designs are more complex and currently

Square lattice designs are resolvable incomplete block designs for r replicates of p*p treatments arranged in blocks of size p where r < p+2 for prime or prime power p or r < 4 for general p. Square lattice designs are constructed algebraically from Latin squares or MOLS.

Lattice designs based on prime-power MOLS require the MOLS package.

All other designs are constructed numerically by optimizing a D-optimality criterion.

Outputs:

- A data frame showing the allocation of treatments to blocks with successive nested strata arranged in standard block order.
- A table showing the replication number of each treatment in the design.
- A table showing the block levels and the achieved D-efficiency and A-efficiency factor for each nested level together with A-efficiency upper bounds, where available.
- A plan showing the allocation of treatments to blocks in the bottom level of the design.

Value

`Treatments`	A table showing the replication number of each treatment in the design.
`Design`	Data frame giving the optimized block and treatment design in plot order.
`Plan`	Data frame showing a plan view of the treatment design in the bottom level of the design.
`BlocksEfficiency`	The D-efficiencies and the A-efficiencies of the blocks in each nested level of the design together with A-efficiency upper-bounds, where available.
`seed`	Numerical seed used for random number generator.
`searches`	Maximum number of searches used for each level.
`jumps`	Number of random treatment swaps used to escape a local maxima.

References

Sailer, M. O. (2013). crossdes: Construction of Crossover Designs. R package version 1.1-1. https://CRAN.R-project.org/package=crossdes

Cochran, W.G., and G.M. Cox. 1957. Experimental Designs, 2nd ed., Wiley, New York.

Examples

```
## The number of searches in the following examples have been limited for fast execution.
## In practice, the number of searches may need to be increased for optimum results.
## Designs should be rebuilt several times to check that a near-optimum design has been found.


## Unstructured treatments partitioned into equally replicated treatment sets

# 3 treatments x 2 replicates + 2 treatments x 4 replicates in two complete randomized blocks
blocks(treatments=c(3,2),replicates=c(2,4),searches=10)

# 50 treatments x 4 replicates with 4 main blocks and 5 nested sub-blocks in each main block
blocks(treatments=50,replicates=4,blocks=c(4,5))

# as above but with 20 additional single replicate treatments, one single treatment per sub-block
blocks(treatments=c(50,20),replicates=c(4,1),blocks=c(4,5))

# 6 replicates of 6 treatments in 4 blocks of size 9 (non-binary block design)
blocks(treatments=6,replicates=6,blocks=4)

# 128 treatments x 2 replicates with two main blocks and 3 levels of nesting
blocks(128,2,c(2,2,2,2))

#' # 64 treatments x 4 replicates with 4 main blocks nested blocks of size 8 (lattice square)
blocks(64,4,c(4,8))

# 100 treatments x 4 replicates with 4 main blocks nested blocks of size 10 (lattice square)
blocks(100,4,c(4,10))
```

`design` *General block and treatment designs.*

Description

Constructs block and treatment designs for any feasible combination of nested or crossed block factors and any feasible combination of qualitative or quantitative level treatment factors.

Usage

```
design(treatments, blocks, treatments_model = NULL, weighting = 0.5,
  searches = NULL, seed = NULL, jumps = 1)
```

Arguments

`treatments`	a single treatment factor or a data frame containing one or more qualitative or quantitative level treatment factors.

`blocks` a single blocks factor or a data frame containing one or more qualitative level block factors in the required order of fitting.

`treatments_model`
a model formula for the treatments design.

`weighting` a weighting factor between 0 and 1 for the relative importance of the block factor interaction effects versus the block factor main effects for a crossed blocks factor design.

`searches` the maximum number of local optima searched at each stage of a design optimization

`seed` an integer initializing the random number generator.

`jumps` the number of pairwise random treatment swaps used to escape a local maxima.

Details

`treatments` is a factor or a data frame for one or more qualitative or quantitative level treatment factors. The `treatments` object is the candidate set of treatments or treatment factor combinations from which the treatment design is selected and should include all the possible feasible treatment combinations that could or should be included in the final design.

`blocks` is a factor or data frame for one or more qualitative block factors where the length or number of rows of `blocks` is the number of plots in the design. If the design is a completely randomized design, the `blocks` object should be a blocks factor of the required length but with only a single factor level.

The ratio of the length of the `blocks` object to the length of the `treatments` object is the replication number and he integer part of the replication number, if any, defines the number of complete replications of the `treatments` object while the fractional part, if any, defines a sample fraction of that size drawn from the candidate set of `treatments` combinations.

The blocks design can comprise any ordered set of crossed or nested block factors provided only that the nested blocks are arranged in a hierarchical order of nesting from largest to smallest. The block factors are added in sequence and each successively added blocks factor is optimised by maximising the current blocks model but with all the previously added blocks factors held constant.

N.B. crossed block designs can fail due to inherent aliasing. For example, `blocksdesign` will always try to fit orthogonal row and column block effects which means that a row-and-column design with two rows, two columns and two treatment replicates will automatically confound one treatment contrast with the rows-by-columns interaction effect, which means that a full second-order block design will always be singular.

`treatments_model` is a design formula for the `treatments` model based on the `models` formula of the `model.matrix` package. The model fits factorial treatment contrasts for qualitative level factors and polynomial treatment contrastst for quantitative level factors. The default treatment model assumes fully crossed factor effects for qualitative level factors and first-order effects for for quantitative level factors.

`weighting` is a weighting parameter between 0 and 1 which differentially weights the higher-order effects of a crossed blocks design relative to the the first-order effects. Designs with crossed blocks are usually assumed to fit an additive main block effects model but this is a very strong assumption and wherever possible, crossed blocks designs should allow for higher-order interaction effects. Setting the weighting parameter between 0 and 1 gives increasing importance to the higher-order

effects in a blocks model relative to the first-order effects as the weighting increases from 0 to 1. The default weighting is 0.5 (only applies when a design has crossed blocks).

`seed` is an integer which can be used to set the initial seed for the `blocksdesign` random number generator. Normally it is best to use the `NULL` seed setting which allows `blocksdesign` to find its own random seed.

The `blocks_model` output shows the overall achieved D and A-efficiency factors for each sequentially fitted blocks model. Efficiency factors are shown for a first-order and a second-order model for each sequentially added block factor. For a fully nested blocks design, the two models are eqivalent and the two sets of efficiency factors will be equal but for a general crossed blocks model the two sets of efficiency factors will be different and may provide some guidance on the best choice of weighting parameter for an efficient design.

The definition of efficiency used by the `design` algorithm is the ratio of the generalized variance of the full treatment design relative to the generalized variance of the optimized block and treatment design. Using this definition, it is possible for quantitative level treatment designs to have efficiency factors greater than one. Therefore the main use of efficiency factors is to compare the relative efficiencies of different optimizations of the same design.

Outputs:

The principle design outputs comprise:

- A data frame showing a randomized allocation of treatments to blocks.
- A table showing the fractional size of the treatment design and the D-efficiency factors of that fraction.
- A table showing the blocks sub-model design and the D- and A-efficiency factor of each successively fitted blocks sub-model.

Value

`design`	The design layout showing the allocation of treatment and block design factors to individual plots.
`treatments_model`	The fractional size of the treatment design together with the D-efficiency of that fraction.
`blocks_model`	The blocks sub-model design together with the D and A-efficiency factor of each successively fitted blocks sub-model.
`seed`	Numerical seed for random number generator.
`searches`	Maximum number of searches in each stratum.
`jumps`	Number of random treatment swaps to escape a local maxima.

References

Cochran W. G. & Cox G. M. (1957) Experimental Designs 2nd Edition John Wiley & Sons.

DURBAN, M., HACKETT, C., MCNICOL, J., NEWTON, A., THOMAS, W., & CURRIE, I. (2003). The practical use of semi-parametric models in field trials, Journal of Agric. Biological and Envir. Stats., 8, 48-66.

Examples

```
## For optimum results, the number of searches may need to be increased in practice.
## Designs can be rebuilt repeatedly to check that a near-optimum design has been found.

## 4 replicates of 12 treatments with 16 nested blocks of size 3
treatments = factor(1:12)
Blocks = factor(rep(1:4,each=12))
subBlocks = factor(rep(1:16,each=3))
blocks = data.frame(Blocks,subBlocks)
design(treatments,blocks)$blocks_model

## 4 x 12 design for 4 replicates of 12 treatments with 16 nested blocks of size 3
## only the default weighting (0.5) will ensure an optimal Trojan design
treatments = factor(1:12)
MainCols = factor(rep(rep(1:4,each=3),4))
MainRows = factor(rep(1:4,each=12))
Cols = factor(rep(1:12,4))
blocks = data.frame(MainRows,MainCols,Cols)
design(treatments,blocks,searches=200,weighting=0)$blocks_model
design(treatments,blocks,searches=200)$blocks_model
design(treatments,blocks,searches=200,weighting=1)$blocks_model

## 4 x 13 Row-and-column design for 4 replicates of 13 treatments
## Youden design Plan 13.5 Cochran and Cox (1957).
treatments=factor(1:13)
Rows =factor(rep(1:4,each=13))
Cols =factor(rep(1:13,4))
blocks =data.frame(Rows,Cols)
design(treatments,blocks,searches=700)

## Durban - 272 treatments in a 16 x 34 design with nested rows-and-columns
data(durban)
durban=durban[c(3,1,2,4,5)]
durban=durban[ do.call(order, durban), ]
treatments=data.frame(gen=durban$gen)
Reps = factor(rep(1:2,each=272))
Rows = factor(rep(1:16,each=34))
Col1 = factor(rep(rep(1:4,c(9,8,8,9)),16))
Col2 = factor(rep(rep(1:8,c(5,4,4,4,4,4,4,5)),16))
Col3 = factor(rep(1:34,16))
blocks = data.frame(Reps,Rows,Col1,Col2,Col3)
design(treatments,blocks,searches=10)$blocks_model
## Compare with efficiency factors of original design; Durban et al (2003)
blockEfficiencies(treatments,blocks)


## differential replication including single replicate treatments (13 to 24)
treatments=factor(c(rep(1:12,2), rep(13:24,1)))
Main=factor(rep(1:2,each=18))
Sub =factor(rep(1:6,each=6))
blocks =data.frame(Main,Sub)
```

```
design(treatments,blocks,searches=5)

## 48 treatments in 2 replicate blocks of size 4 x 12 with 2 main rows and 3 main columns
treatments=factor(1:48)
replicates=factor(rep(1:2,each=48))
rows=factor(rep(rep(1:2,each=24),2))
cols=factor(rep(rep(1:3,each=8),4))
blocks=data.frame(replicates,cols,rows)
design(treatments,blocks,searches=5)

## Factorial treatment designs defined by a treatments data frame and a factorial model equation.
## For some examples, a repeat loop with a break based on an efficiency factor criterion is used
## to search for a global optimal design from amongst a large number of local optimal designs

## Main effects of five 2-level factors in a half-fraction of a 2/2/2 nested blocks design
treatments = expand.grid(F1=factor(1:2),F2=factor(1:2),F3=factor(1:2),F4=factor(1:2),F5=factor(1:2))
blocks=data.frame(b1=factor(rep(1:2,each=8)),b2=factor(rep(1:4,each=4)),b3=factor(rep(1:8,each=2)))
treatments_model="F1 + F2 + F3 + F4 + F5"
repeat {
 z=design(treatments,blocks,treatments_model,searches=5)
if ( z$blocks_model[3,3]==1  ) break }
z

# Second-order model for five qualitative 2-level factors in 4 randomized blocks
treatments=expand.grid(F1=factor(1:2),F2=factor(1:2),F3=factor(1:2),F4=factor(1:2),F5=factor(1:2))
blocks=factor(rep(1:4,each=8))
treatments_model="(F1+F2+F3+F4+F5)^2"
design(treatments,blocks,treatments_model,searches=5)

# Main effects of five 2-level factors in a half-fraction of a 4 x 4 row-and column design.
treatments = expand.grid(F1=factor(1:2),F2=factor(1:2),F3=factor(1:2),F4=factor(1:2),
F5=factor(1:2))
blocks=data.frame( rows=factor(rep(1:4,each=4)), cols=factor(rep(1:4,4)))
treatments_model="~ F1+F2+F3+F4+F5"
design(treatments,blocks,treatments_model,searches=20)

# Quadratic regression for one 6-level numeric factor in 2 randomized
#  blocks assuming 10/6 fraction
treatments=expand.grid(X=1:6)
blocks=factor(rep(1:2,each=5))
treatments_model=" ~ poly(X,2)"
design(treatments,blocks,treatments_model,searches=5)

# First-order model for 1/3rd fraction of four qualitative 3-level factors in 3  blocks
treatments=expand.grid(F1=factor(1:3),F2=factor(1:3),F3=factor(1:3),F4=factor(1:3))
blocks=factor(rep(1:3,each=9))
treatments_model=" ~ F1+F2+F3+F4"
repeat {
z=design(treatments,blocks,treatments_model,searches=10)
if ( z$blocks_model[1,3]==1  ) break }
z

# Second-order model for a 1/3rd fraction of five qualitative 3-level factors in 3 blocks
```

```
treatments=expand.grid( F1=factor(1:3), F2=factor(1:3), F3=factor(1:3), F4=factor(1:3),
F5=factor(1:3))
blocks=factor(rep(1:3,each=27))
treatments_model=" ~ (F1+F2+F3+F4+F5)^2"
repeat {
z=design(treatments,blocks,treatments_model,searches=25)
if ( z$blocks_model[1,3]==1  ) break }
z

# Second-order model for two qualitative and two quantitative level factors in 4 randomized blocks
treatments=expand.grid(F1=factor(1:2),F2=factor(1:3),V1=1:3,V2=1:4)
blocks=factor(rep(1:4,each=18))
treatments_model = " ~ F1 + F2 + poly(V1,2) +  poly(V2,2) + (poly(V1,1)+F1+F2):(poly(V2,1)+F1+F2) "
 design(treatments,blocks,treatments_model,searches=5)

# Plackett and Burman design for eleven 2-level factors in 12 runs (needs large number of searches)
GF=expand.grid(F1=factor(1:2),F2=factor(1:2),F3=factor(1:2),F4=factor(1:2),F5=factor(1:2),
F6=factor(1:2),F7=factor(1:2),F8=factor(1:2),F9=factor(1:2),F10=factor(1:2),F11=factor(1:2))
blocks=factor(rep(1,12))
model=model="~ F1+F2+F3+F4+F5+F6+F7+F8+F9+F10+F11"
design(GF,blocks,model,searches=25)
```

---

durban *Durban example data design*

Description

Actual layout used by DURBAN, M., HACKETT, C., MCNICOL, J., NEWTON, A., THOMAS, W., & CURRIE, I. (2003). The practical use of semi-parametric models in field trials, Journal of Agric Biological and Envir Stats, 8, 48-66.

Usage

```
data(durban)
```

Format

An object of class `data.frame` with 544 rows and 5 columns.

upper_bounds *Efficiency bounds*

Description

Finds upper A-efficiency bounds for regular block designs.

Usage

```
upper_bounds(n, v, b)
```

Arguments

n	the total number of plots in the design.
v	the total number of treatments in the design.
b	the total number of blocks in the design.

Details

Upper bounds for the A-efficiency factors of regular block designs (see Chapter 2.8 of John and Williams 1995). Non-trivial A-efficiency upper bounds are calculated for regular block designs with equal block sizes and equal replication. All other designs return NA.

References

John, J. A. and Williams, E. R. (1995). Cyclic and Computer Generated Designs. Chapman and Hall, London.

Examples

```
# 50 plots, 10 treatments and 10 blocks for a design with 5 replicates and blocks of size 5
upper_bounds(n=50,v=10,b=10)
```

Index