Package 'Crossover'

March 14, 2023

Type Package

Title Analysis and Search of Crossover Designs

Version 0.1-21

Author Kornelius Rohmeyer

Maintainer Kornelius Rohmeyer <rohmeyer@small-projects.de>

Description Generate and analyse crossover designs from combinatorial or search algorithms as well as from literature and a GUI to access them.

Depends R (>= 3.0.2), ggplot2

Imports MASS, crossdes (>= 1.1-1), xtable, methods, Matrix, rJava (>= 0.8-3), CommonJavaJars (>= 1.0.5), Rcpp (>= 0.10.3), RcppArmadillo (>= 0.2.0), JavaGD, multcomp, stats4, digest

Suggests knitr, testthat, nlme

LinkingTo Rcpp (>= 0.10.3), RcppArmadillo (>= 0.2.0)

SystemRequirements Java (>= 5.0)

License GPL-2

VignetteBuilder knitr

URL https://github.com/kornl/Crossover/wiki

BugReports https://github.com/kornl/Crossover/issues

RoxygenNote 7.2.3

NeedsCompilation yes

Repository CRAN

Date/Publication 2023-03-14 14:30:05 UTC

R topics documented:

`Crossover-package` *This package provides more than two hundred cross-over design from literature, a search algorithm to find efficient cross-over designs for various models and a graphical user interface to find/generate appropriate designs.*

Description

This package provides more than two hundred cross-over design from literature, a search algorithm to find efficient cross-over designs for various models and a graphical user interface to find/generate appropriate designs.

Author(s)

Maintainer: Kornelius Rohmeyer <rohmeyer@small-projects.de>

References

Jones, B., & Kenward, M. G. (2003). Design and analysis of cross-over trials (Vol. 98). Chapman & Hall.

John, J. A., Russell, K. G., & Whitaker, D. (2004). CrossOver: an algorithm for the construction of efficient cross-over designs. Statistics in medicine, 23(17), 2645-2658.

`buildSummaryTable` *Build Summary Table For All Examples From Literature*

Description

Build Summary Table For All Examples From Literature

Usage

```
buildSummaryTable(extended = FALSE)
```

Arguments

`extended`	If `TRUE` the summary table will have further columns with extended information as how balanced the design is and whether all treatment effect differences are estimable under all models.

Details

See also the documentation for the data files.

Value

TODO

Author(s)

Kornelius Rohmeyer <rohmeyer@small-projects.de>

References

See the documentation for the data files.

Examples

```
buildSummaryTable()
```

`canonicalOrder` *Sorts sequences of a design into a canonical order*

Description

Sorts sequences of a design into a canonical order.

Usage

```
canonicalOrder(design)
```

Arguments

`design`	Cross-over design.

Details

When comparing bigger designs this ordering easily allows to check whether two designs are equal.

Author(s)

Kornelius Rohmeyer <rohmeyer@small-projects.de>

Examples

```
getDesign("switchback5t")
canonicalOrder(getDesign("switchback5t"))
```

`contrMat2` *Create the design matrix, variance-covariance matrix, the variance of each pairwise comparison and the efficicency of each pairwise comparison for a cross-over design*

Description

Function to read in a cross-over design and create the design matrix X, the variance of each pairwise comparison and the efficicency of each pairwise comparison.

Usage

```
contrMat2(
  type,
  v,
  model,
  eff.factor = rep(1, length(parameterCount(model, v)))
)
```

Arguments

type	Type of contrast. A character vector containing the following: "Dunnett", "Tukey", "none". If the length is 1, this contrast is only applied for the treatment effects and for carry-over effects a "Tukey" contrast is used. Otherwise the specified contrasts are used, see also the examples.
v	Number of treatments
model	Model - one of the following: 1) "Standard additive model", 2) "Second-order carry-over effects", 3) "Full set of interactions", 4) "Self-adjacency model", 5) "Placebo model", 6) "No carry-over into self model", 7) "Treatment decay model", 8) "Proportionality model", 9) "No carry-over effects". Can be specified as number or as character string.
eff.factor	Weight applied to the different sub contrast matrices. A warning is given if it does not sum up to one. See examples.

Details

See the vignette of this package for further details.

Value

A contrast matrix

Author(s)

Kornelius Rohmeyer <rohmeyer@small-projects.de>

Examples

```
contrMat2("Tukey", v=3, model=1)
contrMat2("Dunnett", v=3, model=1)
contrMat2(c("Dunnett", "Dunnett"), v=3, model=1)
contrMat2(c("Dunnett", "none"), v=3, model=1)
contrMat2(c("Dunnett", "none", "none"), v=3, model=8)
contrMat2("Dunnett", v=3, model=1, eff.factor=c(0.9, 0.1))
contrMat2("Dunnett", v=3, model=8, eff.factor=c(0.5, 0.3, 0.2))
```

CrossoverDesign-class *Class CrossoverDesign*

Description

A S4 class for Crossover designs: CrossoverDesign

Slots

list("design") Matrix specifying the design. Rows represent periods and columns the subjects.

list("s") Number of sequences.

list("p") Number of periods.

list("v") Number of treatments.

list("model") A numeric specifying the model the design was searched for or -1 if unknown.

list("description") Optional description of design or reference.

list("attr") List with attributes.

list("misc") List with miscellaneous stuff - not used yet.

Author(s)

Kornelius Rohmeyer <rohmeyer@small-projects.de>

Examples

```
design <- t(rbind(c(1,1,2,2),
                  c(2,2,1,1),
                  c(1,1,2,2),
                  c(2,2,1,1),
                  c(1,2,2,1),
                  c(2,1,1,2)))

new("CrossoverDesign", design)
```

Crossoverdesigns *Selected Cross-Over designs from literature*

Description

Selected Cross-Over designs from literature.

You can access all designs via the function `getDesign` as in the example `getDesign("williams4t")`.

Format

A integer matrix specifying the design. Rows represent periods and columns the subjects.

Details

These data sets are stored combined by prefix, so alternatively to using the recommended function `getDesign` you coud access for example design `fletcher10` by using the command `data(fletcher10)` and afterwards all 31 design from fletcher1 up to fletcher31 are loaded.

The available data sets are:

federerAtkinson3ta, federerAtkinson3tb, federerAtkinson4ta, federerAtkinson4tb, federerAtkinson5ta, fletcher1, fletcher10, fletcher11, fletcher12, fletcher13, fletcher14, fletcher15, fletcher16, fletcher17, fletcher18, fletcher19, fletcher2, fletcher20, fletcher21, fletcher22, fletcher23, fletcher24, fletcher25, fletcher26, fletcher27, fletcher28, fletcher29, fletcher3, fletcher30, fletcher31, fletcher4, fletcher5, fletcher6, fletcher7, fletcher8, fletcher9, iqbalJones1, iqbalJones10, iqbalJones11, iqbalJones12, iqbalJones13, iqbalJones14, iqbalJones15, iqbalJones16, iqbalJones17, iqbalJones18, iqbalJones19, iqbalJones2, iqbalJones20, iqbalJones21, iqbalJones22, iqbalJones23, iqbalJones24, iqbalJones25, iqbalJones26, iqbalJones27, iqbalJones28, iqbalJones29, iqbalJones3, iqbalJones30, iqbalJones31, iqbalJones32, iqbalJones33, iqbalJones34, iqbalJones35, iqbalJones36, iqbalJones37, iqbalJones38, iqbalJones39, iqbalJones4, iqbalJones40, iqbalJones41, iqbalJones42, iqbalJones5, iqbalJones6, iqbalJones7, iqbalJones8, iqbalJones9, lewisFletcherMatthews1, lewisFletcherMatthews10, lewisFletcherMatthews11, lewisFletcherMatthews12, lewisFletcherMatthews13, lewisFletcherMatthews14, lewisFletcherMatthews15, lewisFletcherMatthews16, lewisFletcherMatthews17, lewisFletcherMatthews18, lewisFletcherMatthews19, lewisFletcherMatthews2, lewisFletcherMatthews20, lewisFletcherMatthews3, lewisFletcherMatthews4, lewisFletcherMatthews5, lewisFletcherMatthews6, lewisFletcherMatthews7, lewisFletcherMatthews8, lewisFletcherMatthews9, orthogonalLatinSquare3t, orthogonalLatinSquare4t, orthogonalLatinSquare5t, orthogonalLatinSquare7t, pattersonLucasExtraPeriod30, pattersonLucasExtraPeriod31, pattersonLucasExtraPeriod32, pattersonLucasExtraPeriod33, pattersonLucasExtraPeriod34, pattersonLucasExtraPeriod35, pattersonLucasExtraPeriod36, pattersonLucasExtraPeriod37, pattersonLucasExtraPeriod38, pattersonLucasExtraPeriod39, pattersonLucasExtraPeriod40, pattersonLucasExtraPeriod41, pattersonLucasExtraPeriod42, pattersonLucasExtraPeriod43, pattersonLucasExtraPeriod44, pattersonLucasExtraPeriod45, pattersonLucasExtraPeriod46, pattersonLucasExtraPeriod47, pattersonLucasExtraPeriod48, pattersonLucasExtraPeriod49, pattersonLucasExtraPeriod86, pattersonLucasPBIBD100, pattersonLucasPBIBD101, pattersonLucasPBIBD102, pattersonLucasPBIBD103, pattersonLucasPBIBD104, pattersonLucasPBIBD105, pattersonLucasPBIBD106, pattersonLucasPBIBD107, pattersonLucasPBIBD125, pattersonLucasPBIBD126, pattersonLucasPBIBD127, pattersonLucasPBIBD128, pattersonLucasPBIBD131, pattersonLucasPBIBD132, pattersonLucasPBIBD133, pattersonLucasPBIBD134, pattersonLucasPBIBD135, pattersonLucasPBIBD136, pattersonLucasPBIBD137, pattersonLucasPBIBD138, pattersonLucasPBIBD139, pattersonLucasPBIBD140, pattersonLucasPBIBD141, pattersonLucasPBIBD153, pattersonLucasPBIBD154, pattersonLucasPBIBD155, pattersonLucasPBIBD156, pattersonLucasPBIBD99, pattersonLucasPltT1, pattersonLucasPltT10, pattersonLucasPltT12, pattersonLucasPltT13, pattersonLucasPltT15, pattersonLucasPltT16, pattersonLucasPltT17, pattersonLucasPltT18, pattersonLucasPltT19, pattersonLucasPltT20, pattersonLucasPltT21, pattersonLucasPltT22, pattersonLucasPltT23, pattersonLucasPltT3, pattersonLucasPltT4, pattersonLucasPltT5, pattersonLucasPltT7, pattersonLucasPltT8, pattersonLucasPltT9, pidgeon1, pidgeon10, pidgeon11, pidgeon12, pidgeon13, pidgeon14, pidgeon15, pidgeon16, pidgeon17, pidgeon18, pidgeon19, pidgeon2, pidgeon20, pidgeon3, pidgeon4, pidgeon5, pidgeon6, pidgeon7, pidgeon8, pidgeon9, prescott1, prescott2, quenouille3t1, quenouille3t2, quenouille4t1, quenouille4t2, quenouille4t3, russel4t, russel7t, switchback3t, switchback4t, switchback5t, switchback6t, switchback7t, williams3t, williams4t, williams5t, williams6t, williams7t, williams8t, williams9t, pb2.64.

Source

Anderson, I. and Preece, D.A. (2002) Locally balanced change-over designs. Utilitas Mathematica, To appear.

Anderson, I. (2002) Personal communication.

Archdeacon, D.S., Dinitz J.H., Stinson, D.R. and Tillson, T.W. (1980) Some new row-complete latin squares, Journal of Combinatorial Theory, Series A, 29, 395-398.

Atkinson, G.F. (1966) Designs for sequences of treatments with carry-over effects. Biometrics, 22, 292–309.

Balaam, L.N. (1968) A two-period design with t^2 experimental units. Biometrics, 24, 61–73.

Bate, S. and Jones, B. (2002) The construction of universally optimal uniform cross-over designs. GlaxoSmithKline Biomedical Data Sciences Technical Report 2002-06.

Berenblut, I.I. (1964) Change-over designs with complete balance for residual effects. Biometrics, 23, 578–580.

Blaisdell, E.A. and Raghavarao, D. (1980) Partially balanced change-over designs based on m-associate class PBIB designs. Journal of the Royal Statistical Society, B, 42, 334–338.

Clatworthy, W. H. (1973). Tables of two-associate-class partially balanced designs. US Government Printing Office.

Davis, A.W. and Hall, W.B. (1969) Cyclic change-over designs. Biometrika, 56, 283–293.

Federer, W.T. and Atkinson, G.F. (1964) Tied-double-change-over designs. Biometrics, 20, 168–181.

Fletcher, D.J. (1987) A new class of change-over designs for factorial experiments. Biometrika, 74, 649–654.

Iqbal, I. and Jones, B. (1994) Efficient repeated measurements designs with equal and unequal period sizes. Journal of Statistical Planning and Inference, 42, 79-88.

Factorial cross-over designs in clinical trials, Lewis, S.M., Fletcher, D.J. and Matthews, J.N.S. In Optimal Design and Analysis of Experiments, Editors, Dodge, Y., Fedorov, V.V. and Wynn, H.P. (1988), 133–140, Elsevier Science Publishers B.V. (North-Holland).

Cochran, W.G., Autrey, K.M. and Cannon, C.Y. (1941) A double change-over design for dairy cattle feeding experiments. Journal of Dairy Science, 24, 937–951

Patterson, H.D. and Lucas, H.L. (1962) Change-over designs. North Carolina Agricultural Experiment Station. Tech. Bull. No. 147.

Residual effects designs for comparing treatments with a control. PhD dissertation, Temple University, Phildelphia, PA, 1984.

Prescott, P. (1994) Construction of sequentially counterbalanced designs formed from two or more Latin Squares. Proceedings of the 11th Symposium on Computational Statistics held in Vienna, Austria. Editors Dutter, R. and Grossmann, W., Physica-Verlag: Heidelberg, 435-440.

Prescott, P. (1999) Construction of sequentially counterbalanced designs formed from two latin squares. Utilitas Mathematica, 55, 135–152.

Quenouille, M.H. (1953) The Design and Analysis of Experiments. Griffin, London.

Russell, K.R. (1991) The construction of good change-over designs when there are fewer units than treatments. Biometrika, 78, 305-313.

Lucas, H.L. (1956) Switch-back trials for more than two treatments. Journal of Diary Science, 39, 146–154.

Williams, E.J. (1949) Experimental designs balanced for the estimation of residual effects of treatments. Australian Journal of Science Res(A), 2, 14900168.

Examples

```
getDesign("williams4t")

data(fletcher)
ls(pattern="fletcher*")
fletcher10
```

`CrossoverGUI` *Graphical User Interface for Crossover Designs*

Description

Starts a graphical user interface for accessing and creating crossover designs.

Usage

```
CrossoverGUI()
```

Details

See the vignette of this package for further details, since describing a GUI interface is better done with some nice pictures.

Value

The function itself returns nothing of interest. But from the GUI designs and objects can be created or edited that will be available in R under the specified variable name after saving.

Author(s)

Kornelius Rohmeyer <rohmeyer@small-projects.de>

Examples

```
## Not run:
CrossoverGUI()

## End(Not run)
```

CrossoverSearchResult-class *Class CrossoverSearchResult*

Description

A S4 class for the search result for Crossover designs: CrossoverSearchResult

Slots

list("design") An object of class `CrossoverDesign` describing the best design that was found.

list("startDesigns") A list of start designs to search from.

list("model") A numeric specifying the model the design was searched for or -1 if unknown.

list("eff") List, Progress of the algorithm. TODO: Explain further.

list("search") List, TODO

list("time") Named numeric with the time in seconds the algorithm was searching.

list("misc") List - in the moment not used.

Author(s)

Kornelius Rohmeyer <rohmeyer@small-projects.de>

Examples

```
# n=c(100,10) is very small, but it's just an example and should not take much time
x <- searchCrossOverDesign(s=9, p=5, v=4, model=4, n=c(100,10))
print(x)
```

design.efficiency *Create the design matrix, variance-covariance matrix, the variance of each pairwise comparison and the efficicency of each pairwise comparison for a cross-over design*

Description

Function to read in a cross-over design and create the design matrix X, the variance of each pairwise comparison and the efficicency of each pairwise comparison.

Usage

```
design.efficiency(
  design,
  model = 1,
  model.param = list(),
  v = length(levels(as.factor(design)))
)
```

Arguments

design	Cross-over design.
model	Model - one of the following: 1) "Standard additive model", 2) "Second-order carry-over effects", 3) "Full set of interactions", 4) "Self-adjacency model", 5) "Placebo model", 6) "No carry-over into self model", 7) "Treatment decay model", 8) "Proportionality model", 9) "No carry-over effects".
model.param	List of additional model specific parameters. In the moment these are ppp, the proportionality parameter for the proportionality model, and placebos, the number of placebo treatments in the placebo model.
v	Number of treatments

Details

See the vignette of this package for further details.

Value

A list with the following elements:

- xmat Design matrix for the given model (including subject and period effects)
- var.trt.pair.adj Matrix of treament difference variances
- eff.trt.pair.adj Matrix of treament difference efficiencies

Author(s)

Kornelius Rohmeyer <rohmeyer@small-projects.de>

References

Jones, B., & Kenward, M. G. (2003). Design and analysis of cross-over trials (Vol. 98). Chapman & Hall.

Examples

```
design.efficiency(getDesign("fletcher1"))
design.efficiency(getDesign("fletcher1"), model=7)
design.efficiency(getDesign("switchback4t"), model=7)
```

exampleSearchResults2t

Example search results for two treatments

Description

A list of 16 integer matrices specifying the design. Rows represent periods and columns the subjects.

Format

A list of 16 integer matrices specifying the design. Rows represent periods and columns the subjects.

Details

See vignette.

Source

Found by method searchCrossOverDesign.

general.carryover

Calculate variances of paramater contrasts

Description

Calculate variances of paramater contrasts

Usage

```
general.carryover(
  design,
  v = length(table(design)),
  model,
  ppp = 0.5,
  placebos = 1,
  contrasts
)
```

Arguments

design	Cross-over design.
v	Number of treatments
model	Model - one of the following numbers or Strings: 1 = "Standard additive model", 2 = "Self-adjacency model", 3 = "Proportionality model", 4 = "Placebo model", 5 = "No carry-over into self model", 6 = "Treatment decay model", 7 = "Full set of interactions", 8 = "Second-order carry-over effects"

ppp	The proportionality parameter for the proportionality model.
placebos	The number of placebo treatments in the placebo model.
contrasts	Optionally a contrast matrix or a list of contrast matrix. If missing pairwise differences for treatment and carry-over parameters are calculated.

Details

See the vignette of this package for further details.

Value

A list with the variances of the pairwise differences or specified contrasts. If contrasts are not estimable, NA is returned for variances.

Author(s)

Kornelius Rohmeyer <rohmeyer@small-projects.de>

References

Jones, B., & Kenward, M. G. (2003). Design and analysis of cross-over trials (Vol. 98). Chapman & Hall.

Examples

```
general.carryover(getDesign("fletcher1"), model=1)
general.carryover(getDesign("fletcher1"), model=2)
general.carryover(getDesign("fletcher1"), model=3)
general.carryover(getDesign("switchback4t"), model=7)
```

---

getDesign *Extract Design from a CrossoverSearchResult*

Description

Extract Design from a CrossoverSearchResult

Usage

```
## S4 method for signature 'CrossoverSearchResult'
getDesign(object, ...)
```

Arguments

object	A searchCrossOverDesign object from which the design should be extracted.
...	Possible parameters for subclasses (not yet used).

Value

Returns a numeric matrix representing the crossover design. Rows represent periods, columns represent sequences.

Author(s)

Kornelius Rohmeyer <rohmeyer@small-projects.de>

Examples

```
# n=c(100,10) is very small, but it's just an example and should not take much time
x <- searchCrossOverDesign(s=9, p=5, v=4, model=4, n=c(100,10))
getDesign(x)

getDesign("williams4t")
```

---

getModelNr *Get the number or character string specifying the model*

Description

Get the number or character string specifying the model

Usage

```
getModelNr(model, type = "numeric")
```

Arguments

model	Number or character string specifying the model
type	Eiher "numeric" or "character". If numeric the number of the model will be returned. Otherwise the character string description of the model.

Value

Either number or character string specifying the model.

Examples

```
Crossover:::getModelNr("Self-adjacency model")==Crossover:::getModelNr(2)
"Self-adjacency model"==Crossover:::getModelNr(2, type="character")
Crossover:::getModelNr("Self-adjacency model")==2
```

plot *Plots information about the search algorithm and its process.*

Description

Plots information about the search algorithm and its process.

Usage

```
## S4 method for signature 'CrossoverSearchResult,missing'
plot(x, y, type = 1, show.jumps = FALSE)
```

Arguments

x	Result from searchCrossOverDesign.
y	Missing.
type	Type of plot. Number 1 is more colorful, but number 2 perhaps a bit easier to understand.
show.jumps	If TRUE vertical lines will show where the specified jumps occured.

Details

The x-axis corresponds to the consecutive simulation runs and the y-axis to the design criterion E that depending on the model is either a weighted average of efficiency factors or standardized pairwise variances and described in detail in the vignette of this package. Also see the vignette for a few examples and a discussion what can be derived from this plots.

Value

Returns a ggplot object of the plot.

Author(s)

Kornelius Rohmeyer <rohmeyer@small-projects.de>

Examples

```
## Not run:
x <- searchCrossOverDesign(s=9, p=5, v=4, model=4)
plot(x)

## End(Not run)

x <- searchCrossOverDesign(s=9, p=5, v=4, model=4, n=c(50,10), jumps=c(10, 10))
plot(x, show.jumps=TRUE)
plot(x, type=2)
```

rcd *Create a row column design*

Description

Create a row column design

Usage

```
rcd(X, v, model)
```

Arguments

X	cross-over design
v	number of treatments
model	String or number describing the model. See getModelNr.

Value

A row-column design (as matrix - but not the design matrix).

See Also

rcdMatrix gives the row-column design matrix.

Examples

```
# TODO
```

rcdMatrix *Create the design matrix for a given row column design*

Description

Create the design matrix for a given row column design

Usage

```
rcdMatrix(X, v, model)
```

Arguments

X	row-column design
v	number of treatments
model	String or number describing the model. See getModelNr.

Value

The design matrix for a row-column design.

See Also

rcd gives the row-column design to a given crossover design.

Examples

```
# TODO
```

`searchCrossOverDesign` *Search for a Cross-Over Design*

Description

Search for a Cross-Over Design

Usage

```
searchCrossOverDesign(
  s,
  p,
  v,
  model = "Standard additive model",
  eff.factor = 1,
  v.rep,
  balance.s = FALSE,
  balance.p = FALSE,
  verbose = 0,
  model.param = list(),
  n = c(5000, 20),
  jumps = c(5, 50),
  start.designs,
  random.subject = FALSE,
  contrast,
  correlation = NULL,
  rho = 0
)
```

Arguments

s	Number of sequences.
p	Number of periods.
v	Number of treatments.

`model`	Model - one of the following: "Standard additive model" (2), "Second-order carry-over effects" (3), "Full set of interactions" (3), "Self-adjacency model" (3), "Placebo model" (2), "No carry-over into self model" (2), "Treatment decay model" (2), "Proportionality model" (1), "No carry-over effects" (0). The number in parentheses is the number of different efficiency factors that can be specified.
`eff.factor`	Weights for different efficiency factors. (Not used in the moment.)
`v.rep`	Integer vector specifying how often each treatment should be assigned (sum must equal s*p).
`balance.s`	Boolean specifying whether to allocate the treatments as equally as possible to each sequence (can result in loss of efficiency).
`balance.p`	Boolean specifying whether to allocate the treatments as equally as possible to each period (can result in loss of efficiency).
`verbose`	Level of verbosity, a number between 0 and 10. The default `verbose=0` does not print any output, while `verbose=10` prints any available notes.
`model.param`	List of additional model specific parameters. In the moment these are `ppp`, the proportionality parameter for the proportionality model, and `placebos`, the number of placebo treatments in the placebo model.
`n`	`n=c(n1,n2)` with *n1* the number of hill climbing steps per trial and *n2* the number of searches from random start matrices.
`jumps`	To reduze the possibility of the hill-climbing algorithm to get stuck in local extrema long jumps of distance *d* can be performed all *k* steps. This can be specified as `long.jumps=c(d,k)`. If *long.jumps* has only length 1 the default for *k* is 50. If after *k/2* hill-climbing steps the old design criterion is not enhanced (or at least reached), the algorithm returns to the design from before the jump.
`start.designs`	A single design or a list of start designs. If missing or to few start designs are specified (with regard to parameter *n* which specifies a number of 20 start designs as default) the start designs are generated randomly with the sample function. Alternatively `start.designs="catalog"` can be used to take start designs from the catalog to which random designs are added till *n2* start designs are at hand.
`random.subject`	Should the subject effects be random (`random.subject=TRUE`) or fixed effects (`random.subject=FALSE`).
`contrast`	Contrast matrix to be optimised. TODO: Example and better explanation for contrast.
`correlation`	Either a correlation matrix for the random subject effects or one of the following character strings: "equicorrelated", "autoregressive"
`rho`	Parameter for the correlation if parameter `correlation` is a character string.

Details

See the vignette of this package for further details.

Value

Returns the design as an integer matrix.

Author(s)

Kornelius Rohmeyer <rohmeyer@small-projects.de>

References

John, J. A., Russell, K. G., & Whitaker, D. (2004). CrossOver: an algorithm for the construction of efficient cross-over designs. Statistics in medicine, 23(17), 2645-2658.

Examples

```
## Not run:
x <- searchCrossOverDesign(s=9, p=5, v=4, model=4)

jumps <- c(10000, 200) # Do a long jump (10000 changes) every 200 steps
n <- c(1000, 5)        # Do 5 trials with 1000 steps in each trial
result <- searchCrossOverDesign(s=9, p=5, v=4, model=4, jumps=jumps, n=n)
plot(result)

## End(Not run)
```

Index