

Package ‘randomizeR’

June 15, 2018

Title Randomization for Clinical Trials

Version 1.4.2

Date 2018-06-14

Description This tool enables the user to choose a randomization procedure based on sound scientific criteria. It comprises the generation of randomization sequences as well the assessment of randomization procedures based on carefully selected criteria. Furthermore, ‘randomizeR’ provides a function for the comparison of randomization procedures.

Depends R (>= 3.3.0), methods, ggplot2, plotrix

License GPL (>= 3)

LazyData true

Collate ‘getDesign.R’ ‘randPar.R’ ‘abcdPar.R’ ‘randSeq.R’ ‘abcdSeq.R’
‘getExpectation.R’ ‘normEndp.R’ ‘endpoint.R’ ‘util.R’
‘getStat.R’ ‘power.R’ ‘imbalance.R’ ‘corGuess.R’ ‘doublyF.R’
‘testDec.R’ ‘doublyT.R’ ‘chronBias.R’ ‘selBias.R’ ‘bias.R’
‘issue.R’ ‘assess.R’ ‘bbcdPar.R’ ‘bbcdSeq.R’ ‘ebcPar.R’
‘bsdPar.R’ ‘bsdSeq.R’ ‘chenPar.R’ ‘chenSeq.R’
‘chronBiasStepT.R’ ‘combinedBias.R’ ‘compare.R’ ‘crPar.R’
‘crSeq.R’ ‘createParam.R’ ‘derFunc.R’ ‘desFunc.R’
‘getDesFunc.R’ ‘derringerLs.R’ ‘derringerRs.R’ ‘derringerTs.R’
‘desScores.R’ ‘desirability.R’ ‘ebcSeq.R’ ‘evaluate.R’
‘gbcdPar.R’ ‘gbcdSeq.R’ ‘hadaPar.R’ ‘hadaSeq.R’ ‘mpPar.R’
‘mpSeq.R’ ‘pbrPar.R’ ‘pbrSeq.R’ ‘probUnDes.R’ ‘rtbdSeq.R’
‘rpbrSeq.R’ ‘randomBlockSeq.R’ ‘randomizeROverview.R’
‘randomizeRPackage.R’ ‘rarPar.R’ ‘rarSeq.R’ ‘rpbrPar.R’
‘tbdPar.R’ ‘rtbdPar.R’ ‘saveAssess.R’ ‘saveRand.R’ ‘tbdSeq.R’
‘udPar.R’ ‘udSeq.R’

Suggests testthat, knitr

VignetteBuilder knitr

RoxygenNote 6.0.1

NeedsCompilation no

Author David Schindler [aut],
 Diane Uschner [aut, cre],
 Martin Manolov [ctb],
 Thi Mui Pham [ctb],
 Ralf-Dieter Hilgers [aut, ths],
 Nicole Heussen [aut, ths]

Maintainer Diane Uschner <duschner@ukaachen.de>

Repository CRAN

Date/Publication 2018-06-15 16:04:21 UTC

R topics documented:

randomizeR-package	4
a	5
abcdPar	5
abcdRand	6
assess	7
bbcdPar	8
bbcdRand	9
blocks	10
bsdPar	10
bsdRand	11
chenPar	12
chenRand	13
chronBias	13
coin	15
combineBias	15
compare	16
corGuess	17
createParam	18
createSeq	19
crPar	19
derFunc	20
derFunc-class	21
desirability	21
doublyF_opt	22
doublyF_value	23
doublyF_values	24
doublyT	24
doublyTValues	25
ebcPar	25
evaluate	26
gbcdPar	28
gbcdRand	29
generateAllSequences	29
generateRandomSequences	32
genNcps	38

getbiasCS1	39
getbiasCS2	39
getCorGuesses	40
getDesFunc	40
getDesign	41
getDesScores	43
getExpectation	44
getProbabilities	46
getRandomizationList	47
hadaPar	48
hasAllGroups	49
hatMatrix	49
imbal	50
issue	51
K	52
lambda1	52
lambda2	53
makeBiasedExpectation	53
makeDesignMatrix	54
method	54
mpPar	55
mti	56
mu	56
N	56
normEndp	57
overview	57
pbrPar	59
plot	60
plotDes	61
plotEv	62
plotSeq	63
probUnDes	64
randBlocks	65
randPar	65
randPar-class	66
randSeq-class	66
rarPar	67
ratio	68
rho	68
rpbrPar	68
rtbdPar	69
saveAssess	70
saveRand	71
seed	71
selBias	72
setPower	73
sigma	74
summary	74

tbdPar	75
TV	76
type	76
udPar	77
\$.assessment-method	78
\$.comparison-method	78
\$.derFunc-method	79
\$.desFunc-method	79
\$.desScores-method	80
\$.endpoint-method	80
\$.evaluation-method	81
\$.issue-method	81
\$.probUnDesirable-method	82
\$.randPar-method	82
\$.randSeq-method	83

Index	84
--------------	-----------

randomizeR-package	<i>Randomization for Clinical Trials</i>
--------------------	--

Description

This tool enables the user to choose a randomization procedure based on sound scientific criteria. It comprises the generation of randomization sequences as well the assessment of randomization procedures based on carefully selected criteria. Furthermore, randomizeR provides a function for the comparison of randomization procedures.

Acknowledgement

This research is embedded in the [IDeAI project](#), which has received funding from the European Union's Seventh Framework Programme for research, technological development and demonstration under Grant Agreement no 602552.

Author(s)

David Schindler <dschindler@ukaachen.de>, Diane Uschner <duschner@ukaachen.de>, Ralf-Dieter Hilgers, Nicole Heussen

References

- W. F. Rosenberger and J. M. Lachin (2012) *Randomization in Clinical Trials - Theory and Practice*. Wiley.
- D. Schindler (2016) *Assessment of Randomization Procedures in the Presence of Selection and Chronological Bias*. PhD Thesis.
- Uschner D, Schindler D, Hilgers R and Heussen N (2018). "randomizeR: An R Package for the Assessment and Implementation of Randomization in Clinical Trials." *Journal of Statistical Software*, **85**(8), pp. 1-22. doi: 10.18637/jss.v085.i08 (URL: <http://doi.org/10.18637/jss.v085.i08>).

See Also

For functionality for randomization procedures, see [randPar](#) and [genSeq](#). For the criteria for the assessment of randomization procedures, see [issues](#). For the assessment and comparison of randomization procedures, see [assess](#) and [compare](#).

a *Function returning the adjusting parameter a slot of an S4 object*

Description

Function returning the adjusting parameter a slot of an S4 object

Usage

```
a(obj)
```

Arguments

obj object of class randPar

abcdPar *Representing Accelerated Biased Coin Design*

Description

Represents the randomization procedure Accelerated Biased Coin Design.

Usage

```
abcdPar(N, a, groups = LETTERS[1:2])
```

Arguments

N integer for the total sample size of the trial.

a nonnegative parameter which may be adjusted according to how strongly it is desired to balance the experiment. a = 0 gives the complete randomization, while the assignments become more deterministic as a increases.

groups character vector of labels for the different treatments.

Details

This is a class of 'biased coins' where the probability of selecting the under-represented treatment is dependent from the absolute difference between the two treatment allocations up to the current step.

Value

S4 object of the class abcdPar.

References

A. B. Antognini and A. Giovagnoli (2004) A new 'biased coin design' for the sequential allocation of two treatments. *Journal of the Royal Statistical Society. Series C (Applied Statistics)* **53**, No. 4, 651-664

See Also

Other randomization procedures: [bbcdPar](#), [bsdPar](#), [chenPar](#), [crPar](#), [createParam](#), [ebcPar](#), [gbcdPar](#), [hadaPar](#), [mpPar](#), [pbrPar](#), [rarPar](#), [rpbrPar](#), [rtbdPar](#), [tbdPar](#), [udPar](#)

abcdRand

Sampling algorithm for abcd

Description

Sampling algorithm for abcd

Usage

```
abcdRand(N, a, K = 2)
```

Arguments

N	integer for the total sample size of the trial.
a	nonnegative parameter which may be adjusted according to how strongly it is desired to balance the experiment. $a = 0$ gives the complete randomization, while the assignments become more deterministic as a increases.
K	number of treatment groups (e.g. $K=2$ if we compare one experimental against one control treatment).

Value

A vector with the allocation sequence for a clinical trial. It will contain a zero (resp. 1) at position i , when patient i is allocated to treatment A (resp. B).

References

A. B. Antognini and A. Giovagnoli (2004) A new 'biased coin design' for the sequential allocation of two treatments. *Journal of the Royal Statistical Society. Series C (Applied Statistics)* **53**, No. 4, 651-664

`assess`*Assessing randomization sequences*

Description

Assesses randomization sequences based on specified issues in clinical trials.

Usage

```
assess(randSeq, ..., endp)

## S4 method for signature 'randSeq,missing'
assess(randSeq, ..., endp)

## S4 method for signature 'randSeq,endpoint'
assess(randSeq, ..., endp)
```

Arguments

<code>randSeq</code>	object of class <code>randSeq</code> .
<code>...</code>	at least one object of class <code>issue</code> or just a list of objects of the class <code>issue</code> .
<code>endp</code>	object of class <code>endpoint</code> , or <code>missing</code> .

Details

Randomization sequences behave differently with respect to issues like selection bias, chronological bias, or loss in power estimation. The `assess` function evaluates the behaviour of randomization sequences with respect to these issues. The first argument should be a result of one of the functions [genSeq](#) or [getAllSeq](#). The second argument should be any number of [issues](#) arising in a clinical trial. The last argument `endp` may be provided if the assessment should take the distribution of the treatment groups into account, e.g. for power evaluation.

Value

S4 object of class `assessment` summarizing the assessment of the randomization procedure.

See Also

Representation of randomization procedures: [randPar](#)

Generation of randomization sequences: [genSeq](#)

[issues](#) for the assessment of randomization sequences

Examples

```
# assess the full set of Random Allocation Rule for N=4 patients
sequences <- getAllSeq(rarPar(4))
issue1 <- corGuess("CS")
issue2 <- corGuess("DS")
issue3 <- imbal("imb")
issue4 <- imbal("maxImb")
assess(sequences, issue1, issue2, issue3, issue4)

# assess one sequence of the Big Stick Design with respect to correct guesses
sequence <- genSeq(bsdPar(10, 2), seed = 1909)
assess(sequence, issue1)

# assess the same sequence with respect to selection bias
endp <- normEndp(c(2, 2), c(1, 1))
issue5 <- selBias("CS", 4, "exact")
issue6 <- setPower(2, "exact")
assess(sequence, issue1, issue5, issue6, endp = endp)

# recommended plot for the assessment of rejection probabilities
RP <- getAllSeq(crPar(6))
cB <- chronBias(type = "linT", theta = 1/6, method = "exact")
sB <- selBias(type = "CS", eta = 1/4, method = "exact")
normEndp <- normEndp(c(0, 0), c(1, 1))
A <- assess(RP, cB, sB, endp = normEndp)
D <- A$D
desiredSeq <- round(sum(D[,2][D[,3] <= 0.05 & D[,4] <= 0.05]), digits = 4)
colnames(D) <- c("Seq", "Prob", "SB", "linT")
g <- ggplot(D, aes(x = SB, y = linT))
g <- g + annotate("rect", xmin = 0, xmax = 0.05, ymin = 0, ymax = 0.05,
alpha=0.2, fill="green")
g <- g + geom_point(alpha = 1/10, size = 3, col = "orange")
g <- g + geom_vline(xintercept = 0.05, col = "red")
g <- g + geom_hline(yintercept = 0.05, col = "red")
g <- g + geom_text(data = NULL, x = 0, y = 0,
label = paste("Proportion:", desiredSeq), hjust=0, vjust=0, size = 7)
g
```

bbcdPar

Representing Bayesian Biased Coin Design

Description

Represents the randomization procedure Bayesian Biased Coin Design.

Usage

```
bbcdPar(N, a, groups = LETTERS[1:2])
```


Arguments

N	integer for the total sample size of the trial.
a	nonnegative parameter which may be adjusted according to how strongly it is desired to balance the experiment. $a = 0$ gives the complete randomization, while the assignments become more deterministic as a increases.
groups	character vector of labels for the different treatments.

Details

Extension of Efron's biased coin design.

Value

S4 object of the class `bbcdPar`.

References

A. B. Antognini and Maroussa Zagoraiou (2014) Balance and randomness in sequential clinical trials: the dominant biased coin design. *Pharmaceutical Statistics* **13(2)**, 119-127

See Also

Other randomization procedures: [abcdPar](#), [bsdPar](#), [chenPar](#), [crPar](#), [createParam](#), [ebcPar](#), [gbcdPar](#), [hadaPar](#), [mpPar](#), [pbrPar](#), [rarPar](#), [rpbrPar](#), [rtbdPar](#), [tbdPar](#), [udPar](#)

bbcdRand

Sampling algorithm for bbcd

Description

Sampling algorithm for bbcd

Usage

```
bbcdRand(N, a, K = 2)
```

Arguments

N	integer for the total sample size of the trial.
a	nonnegative parameter which may be adjusted according to how strongly it is desired to balance the experiment. $a = 0$ gives the complete randomization, while the assignments become more deterministic as a increases.
K	number of treatment groups (e.g. $K=2$ if we compare one experimental against one control treatment).

Value

A vector with the allocation sequence for a clinical trial. It will contain a zero (resp. 1) at position i , when patient i is allocated to treatment A (resp. B).

References

A. B. Antognini and Maroussa Zagoraiou (2014) Balance and randomness in sequential clinical trials: the dominant biased coin design. *Pharmaceutical Statistics* **13(2)**, 119-127

blocks	<i>Function returning the block slot of an S4 object</i>
--------	--

Description

Function returning the block slot of an S4 object

Usage

```
blocks(obj)
```

Arguments

obj	object of class pbrPAR
-----	------------------------

bsdPar	<i>Representing Big Stick Design</i>
--------	--------------------------------------

Description

Represents the randomization procedure Big Stick Design.

Usage

```
bsdPar(N, mti, groups = LETTERS[1:2])
```

Arguments

N	integer for the total sample size of the trial.
mti	maximum tolerated imbalance in patient numbers during the trial.
groups	character vector of labels for the different treatments.

Details

Tossing a fair coin as long as the difference in group sizes doesn't exceed the mti . If the mti is reached a deterministic allocation is done, so that the difference in group sizes is reduced.

Value

S4 object of the class bsdPar.

References

J. F. Soares and C. F. Jeff Wu (1983) Some Restricted Randomization Rules in Sequential Designs. *Comm. in Stat.*, **12**, 2017-34.

See Also

Other randomization procedures: [abcdPar](#), [bbcdPar](#), [chenPar](#), [crPar](#), [createParam](#), [ebcPar](#), [gbcdPar](#), [hadaPar](#), [mpPar](#), [pbrPar](#), [rarPar](#), [rpbrPar](#), [rtbdPar](#), [tbdPar](#), [udPar](#)

bsdRand

Sampling algorithm for BSD

Description

Sampling algorithm for BSD

Usage

```
bsdRand(N, mti, K = 2)
```

Arguments

N	integer for the total sample size of the trial.
mti	maximum tolerated imbalance in patient numbers during the trial.
K	number of treatment groups (e.g. K=2 if we compare one experimental against one control treatment).

Value

A vector with the allocation sequence for a clinical trial. It will contain a zero (resp. 1) at position *i*, when patient *i* is allocated to treatment A (resp. B).

References

J. F. Soares and C. F. Jeff Wu (1983) Some Restricted Randomization Rules in Sequential Designs. *Comm. in Stat.*, **12**, 2017-34.

`chenPar`*Representing Chen's Design*

Description

Represents the randomization procedure Chen's Design.

Usage

```
chenPar(N, mti = N, p = 0.5, groups = LETTERS[1:2])
```

Arguments

<code>N</code>	integer for the total sample size of the trial.
<code>mti</code>	maximum tolerated imbalance in patient numbers during the trial.
<code>p</code>	success probability of the biased coin (e.g. in Efron's Biased Coin Design).
<code>groups</code>	character vector of labels for the different treatments.

Details

Flip a biased coin with probability p in favour of the treatment which is allocated less frequently as long as the difference in group sizes doesn't exceed the `mti`. If the `mti` is reached a deterministic allocation is done, so that the difference in group sizes is reduced. If both treatments have been assigned equally often a fair coin is tossed.

Value

S4 object of the class `chenPar`.

References

Chen Yung-Pin (1999) Biased coin design with imbalance tolerance. *Comm. in Stat.*, **15**, 953-975.

See Also

Other randomization procedures: [abcdPar](#), [bbcdPar](#), [bsdPar](#), [crPar](#), [createParam](#), [ebcPar](#), [gbcdPar](#), [hadaPar](#), [mpPar](#), [pbrPar](#), [rarPar](#), [rpbrPar](#), [rtbdPar](#), [tbdPar](#), [udPar](#)

chenRand *Representing Chen's Design*

Description

Representing Chen's Design

Usage

```
chenRand(N, mti, p, K = 2)
```

Arguments

N	integer for the total sample size of the trial.
mti	maximum tolerated imbalance in patient numbers during the trial.
p	success probability of the biased coin (e.g. in Efron's Biased Coin Design).
K	number of treatment groups (e.g. K=2 if we compare one experimental against one control treatment).

Value

A vector with the allocation sequence for a clinical trial. It will contain a zero (resp. 1) at position *i*, when patient *i* is allocated to treatment A (resp. B).

References

Chen Yung-Pin (1999) Biased coin design with imbalance tolerance. *Comm. in Stat.*, 15, 953-975.

chronBias *Representing chronological bias*

Description

Represents the issue of chronological bias in a clinical trial.

Usage

```
chronBias(type, theta, method, saltus, alpha = 0.05)
```

Arguments

type	character string, should be one of "linT", "logT", or "stepT", see Details.
theta	factor of the time trend for further details see type.
method	character string, should be one of "sim" or "exact", see Description.
saltus	integer or missing specifying the patient index (i.e. position) of the step in case of step time trend.
alpha	significance level

Details

Chronological bias can be an issue in the design of a clinical trial. The `chronBias` function is a constructor function for an S4 object of the class `chronBias` representing the issue of chronological bias, s.a. time trends, in a clinical trial. It supports two possible modes, `method="sim"` and `method="exact"`, and three different types of trend.

If `method="sim"`, the object represents the simulated type-I-error rate given the level `alpha`, the selection effect `eta` and the biasing strategy `type`. When calling `assess` for a `chronBias` object with `method="sim"`, one test decision is computed for each sequence of `randSeq`. The type-I-error rate (power) is the proportion of falsely (correctly) rejected null hypotheses.

If `method="exact"`, the object represents the exact type-I-error probability given the level `alpha`, the selection effect `eta` and the biasing strategy `type`. When calling `assess` for a `chronBias` object with `method="exact"`, the exact p -value of each randomization sequence is computed. So far, this is only supported for normal endpoints. Then the type-I-error probability is the sum of the corresponding quantiles of the doubly noncentral t-distribution.

Types of chronological bias:

`type = "linT"` Represents linear time trend. Linear time trend means that the expected response of the patients increases evenly by `theta` with every patient included in the study, until reaching `N * theta` after `N` patients. Linear time trend may occur as a result of gradually relaxing in- or exclusion criteria throughout the trial. It can be presented by the formula:

$$f(i) = i\theta$$

`type = "logT"` Represents logistic time trend. Logistic time trend means that the expected response of the patients increases logistically in the patient index by `theta` with every patient included in the study, until reaching `log(N) * theta` after `N` patients. Logistic time trend may occur as a result of a learning curve, i.e. in a surgical trial. It can be presented by the formula:

$$\log(i)\theta$$

`type = "stepT"` Represents step trend. Step trend means that the expected response of the patients increases by `theta` after a given point ("`saltus`") in the allocation process. Step trend may occur if a new device is used after the point `c = "saltus"`, or if the medical personal changes after after this point. Step time trend can be presented by the formula:

$$f(i) = 1_{c \leq i \leq N} \theta$$

Value

S4 object of class `chronBias`, a formal representation of the issue of chronological bias in a clinical trial.

References

G. K. Rosenkranz (2011) The impact of randomization on the analysis of clinical trials. *Statistics in Medicine*, **30**, 3475-87.

M. Tamm and R.-D. Hilgers (2014) Chronological bias in randomized clinical trials under different types of unobserved time trends. *Methods of Information in Medicine*, **53**, 501-10.

See Also

Other issues: [combineBias](#), [corGuess](#), [imbal](#), [issue](#), [selBias](#), [setPower](#)

coin	<i>Function returning the coin slot of an S4 object</i>
------	---

Description

Function returning the coin slot of an S4 object

Usage

```
coin(obj)
```

Arguments

obj object extending class randPar or randSeq

combineBias	<i>Combined additive bias criterion</i>
-------------	---

Description

This class combines a selBias object and a chronBias object to a new object. In the analysis within the new object the two types of bias are treated as additive. effect.

Usage

```
combineBias(selBias, chronBias)
```

Arguments

selBias object of class selBias
chronBias object of class chronBias

See Also

Other issues: [chronBias](#), [corGuess](#), [imbal](#), [issue](#), [selBias](#), [setPower](#)

Examples

```
chronBias <- chronBias(type="linT", theta=1, method="sim")  
selBias <- selBias(type="CS", eta=1, method="sim")  
combineBias(selBias, chronBias)
```

`compare`*Comparison of randomization procedures*

Description

Compares randomization procedures based on a specified issue in clinical trials.

Usage

```
compare(issue, ..., endp)

## S4 method for signature 'issue,missing'
compare(issue, ..., endp)

## S4 method for signature 'issue,endpoint'
compare(issue, ..., endp)
```

Arguments

<code>issue</code>	object of class <code>issue</code> .
<code>...</code>	at least one object of class <code>randSeq</code> or a list of objects of class <code>randSeq</code> .
<code>endp</code>	object of class <code>endpoint</code> , or <code>missing</code> .

Details

Randomization procedures behave differently with respect to issues like selection bias, chronological bias, or loss in power estimation. The `compare` function evaluates the behaviour of randomization procedures with respect to one issue. Its first argument should represent one of the implemented [issues](#). The second argument should be any number of objects of the class `randSeq`. These objects represent the randomization procedures for the planned comparison. The last argument `endp` may be provided if the assessment should take the distribution of the treatment groups into account, e.g. for power evaluation.

Value

S4 object of class `comparison` summarizing the comparison of the randomization procedures.

See Also

Representation of randomization procedures: [randPar](#)
Generation of randomization sequences: [genSeq](#)
[issues](#) for the assessment of randomization sequences

Examples

```
# compare Random Allocation Rule and Big Stick for N = 4
# with respect to the correct guesses
RAR <- getAllSeq(rarPar(4))
BSD <- getAllSeq(bsdPar(4, mti = 2))
corGuess <- corGuess("CS")
(comp <- compare(corGuess, RAR, BSD))
plot(comp)

# compare the same procedures with respect to selection bias
endp <- normEndp(c(2, 2), c(1, 1))
selBias <- selBias("CS", 4, "exact")
(comp <- compare(selBias, RAR, BSD, endp = endp))
plot(comp)
```

corGuess

Representing the expected number of correct guesses

Description

Represents the expected number of correct guesses of randomization sequences.

Usage

```
corGuess(type)
```

Arguments

type character string, should be one of "CS" or "DS", see Details.

Details

Selection bias can be an issue in the design of a clinical trial. The expected number of correct guesses is one measure for selection bias. The corGuess function is a constructor function for an S4 object of the class corGuess representing the issue of correct guesses in a clinical trial. The parameter type takes the following values:

"CS" refers to "convergence strategy", i.e. the investigator predicts the treatment which has hitherto occurred less often.

"DS" refers to "divergence strategy", i.e. the investigator predicts the treatment which has hitherto occurred more often.

Value

S4 object of class corGuess, a formal representation of the issue of correct guesses in a clinical trial.

References

D. Blackwell and J.L. Hodges Jr. (1957) Design for the control of selection bias. *Annals of Mathematical Statistics*, **25**, 449-60.

See Also

Other issues: [chronBias](#), [combineBias](#), [imbal](#), [issue](#), [selBias](#), [setPower](#)

 createParam

Representing any randomization procedure

Description

Represents any randomization procedure for a two-armed clinical trial.

Usage

```
createParam(method, N, mti, bc, rb, p, ini, add, filledBlock)
```

Arguments

method	method that is used to generate the (random) allocation sequence. It can take values PBR, RAR, HAD, PWR, EBC, BSD, CR, TBD, UD, and MP.
N	integer for the total sample size of the trial.
mti	maximum tolerated imbalance in patient numbers during the trial.
bc	vector which contains the lengths k_1, \dots, k_l of each block. This means that the vector bc will have one entry for each block.
rb	block lengths of the blocks that can be selected equiprobable at random.
p	success probability of the biased coin (e.g. in Efron's Biased Coin Design).
ini	integer representing the initial urn composition.
add	integer representing the number of balls that are added to the urn in each step.
filledBlock	logical whether the last block should be filled or not.

Details

Depending on the input of the user, createParam creates an object representing a randomization procedure for a two-armed clinical trial (see also [randPar](#)).

Value

S4object of the corresponding randomization procedure class.

See Also

Other randomization procedures: [abcdPar](#), [bbcdPar](#), [bsdPar](#), [chenPar](#), [crPar](#), [ebcPar](#), [gbcdPar](#), [hadaPar](#), [mpPar](#), [pbrPar](#), [rarPar](#), [rpbrPar](#), [rtbdPar](#), [tbdPar](#), [udPar](#)

createSeq	<i>Query to create a randomization sequence of a particular randomization procedure</i>
-----------	---

Description

This function is a query to create an corresponding randomization sequence for a two-armed clinical trial. If `file` is defined, the generated sequence is automatically saved to the corresponding path.

Usage

```
createSeq(file)
```

Arguments

<code>file</code>	A connection, or a character string naming the file to write to.
-------------------	--

Value

an object `Param`, which is available

<code>crPar</code>	<i>Representing Complete Randomization</i>
--------------------	--

Description

Represents the randomization procedure Complete Randomization.

Usage

```
crPar(N, K = 2, ratio = rep(1, K), groups = LETTERS[1:K])
```

Arguments

<code>N</code>	integer for the total sample size of the trial.
<code>K</code>	number of treatment groups (e.g. <code>K=2</code> if we compare one experimental against one control treatment).
<code>ratio</code>	vector of length <code>K</code> . The total sample number <code>N</code> and all used block lengths (<code>bc</code>) have to be divisible by <code>sum(ratio)</code> .
<code>groups</code>	character vector of labels for the different treatments.

Details

Toss a fair coin `N` times in case `K=2` and assign the treatments according to the result of the coin. In case of `K>2`, replace the coin by a die with `K` sides.

Value

S4 object of the class crPar.

References

W. F. Rosenberger and J. M. Lachin (2002) *Randomization in Clinical Trials*. Wiley.

See Also

Other randomization procedures: [abcdPar](#), [bbcdPar](#), [bsdPar](#), [chenPar](#), [createParam](#), [ebcPar](#), [gbcdPar](#), [hadaPar](#), [mpPar](#), [pbrPar](#), [rarPar](#), [rpbrPar](#), [rtbdPar](#), [tbdPar](#), [udPar](#)

 derFunc

Representing Derringer-Suich desirability functions

Description

Represents the Derringer-Suich desirability approach.

Usage

```
derFunc(TV, SLs, b)
```

Arguments

TV	numeric specifying the optimal desired value called the target value.
SLs	numeric vector of length at most 2 specifying the lower and/or upper specified border.
b	numeric vector of length at most 2 specifying the weight(s) for the punishment of

Details

derFunc represents the framework for left, right and two-sided desirability functions introduced by Derringer and Suich (1980). For all three different kinds of desirability functions the parameter TV must be specified. If the parameter SLs has length 1, either the left- or right-sided desirability function is created depending from whether the value is smaller (left-sided) or greater (right-sided) than the target value. By specifying SLs as a vector of length 2 a two-sided desirability function is created where the lower specified border is determined as the smaller value of SLs and thus the upper specified border is determined as the greater value. If there are no values specified for the weights, then they are automatically set to 1 (linear loss).

Value

S4 object of class derFunc, a formal representation of desirability functions introduced by Derringer and Suich.

References

Derringer, G., and Suich, R., (1980) Simultaneous Optimization of Several Response Variables. *Journal of Quality Technology*, **12**, 214-219.

See Also

Other desirability topics: [evaluate](#), [getDesScores](#), [plotDes](#), [plotEv](#), [probUnDes](#)

Examples

```
# Create an object of a left-sided desirability function
dLeft <- derFunc(0.5, 0.3, 2)
# Create an object of a right-sided desirability function
dRight <- derFunc(0.5, 0.8, 1)
# Create an object of a two-sided desirability function
dLR <- derFunc(0.5, c(0.3, 0.9), c(3, 1))
```

derFunc-class

Parameters generic

Description

Parameters generic

desirability

Desirability functions within the scope of clinical trials

Description

Illustrates the interplay between functions related to desirability indices.

Details

Currently, `randomizeR` encompasses the class of desirability functions introduced by Derringer and Suich (1980) and corresponding functions to evaluate and compare randomization sequences which have been assessed on the basis of desirability indices of specific issues:

- **derFunc** represents the class of desirability functions according to Derringer-Suich (1980).
- **getDesScores** can be applied to an object of class `assessment` together with prespecified desirability functions to compare the behaviour of randomization sequences (on a common scale [0,1]).
- **plotDes** plots a `desScores` object on a radar chart.
- **evaluate** performs a comparison of sequences from different randomization sequences on the basis of object of the class `desScores`.
- **plotEv** plots an `evaluation` object on a radar chart.
- **probUnDes** computes the probability of undesired randomization sequences with respect to certain issues and desirability functions.

Examples

```

# Suppose we would like to perform a comparison of sequences from different
# randomization procedures with the help of desirability functions:

issue1 <- corGuess("CS")
issue2 <- chronBias(type = "linT", theta = 1/4, method = "exact")
RAR <- getAllSeq(rarPar(4))
BSD <- getAllSeq(bsdPar(4, mti = 2))
A1 <- assess(RAR, issue1, issue2, endp = normEndp(c(0,0), c(1,1)))
A2 <- assess(BSD, issue1, issue2, endp = normEndp(c(0,0), c(1,1)))

d1 <- derFunc(TV = 0.5, 0.75, 2)
d2 <- derFunc(0.05, c(0, 0.1), c(1, 1))

# By applying the \code{getDesScores} function to the assessment output together
# with the specified desirability functions the behaviour of randomization sequences
# is evaluated and scaled to [0,1]:
DesScore <- getDesScores(A1, d1, d2, weights = c(5/6, 1/6))
DesScore2 <- getDesScores(A2, d1, d2, weights = c(5/6, 1/6))

# Plotting the desScores objects:
plotDes(DesScore, quantiles = TRUE)
plotDes(DesScore2, quantiles = TRUE)

# Summarizing the results of getDesScore with respect to the statistic "mean":
evaluate(DesScore, DesScore2)

# Plotting the evaluation objects allows a visualized comparison:
plotEv(evaluate(DesScore, DesScore2))

# Which randomization procedure produces more undesired randomization sequences
# with respect to certain issues and desirability functions?
probUnDes(DesScore)
probUnDes(DesScore2)

```

doublyF_opt

Distribution function of the non central F-distribution

Description

Distribution function of the non central F-distribution

Usage

```
doublyF_opt(x, df1, df2, lambda1, lambda2, acc = 100, ex = 5)
```

Arguments

x	quantile, a single numeric value
df1	first degree of freedom, a single integer value
df2	second degree of freedom, a single integer value
lambda1	first non centrality parameter, a single numeric value
lambda2	second non centrality parameter, a single numeric value
acc	accuracy; last index of the approximation of the infinite sum
ex	exactness; break early, if the summands are smaller than 10^{-ex}

Value

Probability of observing a value larger than x

doublyF_value	<i>Rejection probability for one sequence in the presence of selection bias</i>
---------------	---

Description

Rejection probability for one sequence in the presence of selection bias

Usage

```
doublyF_value(R, bias, endp)
```

Arguments

R	object of type randSeq representing the randomization procedure
bias	selection or chronological bias object, containing eta(or theta) and alpha
endp	endpoint object, containing mu and sigma

Value

data frame containing the non centrality parameters, the degrees of freedom, the 1-alpha quantile corresponding to the central f distribution and a numeric value for probability of false rejection of the null hypothesis of no difference in the presence of selection bias.

doublyF_values	<i>Rejection probability in case of selection bias in multi-arm trials</i>
----------------	--

Description

calculates the non-centrality parameters of the F-distribution under third order selection bias.

Usage

```
doublyF_values(randSeq, bias, endp)
```

Arguments

randSeq	the object containing the randomization sequences
bias	selection bias object, containing eta and alpha
endp	endpoint object, containing mu and sigma

Value

data frame with the sequences and their corresponding ncps and rejection probabilities.

doublyT	<i>Approximation of the distribution function of the doubly noncentral t-distribution</i>
---------	---

Description

Computes the value of the distribution function of the doubly noncentral t-distribution at x.

Usage

```
doublyT(x, df, delta, lambda, lb = 0, ub)
```

Arguments

x	a variable x.
df	degrees of freedom (i.a. N-2).
delta	first noncentrality parameter of the doubly noncentral t-distribution.
lambda	(second) noncentrality parameter of the doubly noncentral t-distribution.
lb	lower bound for the starting value of the poisson distribution.
ub	upper bound for the last value of the poisson distribution.

Value

Distribution value of the doubly noncentral t-distribution at x.

doublyTValues	<i>Calculation of the biased type-one-error (resp. power) of Student's t-test</i>
---------------	---

Description

Computes the biased type-one-error (resp. power) of Student's t-test due to shifts in the expectation vectors in both treatment groups.

Usage

```
doublyTValues(randSeq, bias, endp)
```

Arguments

randSeq	object of the class randSeq.
bias	object of the class bias.
endp	object of the class endpoint.

Value

the biased type-one-error (resp. power) of all randomization sequences.

Examples

```
myPar <- crPar(4)
M <- getAllSeq(myPar)
cs <- selBias("CS", 1, "exact")
endp <- normEndp(mu = c(0, 0), sigma = c(1, 1))
doublyTValues(M, cs, endp)
```

ebcPar	<i>Representing Efron's Biased Coin Design</i>
--------	--

Description

Represents the randomization procedure Efron's Biased Coin Design.

Usage

```
ebcPar(N, p, groups = LETTERS[1:2])
```

Arguments

N	integer for the total sample size of the trial.
p	success probability of the biased coin (e.g. in Efron's Biased Coin Design).
groups	character vector of labels for the different treatments.

Details

Flip a biased coin with probability p in favour of the treatment which is allocated less frequently. If both treatments have been assigned equally often a fair coin is tossed.

Value

S4 object of the class ebcPar.

References

B. Efron (1971) Forcing a sequential experiment to be balanced. *Biometrika*, **58**, 403-17.

See Also

Other randomization procedures: [abcdPar](#), [bbcdPar](#), [bsdPar](#), [chenPar](#), [crPar](#), [createParam](#), [gbcdPar](#), [hadaPar](#), [mpPar](#), [pbrPar](#), [rarPar](#), [rpbrPar](#), [rtbdPar](#), [tbdPar](#), [udPar](#)

evaluate

Evaluation of several randomization procedures with respect to certain desirability functions applied to specified issues.

Description

Evaluation of several randomization procedures with respect to certain desirability functions applied to specified issues.

Usage

```
evaluate(..., statistic)
```

```
## S4 method for signature 'missing'
evaluate(..., statistic)
```

```
## S4 method for signature 'character'
evaluate(..., statistic)
```

Arguments

...	at least one object of the class desScores or a list of objects of the class desScores.
statistic	character string that specifies on the basis of which statistic the evaluate function should be applied. The statistic can be chosen from "mean", "median", "min" or "max".

Details

The `evaluate` function allows the user to compare and evaluate different randomization procedures. It expects a number of objects that result when applying the `getDesScores` function to an `assess` object and specified desirability functions. The `evaluate` function summarizes the desirability scores of each randomization procedure on the basis of a prespecified statistic and incorporates them into a data frame. If no statistic is specified then it is automatically set to mean. If the function is applied to only one object it corresponds simply to `summary(getDesScores(...))`.

Value

S4 object of class `evaluation` Comparison of randomization procedures with respect to desirability functions applied to specified issues, summarized by a prespecified statistic.

References

D. Schindler *Assessment of Randomization Procedures in the Presence of Selection and Chronological Bias*. PhD Thesis.

See Also

Representation of randomization procedures: [randPar](#)

Generation of randomization sequences: [genSeq](#)

[issues](#) for the desirability of randomization sequences

Other desirability topics: [derFunc](#), [getDesScores](#), [plotDes](#), [plotEv](#), [probUnDes](#)

Examples

```
# Compare Random Allocation Rule to Big Stick Design with respect to different issues
# and their corresponding desirability functions
issue1 <- corGuess("CS")
issue2 <- corGuess("DS")
RAR <- getAllSeq(rarPar(4))
BSD <- getAllSeq(bsdPar(4, mti = 2))
A1 <- assess(RAR, issue1, issue2)
A2 <- assess(BSD, issue1, issue2)

d1 <- derFunc(TV = 0.1, 0.7, 2)
d2 <- derFunc(0.5, c(0.3, 0.8), c(1, 1))
DesScore <- getDesScores(A1, d1, d2, weights = c(5/6, 1/6))
DesScore2 <- getDesScores(A2, d1, d2, weights = c(5/6, 1/6))

evaluate(DesScore, DesScore2)
evaluate(DesScore, DesScore2, statistic = "max")
```

`gbcdPar`*Representing Generalized Biased Coin Design*

Description

Represents the randomization procedure Generalized Biased Coin Design.

Usage

```
gbcdPar(N, rho, groups = LETTERS[1:2])
```

Arguments

<code>N</code>	integer for the total sample size of the trial.
<code>rho</code>	nonnegative parameter which may be adjusted according to how strongly it is desired to balance the experiment. If $\rho = 1$, we have Wei's urn design with $\alpha = 0$. If $\rho = 0$, we have complete randomization.
<code>groups</code>	character vector of labels for the different treatments.

Details

Generalization of Wei's urn and Efron's biased coin design.

Value

S4 object of the class `gbcdPar`.

References

R. L. Smith (1984) Sequential treatment allocation using biased coin designs. *Journal of the Royal Statistical Society B*, **46**, 519-543.
W. F. Rosenberger and J. M. Lachin (2002) Randomization in Clinical Trials. *Wiley*, 64-65

See Also

Other randomization procedures: [abcdPar](#), [bbcdPar](#), [bsdPar](#), [chenPar](#), [crPar](#), [createParam](#), [ebcPar](#), [hadaPar](#), [mpPar](#), [pbrPar](#), [rarPar](#), [rpbrPar](#), [rtbdPar](#), [tbdPar](#), [udPar](#)

gbcdRand	<i>Sampling algorithm for gbcd</i>
----------	------------------------------------

Description

Sampling algorithm for gbcd

Usage

```
gbcdRand(N, rho, K = 2)
```

Arguments

N	integer for the total sample size of the trial.
rho	nonnegative parameter which may be adjusted according to how strongly it is desired to balance the experiment. If $\rho = 1$, we have Wei's urn design with $\alpha = 0$. If $\rho = 0$, we have complete randomization.
K	number of treatment groups (e.g. $K=2$ if we compare one experimental against one control treatment).

Value

A vector with the allocation sequence for a clinical trial. It will contain a zero (resp. 1) at position i , when patient i is allocated to treatment A (resp. B).

References

R. L. Smith (1984) Sequential treatment allocation using biased coin designs. *Journal of the Royal Statistical Society B*, **46**, 519-543.
 W. F. Rosenberger and J. M. Lachin (2002) Randomization in Clinical Trials. *Wiley*, 64-65

generateAllSequences	<i>Complete set of randomization sequences</i>
----------------------	--

Description

Computes all randomization sequences for the given randomization procedure, and stores them in an object along with the parameters belonging to the randomization procedure.

Usage

```
getAllSeq(obj)

## S4 method for signature 'abcdPar'
getAllSeq(obj)

## S4 method for signature 'bbcdPar'
getAllSeq(obj)

## S4 method for signature 'ebcPar'
getAllSeq(obj)

## S4 method for signature 'bsdPar'
getAllSeq(obj)

## S4 method for signature 'chenPar'
getAllSeq(obj)

## S4 method for signature 'crPar'
getAllSeq(obj)

## S4 method for signature 'gbcdPar'
getAllSeq(obj)

## S4 method for signature 'hadaPar'
getAllSeq(obj)

## S4 method for signature 'mpPar'
getAllSeq(obj)

## S4 method for signature 'pbrPar'
getAllSeq(obj)

## S4 method for signature 'rarPar'
getAllSeq(obj)

## S4 method for signature 'tbdPar'
getAllSeq(obj)

## S4 method for signature 'udPar'
getAllSeq(obj)
```

Arguments

`obj` object specifying the randomization procedure, see [randPar](#) or [createParam](#).

Details

getAllSeq is a generic function which dispatches different methods depending on the type of input. The set of sequences of a procedure is computed by enumerating all possible sequences and eliminating those that are not possible in the randomization procedure specified by obj. The parameters of the randomization procedure are saved along with the sequences to ensure reproducibility of the results.

Value

An object inheriting from [randSeq](#), representing the set of randomization sequences for the given parameters. The output consists of the parameters used for the generation of the randomization sequences (see [createParam](#)) and the matrix M that stores the randomization sequences in its rows.

See Also

[createParam](#)

Examples

```
# CR
myPar <- crPar(6)
getAllSeq(myPar)

# EBC
myPar <- ebcPar(6, 0.667)
getAllSeq(myPar)

# BSD
myPar <- bsdPar(6, 2)
getAllSeq(myPar)

# PBR
myPar <- pbrPar(c(4, 2))
getAllSeq(myPar)

# RAR
myPar <- rarPar(8)
getAllSeq(myPar)

# MP
myPar <- mpPar(8, 2)
getAllSeq(myPar)

# HAD
myPar <- hadaPar(8)
getAllSeq(myPar)

# TBD
myPar <- tbdPar(8)
getAllSeq(myPar)
```

```
# GBCD
myPar <- gbcdPar(8, 2)
getAllSeq(myPar)

# ABCD
myPar <- abcdPar(8, 3)
getAllSeq(myPar)

# BBCD
myPar <- bbcdPar(8, 5)
getAllSeq(myPar)

# CHEN
myPar <- chenPar(8, 2, 0.667)
getAllSeq(myPar)
```

generateRandomSequences

Generate random sequences

Description

Generates randomization sequences from a given randomization procedure.

Usage

```
genSeq(obj, r, seed)
```

```
## S4 method for signature 'abcdPar,numeric,numeric'
genSeq(obj, r, seed)
```

```
## S4 method for signature 'abcdPar,numeric,missing'
genSeq(obj, r, seed)
```

```
## S4 method for signature 'abcdPar,missing,numeric'
genSeq(obj, r, seed)
```

```
## S4 method for signature 'abcdPar,missing,missing'
genSeq(obj, r, seed)
```

```
## S4 method for signature 'bbcdPar,numeric,numeric'
genSeq(obj, r, seed)
```

```
## S4 method for signature 'bbcdPar,numeric,missing'
genSeq(obj, r, seed)
```

```
## S4 method for signature 'bbcdPar,missing,numeric'
```



```
genSeq(obj, r, seed)

## S4 method for signature 'bbcdPar,missing,missing'
genSeq(obj, r, seed)

## S4 method for signature 'ebcPar,numeric,numeric'
genSeq(obj, r, seed)

## S4 method for signature 'ebcPar,missing,numeric'
genSeq(obj, r, seed)

## S4 method for signature 'ebcPar,numeric,missing'
genSeq(obj, r, seed)

## S4 method for signature 'ebcPar,missing,missing'
genSeq(obj, r, seed)

## S4 method for signature 'bsdPar,numeric,numeric'
genSeq(obj, r, seed)

## S4 method for signature 'bsdPar,numeric,missing'
genSeq(obj, r, seed)

## S4 method for signature 'bsdPar,missing,numeric'
genSeq(obj, r, seed)

## S4 method for signature 'bsdPar,missing,missing'
genSeq(obj, r, seed)

## S4 method for signature 'chenPar,numeric,numeric'
genSeq(obj, r, seed)

## S4 method for signature 'chenPar,numeric,missing'
genSeq(obj, r, seed)

## S4 method for signature 'chenPar,missing,numeric'
genSeq(obj, r, seed)

## S4 method for signature 'chenPar,missing,missing'
genSeq(obj, r, seed)

## S4 method for signature 'crPar,numeric,numeric'
genSeq(obj, r, seed)

## S4 method for signature 'crPar,missing,numeric'
genSeq(obj, r, seed)

## S4 method for signature 'crPar,numeric,missing'
```

```
genSeq(obj, r, seed)

## S4 method for signature 'crPar,missing,missing'
genSeq(obj, r, seed)

## S4 method for signature 'gbcdPar,numeric,numeric'
genSeq(obj, r, seed)

## S4 method for signature 'gbcdPar,numeric,missing'
genSeq(obj, r, seed)

## S4 method for signature 'gbcdPar,missing,numeric'
genSeq(obj, r, seed)

## S4 method for signature 'gbcdPar,missing,missing'
genSeq(obj, r, seed)

## S4 method for signature 'hadaPar,numeric,numeric'
genSeq(obj, r, seed)

## S4 method for signature 'hadaPar,missing,numeric'
genSeq(obj, r, seed)

## S4 method for signature 'hadaPar,numeric,missing'
genSeq(obj, r, seed)

## S4 method for signature 'hadaPar,missing,missing'
genSeq(obj, r, seed)

## S4 method for signature 'mpPar,numeric,numeric'
genSeq(obj, r, seed)

## S4 method for signature 'mpPar,missing,numeric'
genSeq(obj, r, seed)

## S4 method for signature 'mpPar,numeric,missing'
genSeq(obj, r, seed)

## S4 method for signature 'mpPar,missing,missing'
genSeq(obj, r, seed)

## S4 method for signature 'pbrPar,missing,numeric'
genSeq(obj, r, seed)

## S4 method for signature 'pbrPar,numeric,numeric'
genSeq(obj, r, seed)

## S4 method for signature 'pbrPar,missing,missing'
```

```
genSeq(obj, r, seed)

## S4 method for signature 'pbrPar,numeric,missing'
genSeq(obj, r, seed)

## S4 method for signature 'rarPar,numeric,numeric'
genSeq(obj, r, seed)

## S4 method for signature 'rarPar,missing,numeric'
genSeq(obj, r, seed)

## S4 method for signature 'rarPar,numeric,missing'
genSeq(obj, r, seed)

## S4 method for signature 'rarPar,missing,missing'
genSeq(obj, r, seed)

## S4 method for signature 'rpbrPar,missing,numeric'
genSeq(obj, r, seed)

## S4 method for signature 'rpbrPar,numeric,numeric'
genSeq(obj, r, seed)

## S4 method for signature 'rpbrPar,missing,missing'
genSeq(obj, r, seed)

## S4 method for signature 'rpbrPar,numeric,missing'
genSeq(obj, r, seed)

## S4 method for signature 'tbdPar,numeric,numeric'
genSeq(obj, r, seed)

## S4 method for signature 'tbdPar,missing,numeric'
genSeq(obj, r, seed)

## S4 method for signature 'tbdPar,numeric,missing'
genSeq(obj, r, seed)

## S4 method for signature 'tbdPar,missing,missing'
genSeq(obj, r, seed)

## S4 method for signature 'rtbdPar,numeric,numeric'
genSeq(obj, r, seed)

## S4 method for signature 'rtbdPar,missing,numeric'
genSeq(obj, r, seed)

## S4 method for signature 'rtbdPar,numeric,missing'
```

```

genSeq(obj, r, seed)

## S4 method for signature 'rtbdPar,missing,missing'
genSeq(obj, r, seed)

## S4 method for signature 'udPar,numeric,numeric'
genSeq(obj, r, seed)

## S4 method for signature 'udPar,missing,numeric'
genSeq(obj, r, seed)

## S4 method for signature 'udPar,numeric,missing'
genSeq(obj, r, seed)

## S4 method for signature 'udPar,missing,missing'
genSeq(obj, r, seed)

```

Arguments

obj	object specifying the randomization procedure, see randPar or createParam .
r	numeric indicating the number of random sequences to be generated at random, or missing.
seed	a single value, interpreted as an integer, that specifies the seed for the random number generation.

Details

genSeq generates randomization sequences for a randomization procedure as defined by the input paramters. genSeq has two modes, according to the input.

1. genSeq(obj, r): gives r random sequences from the design specified by obj, along with the parameters stored in obj.
2. genSeq(obj): gives one random sequences from the design specified by obj, along with the parameters stored in obj.

The sequences are generated by using the Monte-Carlo sampling technique to sample from the true distribution of the sequences according to the randomization procedure specified by obj. The parameters of the randomization procedure are saved along with the sequences to ensure reproducibility of the results.

Value

An object inheriting from [randSeq](#), representing the r randomisation sequences generated at random for the specified randomization procedure. The output consists of the parameters used for the generation of the randomization sequences (see [createParam](#)) and the matrix M that stores the randomization sequences in its r rows. If r is missing, one sequence is generated by default.

Examples

```
# CR
myPar <- crPar(10)
genSeq(myPar, 4)
genSeq(myPar)

# EBC
myPar <- ebcPar(10, 0.667)
genSeq(myPar, 4)
genSeq(myPar)

# BSD
myPar <- bsdPar(10, 2)
genSeq(myPar, 4)
genSeq(myPar)

# PBR
myPar <- pbrPar(c(4, 4))
genSeq(myPar, 4)
genSeq(myPar)

# RAR
myPar <- rarPar(10)
genSeq(myPar, 4)
genSeq(myPar)

# MP
myPar <- mpPar(10, 2)
genSeq(myPar, 4)
genSeq(myPar)

# HAD
myPar <- hadaPar(10)
genSeq(myPar, 4)
genSeq(myPar)

# UD
myPar <- udPar(8, 0, 1)
genSeq(myPar, 4)
genSeq(myPar)

# TBD
myPar <- tbdPar(c(4, 6))
genSeq(myPar, 4)
genSeq(myPar)

# GBCD
myPar <- gbcdPar(8, 2)
genSeq(myPar, 4)
genSeq(myPar)

# ABCD
```

```

myPar <- abcdPar(8, 3)
genSeq(myPar, 4)
genSeq(myPar)

# BBCD
myPar <- bbcdPar(8, 5)
genSeq(myPar, 5)
genSeq(myPar)

# CHEN
myPar <- chenPar(8, 2, 0.667)
genSeq(myPar, 5)
genSeq(myPar)

```

genNcps

Calculation of the NCPs of each randomization sequence for the doubly noncentral t-distribution

Description

Computes the noncentrality parameters delta and lambda for the doubly noncentral t-distribution of each randomization sequence.

Usage

```
genNcps(randSeq, bias, endp)
```

Arguments

randSeq	object of the class randSeq.
bias	object of the class bias.
endp	object of the class endpoint.

Value

matrix containing the noncentrality parameters delta and lambda of all randomization sequences.

Examples

```

myPar <- crPar(4)
M <- getAllSeq(myPar)
cs <- selBias("CS", 1, "exact")
endp <- normEndp(mu = c(0, 0), sigma = c(1, 1))
genNcps(M, cs, endp)

```

`getbiasCS1`*Biasing Policy for a Group of Favoured Treatments*

Description

Calculate vector with the selection bias for each patient

Usage

```
getbiasCS1(R, K, pref)
```

Arguments

R	randomization sequence, a integer vector with entries $1, \dots, K$, of length N
K	number of treatment groups, a single integer value
pref	preferred groups for the guessing

Value

vector with the selection bias for each patient

`getbiasCS2`*Biasing Policy for Avoiding the Placebo Treatment*

Description

Calculate vector with the selection bias for each patient

Usage

```
getbiasCS2(R, K, avoid)
```

Arguments

R	randomization sequence, a integer vector with entries $1, \dots, K$, of length N
K	number of treatment groups, a single integer value
avoid	avoided groups for the guessing

Value

vector with the selection bias for each patient

getCorGuesses	<i>Matrix of the guesses of the investigator</i>
---------------	--

Description

Calculates the guesses of the investigator of a randomization list following the specified guessing strategy.

Usage

```
getCorGuesses(randSeq, guessing)
```

Arguments

randSeq	object of the class randSeq.
guessing	object of the class corGuess.

Value

Matrix of the guesses of the investigator following the specified guessing strategy. No guess is abbreviated with "nG".

Examples

```
myPar <- bsdPar(10, 2)
M <- genSeq(myPar, 2)
type <- corGuess("CS")
getCorGuesses(M, type)
```

getDesFunc	<i>Type of Desirability function</i>
------------	--------------------------------------

Description

Generates a character vector which specifies the used desirability function and its parameters

Usage

```
getDesFunc(obj)

## S4 method for signature 'derringerLs'
getDesFunc(obj)

## S4 method for signature 'derringerRs'
getDesFunc(obj)
```



```
## S4 method for signature 'derringerTs'  
getDesFunc(obj)
```

Arguments

obj object of the class desFunc.

getDesign	<i>Design of a randomization procedure</i>
-----------	--

Description

Generates a character vector which specifies the used randomization method

Usage

```
getDesign(obj)  
  
## S4 method for signature 'abcdPar'  
getDesign(obj)  
  
## S4 method for signature 'abcdSeq'  
getDesign(obj)  
  
## S4 method for signature 'bbcdPar'  
getDesign(obj)  
  
## S4 method for signature 'bbcdSeq'  
getDesign(obj)  
  
## S4 method for signature 'ebcPar'  
getDesign(obj)  
  
## S4 method for signature 'bsdPar'  
getDesign(obj)  
  
## S4 method for signature 'bsdSeq'  
getDesign(obj)  
  
## S4 method for signature 'chenPar'  
getDesign(obj)  
  
## S4 method for signature 'chenSeq'  
getDesign(obj)  
  
## S4 method for signature 'crPar'
```

```
getDesign(obj)

## S4 method for signature 'crSeq'
getDesign(obj)

## S4 method for signature 'ebcSeq'
getDesign(obj)

## S4 method for signature 'gbcdPar'
getDesign(obj)

## S4 method for signature 'gbcdSeq'
getDesign(obj)

## S4 method for signature 'hadaPar'
getDesign(obj)

## S4 method for signature 'hadaSeq'
getDesign(obj)

## S4 method for signature 'mpPar'
getDesign(obj)

## S4 method for signature 'mpSeq'
getDesign(obj)

## S4 method for signature 'pbrPar'
getDesign(obj)

## S4 method for signature 'pbrSeq'
getDesign(obj)

## S4 method for signature 'rRtbdSeq'
getDesign(obj)

## S4 method for signature 'rRpbrSeq'
getDesign(obj)

## S4 method for signature 'rarPar'
getDesign(obj)

## S4 method for signature 'rarSeq'
getDesign(obj)

## S4 method for signature 'rpbrPar'
getDesign(obj)

## S4 method for signature 'tbdPar'
```

```

getDesign(obj)

## S4 method for signature 'rtbdPar'
getDesign(obj)

## S4 method for signature 'tbdSeq'
getDesign(obj)

## S4 method for signature 'udPar'
getDesign(obj)

## S4 method for signature 'udSeq'
getDesign(obj)

```

Arguments

`obj` object of the class `randSeq` or `randPar`.

<code>getDesScores</code>	<i>Applying desirability functions on issues of individual randomization sequences</i>
---------------------------	--

Description

Applying desirability function on issues of individual randomization sequences.

Usage

```

getDesScores(assess, ..., weights)

## S4 method for signature 'assessment,missing'
getDesScores(assess, ..., weights)

## S4 method for signature 'assessment,numeric'
getDesScores(assess, ..., weights)

```

Arguments

`assess` object of class `assessment`.

`...` at least one object of class `derFunc` or a list of objects of the class `derFunc`.

`weights` weights for computing the geometric mean of several desirability scores. If missing, the issues are automatically equally weighted.

Details

Randomization sequences behave differently with respect to issues like selection bias, chronological bias, or loss in power estimation. The `getDesScores` function evaluates the behaviour of randomization sequences with respect to these issues. The difference to the `assess` function is that it scales them to $[0,1]$ and makes them easier interpretable. The first argument should be a result of the `assess` function. The second argument should be any number of `derFunc` objects that represent the desirability functions. The last argument `weights` may be provided if the desirability functions should be weighted differently.

Value

S4 object of class `desirability` summarizing the desirability of the randomization procedure.

See Also

Representation of randomization procedures: `randPar`

Generation of randomization sequences: `genSeq`

`issues` for the desirability of randomization sequences

Other desirability topics: `derFunc`, `evaluate`, `plotDes`, `plotEv`, `probUnDes`

Examples

```
# Compute the desire-function for the full set of Random Allocation Rule for N=4 patients
sequences <- getAllSeq(rarPar(4))
issue1 <- corGuess("CS")
issue2 <- chronBias("linT", 0.25, "exact")
endp <- normEndp(mu = c(0,0), sigma = c(1,1))
A <- assess(sequences, issue1, issue2, endp = endp)
d1 <- derFunc(0.5, 0.75, 1)
d2 <- derFunc(0.05, 0.1, 1)

D1 <- getDesScores(A, d1, d2)
summary(D1)

D2 <- getDesScores(A, d1, d2, weights = c(3/4, 1/4))
summary(D2)
```

getExpectation

Get expectations of a randomization list

Description

Generates a matrix of the expectations of the included patients in the clinical trial.

Usage

```

getExpectation(randSeq, issue, endp)

## S4 method for signature 'randSeq,missing,normEndp'
getExpectation(randSeq, endp)

## S4 method for signature 'randSeq,power,normEndp'
getExpectation(randSeq, issue, endp)

## S4 method for signature 'randSeq,chronBias,normEndp'
getExpectation(randSeq, issue, endp)

## S4 method for signature 'randSeq,chronBias,missing'
getExpectation(randSeq, issue)

## S4 method for signature 'randSeq,selBias,normEndp'
getExpectation(randSeq, issue, endp)

## S4 method for signature 'randSeq,selBias,missing'
getExpectation(randSeq, issue)

## S4 method for signature 'randSeq,combinedBias,normEndp'
getExpectation(randSeq, issue, endp)

## S4 method for signature 'randSeq,combinedBiasStepTrend,normEndp'
getExpectation(randSeq,
  issue, endp)

```

Arguments

randSeq	object of the class randSeq.
issue	object of the class issue (optional).
endp	object of the class endpoint (optional).

Details

It is assumed that the expectations of the included patients in a clinical trial can be influenced in three different ways:

- The strength of selection bias and the guessing strategy of the investigator (see [selBias](#)).
- The strength of a linear time trend, which is described by an object of the class [chronBias](#).
- The expectations of the investigated treatment groups can be different (see e.g. [normEndp](#)).

Examples

```

myPar <- bsdPar(10, 2)
M <- genSeq(myPar, 2)
cs <- selBias("CS", 2, "sim")

```

```
endp <- normEndp(mu = c(2, 2), sigma = c(1, 1))  
getExpectation(M, cs, endp)
```

getProbabilities *Theoretical probability for randomization sequences*

Description

Calculate theoretical probability for observed randomization sequences

Usage

```
getProb(obj)  
  
## S4 method for signature 'abcdSeq'  
getProb(obj)  
  
## S4 method for signature 'bbcdSeq'  
getProb(obj)  
  
## S4 method for signature 'bsdSeq'  
getProb(obj)  
  
## S4 method for signature 'chenSeq'  
getProb(obj)  
  
## S4 method for signature 'crSeq'  
getProb(obj)  
  
## S4 method for signature 'ebcSeq'  
getProb(obj)  
  
## S4 method for signature 'gbcdSeq'  
getProb(obj)  
  
## S4 method for signature 'hadaSeq'  
getProb(obj)  
  
## S4 method for signature 'mpSeq'  
getProb(obj)  
  
## S4 method for signature 'pbrSeq'  
getProb(obj)  
  
## S4 method for signature 'rarSeq'  
getProb(obj)
```

```
## S4 method for signature 'tbdSeq'  
getProb(obj)  
  
## S4 method for signature 'udSeq'  
getProb(obj)
```

Arguments

obj object of a class inheriting from `randSeq`. Formal representation of a randomization sequences together with the parameters that belong to the procedure that generated the sequences.

Examples

```
myPar <- bsdPar(10, 2)  
M <- genSeq(myPar, 2)  
getProb(M)  
  
# All Sequences  
par <- pbrPar(bc=c(2,2))  
refSet <- getAllSeq(myPar)  
probs <- getProb(refSet)  
  
# Sequences with probabilities  
cbind(probs, refSet$M)
```

`getRandomizationList` *Accessor function for the randomization list*

Description

Get the randomization list coded in its groups.

Usage

```
getRandList(obj)
```

Arguments

obj object specifying the randomization procedure, see [randPar](#) or [createParam](#).

Examples

```
myPar <- bsdPar(10, 2)  
M <- genSeq(myPar, 2)  
getRandList(M)
```

hadaPar

Representing Hadamard Randomization

Description

Represents the randomization procedure Hadamard Randomization.

Usage

```
hadaPar(N, groups = LETTERS[1:2])
```

Arguments

N	integer for the total sample size of the trial.
groups	character vector of labels for the different treatments.

Details

Hadamard randomization has been proposed by R.A. Bailey. The key idea is to use the columns of a special Hadamard Matrix as a randomization scheme. The implemented algorithm uses the Hadamard Matrix with N=12 columns proposed in the paper, see references.

Value

S4 object of the class hadaPar.

Note

getProb and getAllSeq are currently only supported for hadaPar with total sample size N=12.

References

R.A. Bailey and P.R. Nelson (2003) Hadamard Randomization: A valid restriction of random permuted blocks. *Biometrical Journal*, **45**, 554-60.

See Also

Other randomization procedures: [abcdPar](#), [bbcdPar](#), [bsdPar](#), [chenPar](#), [crPar](#), [createParam](#), [ebcPar](#), [gbcdPar](#), [mpPar](#), [pbrPar](#), [rarPar](#), [rpbrPar](#), [rtbdPar](#), [tbdPar](#), [udPar](#)

hasAllGroups	<i>Check function for occurrence of all treatment groups in the sequence</i>
--------------	--

Description

checks whether each group has its value coming up at least once in the sequence

Usage

```
hasAllGroups(seq, K)
```

Arguments

seq	randomization sequence as inverted matrix
K	number of treatment arms

Value

TRUE if all groups represented, FALSE otherwise

hatMatrix	<i>Calculate hat matrix</i>
-----------	-----------------------------

Description

Calculate hat matrix

Usage

```
hatMatrix(M)
```

Arguments

M	Design Matrix representing the randomization sequence
---	---

Value

Hat matrix, i.e. $M * (M^T M)^{-1} M^T$

`imbal`*Representing the allocation imbalance*

Description

Represents the imbalance of the treatment assignments of patients in a clinical trial.

Usage

```
imbal(type)
```

Arguments

`type` character string, should be one of "imb", "absImb", "loss", or "maxImb", see Details.

Details

Balance of the treatment assignment of patients can be an issue in the design of a clinical trial. The `imbal` function is a constructor function for an S4 object of class `imbal` representing the issue of imbalance of a clinical trial. The parameter `type` can take the following values: The type

"imb" the final imbalance, i.e. difference in group sizes at the end of a trial

"absImb" the absolute value of the final imbalance

"loss" the loss in power estimation, i.e. imb^2/N

"maxImb" the maximal attained imbalance during the trial

Value

S4 object of class `imbal`, a formal representation of the issue of imbalance in a clinical trial.

References

A.C. Atkinson (2014) Selecting a biased coin design. *Statistical Science*, **29**, Vol. 1, 144-163.

See Also

Other issues: [chronBias](#), [combineBias](#), [corGuess](#), [issue](#), [selBias](#), [setPower](#)

Description

Summarizes the criteria for the assessment of randomization procedures.

Details

Randomization in clinical trials is supposed to control certain properties in clinical trials. In the `randomizeR` package, these properties are called `issues`. It is crucial to decide which of the issues is relevant in the present clinical trial, because a randomization procedure that manages well one issue might behave very badly for another. The issues include

- **Selection bias** can occur if future treatment allocations are predictable due to restricted randomization and unmasking of past treatment assignments. The influence of selection bias on the test decision is represented by the `selBias` class. The measure for the predictability of a randomization procedure is implemented in the `corGuess` class representing the expected number of correct guesses.
- **Chronological bias** can occur if a time trend is present in the data. Time trends occur due to learning curves, relaxed inclusion/ exclusion criteria or new co-medication. Chronological bias is represented by the `chronBias` class.
- **Additive combination of chronological and selection bias** may occur if a time trend and selection bias are present in the data. The combined bias is represented by the `combineBias` class.
- **Balance** is important in order to ensure proper power estimation properties of the treatments. However, a high degree of balance favours selection bias. Depending on the clinical context, a randomization procedure should be chosen that admits a suitable imbalance. Imbalance bias is represented by the `imbal` class. The power loss due to imbalance can be assessed directly via the `setPower` class

See Also

Representation of randomization procedures: `randPar`

Generation of randomization sequences: `genSeq`

Assessment of randomization sequences: `assess`

Comparison of randomization sequences: `compare`

Other issues: `chronBias`, `combineBias`, `corGuess`, `imbal`, `selBias`, `setPower`

K	<i>Function returning the total sample size slot of an S4 object</i>
---	--

Description

Function returning the total sample size slot of an S4 object

Usage

K(obj)

Arguments

obj object of class randPar

lambda1	<i>Calculate first non centrality parameter of the doubly non central F-distribution</i>
---------	--

Description

Calculate first non centrality parameter of the doubly non central F-distribution

Usage

lambda1(H, EY)

Arguments

H Hat Matrix
EY (Biased) expectation of the responses

Value

First non centrality parameter, a single numeric value

lambda2	<i>Calculate second non centrality parameter of the doubly non central F-distribution</i>
---------	---

Description

Calculate second non centrality parameter of the doubly non central F-distribution

Usage

```
lambda2(H, EY)
```

Arguments

H	Hat Matrix
EY	(Biased) expectation of the responses

Value

Second non centrality parameter, a single numeric value

makeBiasedExpectation	<i>Calculate Expectation vector</i>
-----------------------	-------------------------------------

Description

Calculate Expectation vector

Usage

```
makeBiasedExpectation(R, mu, bias)
```

Arguments

R	randomization sequence, an object of type randSeq
mu	vector of length K containing the expectation values of groups 1, ..., K
bias	selection or chronological bias object, containing eta(or theta) and alpha

Value

vector of length N with the biased expectation for each patient.

makeDesignMatrix	<i>Design Matrix</i>
------------------	----------------------

Description

Calculate Design Matrix from randomization sequence

Usage

```
makeDesignMatrix(R)
```

Arguments

R randomization sequence, object of type randSeq

Value

makeDesignMatrix converts the randomization sequence R to its Matrix form. The resulting matrix has K columns, one for each treatment group, and N rows, one for each subject. If a subject i is randomized to a certain treatment j, the entry of (i, j) of the matrix will be one, and all other entries in this row will be zero.

method	<i>Function returning the allocation ratio slot of an S4 object</i>
--------	---

Description

Function returning the allocation ratio slot of an S4 object

Usage

```
method(obj)
```

Arguments

obj object of class randPar

mpPar *Representing Maximal Procedure*

Description

Represents the Maximal Procedure.

Usage

```
mpPar(N, mti, ratio = c(1, 1), groups = LETTERS[1:2])
```

Arguments

N	integer for the total sample size of the trial.
mti	maximum tolerated imbalance in patient numbers during the trial.
ratio	vector of length K. The total sample number N and all used block lengths (bc) have to be divisible by sum(ratio).
groups	character vector of labels for the different treatments.

Details

Fix the total sample size N and the mti. Afterwards, the patients are assigned to each treatment arm according to the ratio. All randomization sequences are equiprobable.

Value

S4 object of the class mpPar.

References

V.W. Berger, A. Ivanova and M.D. Knoll (2003) Minimizing predictability while retaining balance through the use of less restrictive randomization procedures. *Statistics in Medicine*, **19**, 3017-28.

See Also

Other randomization procedures: [abcdPar](#), [bbcdPar](#), [bsdPar](#), [chenPar](#), [crPar](#), [createParam](#), [ebcPar](#), [gbcdPar](#), [hadaPar](#), [pbrPar](#), [rarPar](#), [rpbrPar](#), [rtbdPar](#), [tbdPar](#), [udPar](#)

mti	<i>Function returning the MTI slot of an S4 object</i>
-----	--

Description

Function returning the MTI slot of an S4 object

Usage

mti(obj)

Arguments

obj object of class bsdPar or mpPar

mu	<i>Access the expectation value slot of a normEndp S4 object</i>
----	--

Description

Access the expectation value slot of a normEndp S4 object

Usage

mu(obj)

Arguments

obj object of class normEndp

N	<i>Function returning the sample size slot of an S4 object</i>
---	--

Description

Function returning the sample size slot of an S4 object

Usage

N(obj)

Arguments

obj object inheriting from randPar

normEndp	<i>Representation of normally distributed endpoints</i>
----------	---

Description

Represents normally distributed endpoints in clinical trials.

Usage

```
normEndp(mu, sigma)
```

Arguments

mu	vector of the expected responses of the treatment groups, should have length K (i.e. one entry for each treatment group).
sigma	vector of the standard deviations in each the treatment group, should have length K (i.e. one entry for each treatment group).

Details

The normEnd function is a constructor function for an S4 object of the class normEnd representing a normally distributed endpoint in a clinical trial. In conjunction with the assess function, normal endpoints admit the calculation of the exact type-I-error probability and power.

See Also

Compute exact or simulated type-I-error: [assess](#).

overview	<i>Overview over the parameters used in the randomizeR package</i>
----------	--

Description

This list of parameters yields a comprehensive overview of the parameters used in the randomizeR package.

Arguments

add	integer representing the number of balls that are added to the urn in each step.
alpha	the level of the t.test in each simulation.
bc	vector which contains the lengths k_1, \dots, k_l of each block. This means that the vector bc will have one entry for each block.
b	numeric vector of length at most 2 specifying the weight(s) for the punishment of

<code>ini</code>	integer representing the initial urn composition.
<code>compr</code>	factor of compression for the sigmoid-time trend.
<code>d</code>	effect size.
<code>delta</code>	first noncentrality parameter of the doubly noncentral t-distribution.
<code>df</code>	degrees of freedom (i.a. $N-2$).
<code>eta</code>	numeric specifying the magnitude of selection bias.
<code>file</code>	A connection, or a character string naming the file to write to.
<code>filledBlock</code>	logical whether the last block should be filled or not.
<code>FTI</code>	final tolerated imbalance. This is the difference in number of patients of groups A and B that is permitted at the end of a trial. Usually this is set to zero.
<code>groups</code>	character vector of labels for the different treatments.
<code>k</code>	length of the block to be permuted. <code>k</code> should be divisible by the number of treatment arms.
<code>K</code>	number of treatment groups (e.g. $K=2$ if we compare one experimental against one control treatment).
<code>lb</code>	lower bound for the starting value of the poisson distribution.
<code>lambda</code>	(second) noncentrality parameter of the doubly noncentral t-distribution.
<code>method</code>	method that is used to generate the (random) allocation sequence. It can take values PBR, RAR, HAD, PWR, EBC, BSD, CR, TBD, UD, and MP.
<code>mti</code>	maximum tolerated imbalance in patient numbers during the trial.
<code>N</code>	integer for the total sample size of the trial.
<code>name</code>	name of a variable.
<code>mu</code>	vector of the expected responses of the treatment groups, should have length K (i.e. one entry for each treatment group).
<code>obj</code>	object specifying the randomization procedure, see randPar or createParam .
<code>object</code>	any R object.
<code>p</code>	success probability of the biased coin (e.g. in Efron's Biased Coin Design).
<code>pr</code>	vector with patient responses, i.e. each patients resulting value after the treatment.
<code>q</code>	"cut-off" value in $[0.5, 1]$. This is the ratio of patients up from which the experimenter imposes selection bias on the data.
<code>r</code>	numeric indicating the number of random sequences to be generated at random, or missing.
<code>ratio</code>	vector of length K . The total sample number N and all used block lengths (<code>bc</code>) have to be divisible by <code>sum(ratio)</code> .
<code>rb</code>	block lengths of the blocks that can be selected equiprobable at random.
<code>rsob</code>	randomization sequence (of one block).
<code>rs</code>	randomization sequence (of all blocks).
<code>S</code>	matrix for the computation of the probabilities in the maximal procedure.

saltus	integer or missing specifying the patient index (i.e. position) of the step in case of step time trend.
seed	a single value, interpreted as an integer, that specifies the seed for the random number generation.
sigma	vector of the standard deviations in each the treatment group, should have length K (i.e. one entry for each treatment group).
SLs	numeric vector of length at most 2 specifying the lower and/or upper specified border.
theta	factor of the time trend for further details see type.
type	character vector indicating which biasing strategy the experimenter is using (selection bias) and which other bias is present in the clinical trial (e.g. time trend). All biases included in the vector are combined (i.e. added up) to form the total bias. Possible values are "none" (if no bias occurs), "CS" (resp. "DS") (if the experimenter uses the convergence (resp. divergence) strategy to invoke selection bias), LinT for linear time trend, LogT for log-linear time trend, StepT for step time trend, SigT for sigmoid time trend, PWR for knowledge of all up to the first observation in each block, MTI the next observation after reaching the maximal tolerated imbalance is reached will be known to the physician.
TV	numeric specifying the optimal desired value called the target value.
varEq	logical parameter for the t.test: Shall the variances be treated as equal (TRUE=t.test) or different (FALSE= Welch.test).
ub	upper bound for the last value of the poisson distribution.
x	a variable x.
rho	nonnegative parameter which may be adjusted according to how strongly it is desired to balance the experiment. If $\rho = 1$, we have Wei's urn design with $\alpha = 0$. If $\rho = 0$, we have complete randomization.
a	nonnegative parameter which may be adjusted according to how strongly it is desired to balance the experiment. $a = 0$ gives the complete randomization, while the assignments become more deterministic as a increases.
a	nonnegative parameter which controls the degree of randomness: For decreasing a the allocations become deterministic, while for increasing a this procedure tends to complete randomization.

pbrPar

Representing Permuted Block Randomization

Description

Represents the randomization procedure Permuted Block Randomization.

Usage

```
pbrPar(bc, K = 2, ratio = rep(1, K), groups = LETTERS[1:K])
```

Arguments

bc	vector which contains the lengths k_1, \dots, k_l of each block. This means that the vector bc will have one entry for each block.
K	number of treatment groups (e.g. $K=2$ if we compare one experimental against one control treatment).
ratio	vector of length K. The total sample number N and all used block lengths (bc) have to be divisible by <code>sum(ratio)</code> .
groups	character vector of labels for the different treatments.

Details

Fix the block constellation bc, the number of treatment groups K, and the vector of the ratio. Afterwards, in each block the patients are assigned according to the ratio to the corresponding treatment groups. All generated randomization sequences are equiprobable.

Value

S4 object of the class pbrPar.

References

W. F. Rosenberger and J. M. Lachin (2002) *Randomization in Clinical Trials*. Wiley.

See Also

Other randomization procedures: [abcdPar](#), [bbcdPar](#), [bsdPar](#), [chenPar](#), [crPar](#), [createParam](#), [ebcPar](#), [gbcdPar](#), [hadaPar](#), [mpPar](#), [rarPar](#), [rpbrPar](#), [rtbdPar](#), [tbdPar](#), [udPar](#)

plot	<i>Generic plotting of comparison objects</i>
------	---

Description

Generic plotting of comparison objects

Usage

```
plot(x, y, ...)
```

```
## S4 method for signature 'comparison,character'
```

```
plot(x, y)
```

```
## S4 method for signature 'comparison,missing'
```

```
plot(x, y)
```

Arguments

x	object of class comparison.
y	character "boxplot", or "violin", or "missing".
...	"missing"

Details

Creates a box- or violinplot of an object x of the class comparison.

Value

A plot created with the additional package ggplot2.

See Also

[compare](#) for creating S4 objects of the class comparison

Examples

```
# compare Random Allocation Rule and Big Stick for N = 4
# with respect to the correct guesses
RAR <- getAllSeq(rarPar(4))
BSD <- getAllSeq(bsdPar(4, mti = 2))
corGuess <- corGuess("CS")
comp <- compare(corGuess, RAR, BSD)
plot(comp)
```

plotDes

desScore plotting

Description

Plot of an desScore object.

Usage

```
plotDes(desScore, labels, colAv = "red", quantiles = FALSE)
```

Arguments

desScore	object of type desScore.
labels	labels used in the plot. Can be NULL.
colAv	color of the line representing the average of the desirability scores in the plot.
quantiles	logical whether the quantiles should be depicted in the plot.

See Also

Other desirability topics: [derFunc](#), [evaluate](#), [getDesScores](#), [plotEv](#), [probUnDes](#)

Examples

```
# Compute the desirability scores of the full set of PBR(4)
sequences <- getAllSeq(rarPar(4))
issue1 <- corGuess("CS")
issue2 <- chronBias("linT", 1/4, "exact")
endp <- normEndp(mu = c(0,0), sigma = c(1,1))
A <- assess(sequences, issue1, issue2, endp = endp)
d1 <- derFunc(0.5, 0.75, 1)
d2 <- derFunc(0.05, 0.1, 1)

D <- getDesScores(A, d1, d2)
summary(D)
plotDes(D)
plotDes(D, quantiles = TRUE)
```

plotEv

Evaluation plotting

Description

Plot of an evaluation object.

Usage

```
plotEv(evaluation, labels, cols)
```

Arguments

evaluation	object of type evaluation.
labels	labels used in the plot. Can be NULL.
cols	colors of the lines representing the desirability scores in the plot. Can be NULL.

See Also

Other desirability topics: [derFunc](#), [evaluate](#), [getDesScores](#), [plotDes](#), [probUnDes](#)

Examples

```

# Compare Random Allocation Rule to Big Stick Design with respect to different issues
# and their corresponding desirability functions
issue1 <- corGuess("CS")
issue2 <- chronBias(type = "linT", theta = 1/4, method = "exact")
RAR <- getAllSeq(rarPar(4))
BSD <- getAllSeq(bsdPar(4, mti = 2))
A1 <- assess(RAR, issue1, issue2, endp = normEndp(c(0,0), c(1,1)))
A2 <- assess(BSD, issue1, issue2, endp = normEndp(c(0,0), c(1,1)))

d1 <- derFunc(TV = 0.5, 0.75, 2)
d2 <- derFunc(0.05, c(0, 0.1), c(1, 1))
DesScore <- getDesScores(A1, d1, d2, weights = c(5/6, 1/6))
DesScore2 <- getDesScores(A2, d1, d2, weights = c(5/6, 1/6))

E <- evaluate(DesScore, DesScore2)
plotEv(E)

```

plotSeq

Sequence plotting

Description

Plot all randomization sequences of a randSeq object

Usage

```
plotSeq(sequences, plotAllSeq = FALSE, emph = NA, rs = NA)
```

Arguments

sequences	object of type randSeq
plotAllSeq	logical. If plotAllSeq=TRUE, the complete set of randomization sequences will be plotted in light gray.
emph	integer indicating which sequence should be highlighted in blue.
rs	vector of a randomization sequence that should be highlighted.

`probUnDes`*Computing the probability of having desirability scores of zero*

Description

Computing the probability of having desirability scores of zero for each desirability function applied to an issue.

Usage

```
probUnDes(desScore)
```

```
## S4 method for signature 'desScores'  
probUnDes(desScore)
```

Arguments

`desScore` an object of the class `desScores`, i.e. an object resulting from applying the function [getDesScores](#)

Details

The function `probUnDes` expects an object that results from the [getDesScores](#) function. For each issue it computes the probability that it achieves an undesirable score, i.e. a desirability score of 0. In doing so, it weights the zero desirability scores with the probability that the sequence occurs.

Value

S4 object of class `probUnDesirable` computing the probability of getting undesirable scores, i.e. desirability scores of 0.

See Also

Representation of randomization procedures: [randPar](#)

Generation of randomization sequences: [genSeq](#)

[issues](#) for the desirability of randomization sequences

Other desirability topics: [derFunc](#), [evaluate](#), [getDesScores](#), [plotDes](#), [plotEv](#)

Examples

```
# Compare Random Allocation Rule to Big Stick Design with respect to different issues  
# and their corresponding desirability functions  
RAR <- getAllSeq(randPar(4))  
issue1 <- corGuess("CS")  
issue2 <- corGuess("DS")  
A1 <- assess(RAR, issue1, issue2)
```



```
d1 <- derFunc(TV = 0.1, 0.7, 2)
d2 <- derFunc(0.5, c(0.3, 0.8), c(1, 1))
DesScore <- getDesScores(A1, d1, d2, weights = c(5/6, 1/6))

probUnDes(DesScore)
```

randBlocks	<i>Function returning the block slot of an S4 object</i>
------------	--

Description

Function returning the block slot of an S4 object

Usage

```
randBlocks(obj)
```

Arguments

obj	object of class pbrPAR
-----	------------------------

randPar	<i>Settings for randomization procedures</i>
---------	--

Description

Randomization procedures in `randomizeR` are represented by objects that inherit from `randPar`. The representation can then be used in order to generate randomization sequences. In order generate a representation of a randomization procedure, call `createParam` or one of the following functions.

Supported randomization procedures

- Complete Randomization (`crPar`)
- Random Allocation Rule (`rarPar`)
- Permuted Block Randomization (`pbrPar`)
- Permuted Block Randomization with random block length (`rpbrPar`)
- Truncated Binomial Design (`tbdPar`)
- Truncated Binomial Design with random block length (`rtbdPar`)
- Efron's Biased Coin Design (`ebcPar`)
- Big Stick Design (`bsdPar`)
- Maximal Procedure (`mpPar`)

- Wei's Urn Design ([udPar](#))
- Chen's Design ([chenPar](#))
- Generalized Biased Coin Design ([gbcdPar](#))
- Accelerated Biased Coin Design ([abcdPar](#))
- Bayesian Biased Coin Design ([bbcdPar](#))
- Hadamard Randomization ([hadaPar](#))

See Also

Generate randomization sequences [genSeq](#). Calculate the the complete set of randomization sequences of a randomization procedure. [getAllSeq](#).

randPar-class

Randomization paramters

Description

Randomization paramters

randSeq-class

An S4 Class for the representation of randomization sequences

Description

This set of classes provides functionality of storing randomization sequences of different randomization procedures along with the parameters representing the design.

Slots

N total number of patients included in the trial

M matrix containing randomization sequences of length N in its rows.

K number of treatment groups

groups character string of length K defining the names of the treatment groups

rarPar *Representing Random Allocation Rule*

Description

Represents the randomization procedure Random Allocation Rule.

Usage

```
rarPar(N, K = 2, ratio = rep(1, K), groups = LETTERS[1:K])
```

Arguments

N	integer for the total sample size of the trial.
K	number of treatment groups (e.g. K=2 if we compare one experimental against one control treatment).
ratio	vector of length K. The total sample number N and all used block lengths (bc) have to be divisible by <code>sum(ratio)</code> .
groups	character vector of labels for the different treatments.

Details

Fix a total sample size N the number of treatment groups K, and the vector of the ratio. Afterwards, all patients are assigned according to the ratio to the corresponding treatment groups. All randomization sequences are equiprobable.

Value

S4 object of the class rarPar.

References

W. F. Rosenberger and J. M. Lachin (2002) *Randomization in Clinical Trials*. Wiley.

See Also

Other randomization procedures: [abcdPar](#), [bbcdPar](#), [bsdPar](#), [chenPar](#), [crPar](#), [createParam](#), [ebcPar](#), [gbcdPar](#), [hadaPar](#), [mpPar](#), [pbrPar](#), [rpbrPar](#), [rtbdPar](#), [tbdPar](#), [udPar](#)

ratio	<i>Function returning the allocation ratio slot of an S4 object</i>
-------	---

Description

Function returning the allocation ratio slot of an S4 object

Usage

```
ratio(obj)
```

Arguments

obj	object of class randPar
-----	-------------------------

rho	<i>Function returning the adjusting parameter rho slot of an S4 object</i>
-----	--

Description

Function returning the adjusting parameter rho slot of an S4 object

Usage

```
rho(obj)
```

Arguments

obj	object of class randPar
-----	-------------------------

rpbrPar	<i>Representing Randomized Permuted Block Randomization</i>
---------	---

Description

Represents the randomization procedure Randomized Permuted Block Randomization.

Usage

```
rpbrPar(N, rb, K = 2, ratio = rep(1, K), groups = LETTERS[1:K],  
filledBlock = FALSE)
```

Arguments

N	integer for the total sample size of the trial.
rb	block lengths of the blocks that can be selected equiprobable at random.
K	number of treatment groups (e.g. K=2 if we compare one experimental against one control treatment).
ratio	vector of length K. The total sample number N and all used block lengths (bc) have to be divisible by <code>sum(ratio)</code> .
groups	character vector of labels for the different treatments.
filledBlock	logical whether the last block should be filled or not.

Details

Fix the possible random block lengths `rb`, the number of treatment groups `K`, the sample size `N` and the vector of the `ratio`. Afterwards, one block length is randomly selected of the random block lengths. The patients are assigned according to the ratio to the corresponding treatment groups. This procedure is repeated until `N` patients are assigned. Within each block all possible randomization sequences are equiprobable.

Value

S4 object of the class `rpbrPar`.

References

W. F. Rosenberger and J. M. Lachin (2002) *Randomization in Clinical Trials*. Wiley.

See Also

Other randomization procedures: [abcdPar](#), [bbcdPar](#), [bsdPar](#), [chenPar](#), [crPar](#), [createParam](#), [ebcPar](#), [gbcPar](#), [hadaPar](#), [mpPar](#), [pbrPar](#), [rarPar](#), [rtbdPar](#), [tbdPar](#), [udPar](#)

rtbdPar

Representing Randomized Truncated Binomial Design

Description

Represents the randomization procedure Randomized Truncated Binomial Design.

Usage

```
rtbdPar(N, rb = N, groups = LETTERS[1:2], filledBlock = FALSE)
```

Arguments

N	integer for the total sample size of the trial.
rb	block lengths of the blocks that can be selected equiprobable at random.
groups	character vector of labels for the different treatments.
filledBlock	logical whether the last block should be filled or not.

Details

Fix the possible random block lengths `rb` and the sample size of the trial `N`. Afterwards, one block length is randomly selected of the random block lengths. In this block a fair coin is tossed for the patient assignments until half of the patients have been assigned to one of the treatment arms. Afterwards, the block is filled with the other treatment. This procedure is repeated until `N` patients are assigned.

Value

S4 object of the class `rtbdPar`.

References

W. F. Rosenberger and J. M. Lachin (2002) *Randomization in Clinical Trials*. Wiley.

See Also

Other randomization procedures: [abcdPar](#), [bbcdPar](#), [bsdPar](#), [chenPar](#), [crPar](#), [createParam](#), [ebcPar](#), [gbcdPar](#), [hadaPar](#), [mpPar](#), [pbrPar](#), [rarPar](#), [rpbrPar](#), [tbdPar](#), [udPar](#)

saveAssess

Saving an assess object

Description

Saves the full information of an `assess` object in a `.csv` data sheet.

Usage

```
saveAssess(obj, file = "assessObject.csv")
```

Arguments

<code>obj</code>	object of class <code>assessmentment</code> , e.g. the output of the assess function.
<code>file</code>	A connection, or a character string naming the file to write to.

Value

Creates a `.csv` data in the home folder.

See Also

Other saving functions: [saveRand](#)

saveRand	<i>Saving a randomization lists</i>
----------	-------------------------------------

Description

Saves the parameters of a randSeq object in a .csv data sheet.

Usage

```
saveRand(obj, file = "randList.csv")
```

Arguments

obj	object of class randSeq containing a single randomization sequence.
file	A connection, or a character string naming the file to write to.

Value

Creates a .csv data in the home folder and saves the randomization list as a column vector.

See Also

Other saving functions: [saveAssess](#)

seed	<i>Function returning the allocation seed slot of an object</i>
------	---

Description

Returns the seed that was either generated at random or user specified. The seed can be specified for any random operation e.g. genSeq.

Usage

```
seed(obj)
```

Arguments

obj	object specifying the randomization procedure, see randPar or createParam .
-----	---

selBias	<i>Representing selection bias</i>
---------	------------------------------------

Description

Represents the issue of selection bias in a clinical trial.

Usage

```
selBias(type, eta, method, alpha = 0.05)
```

Arguments

type	character string, should be one of "CS", "CS2" or "DS", see Details.
eta	numeric specifying the magnitude of selection bias.
method	character string, should be one of "sim" or "exact", see Details.
alpha	significance level.

Details

Selection bias can be an issue in the design of a clinical trial. The selBias function is a constructor function for an S4 object of the class selBias representing the issue of third order selection bias in a clinical trial. It supports two possible modes, method="sim" and method="exact". This representation is particularly useful in interaction with the [assess](#) function.

method="sim" Represents the simulated type-I-error rate given the level alpha, the selection effect eta and the biasing strategy type. When calling assess for a selBias object with method="sim", one test decision is computed for each sequence of randSeq. The type-I-error rate (power) is the proportion of falsely (correctly) rejected null hypotheses.

method="exact" Represents the exact type-I-error probability given the level alpha, the selection effect eta and the biasing strategy type. When calling assess for a selBias object with method="exact", the exact p -value of each randomization sequence is computed. So far, this is only supported for normal endpoints. Then the type-I-error probability is the sum of the corresponding quantiles of the doubly noncentral t-distribution.

Value

S4 object of class selBias, a formal representation of the issue of selection bias in a clinical trial.

References

- D. Blackwell and J.L. Hodges Jr. (1957) Design for the control of selection bias. *Annals of Mathematical Statistics*, **25**, 449-60.
- M. Proschan (1994) Influence of selection bias on the type-I-error rate under random permuted block designs. *Statistica Sinica*, **4**, 219-31.

See Also

Compute exact or simulated type-I-error: [assess](#).

Other issues: [chronBias](#), [combineBias](#), [corGuess](#), [imbal](#), [issue](#), [setPower](#)

setPower

Representing the power

Description

Represents the expected power of the individual randomization sequences.

Usage

```
setPower(d, method, alpha = 0.05)
```

Arguments

d	effect size.
method	character string, should be one of "sim" or "exact", see Description.
alpha	significance level.

Details

The attained power of an individual randomization sequence can be an issue in the design of a clinical trial. The power of a randomization sequence is computed dependent on the effect size d and the difference in group sizes in the end if.

If `method="sim"`, the object represents the simulated power of an individual randomization sequence. When calling `assess` for a power object with `method="sim"`, one test decision is computed for each randomization sequence of `randSeq`. The power is the proportion of falsely (correctly) rejected null hypotheses.

If `method="exact"`, the object represents the exact power of an individual randomization sequence. When calling `assess` for a power object with `method="exact"`, the exact p -value of each randomization sequence is computed. So far, this is only supported for normal endpoints. Then the power is the sum of the corresponding quantiles of the noncentral t -distribution.

Value

S4 object of class `power`, a formal representation of the issue of power in a clinical trial.

See Also

Other issues: [chronBias](#), [combineBias](#), [corGuess](#), [imbal](#), [issue](#), [selBias](#)

sigma	<i>Function returning the standard deviation slot of a normEndp S4 object</i>
-------	---

Description

Function returning the standard deviation slot of a normEndp S4 object

Usage

```
sigma(obj)
```

Arguments

obj	object of class normEndp
-----	--------------------------

summary	<i>Summary of assessments of a randomization procedure</i>
---------	--

Description

Summary of assessments of a randomization procedure

Summary of desirability scores of a randomization procedure

Usage

```
summary(object, ...)
```

```
## S4 method for signature 'assessment'
summary(object)
```

```
summary(object, ...)
```

```
## S4 method for signature 'desScores'
summary(object)
```

Arguments

object	assessment object.
--------	--------------------

...	additional arguments affecting the summary that will be produced.
-----	---

Details

For each issue the assessment of the sequences is summarized to permit a design-based assessment of the randomization procedure. This approach uses the sequence-wise values of the assessment and the probabilities in order to give an overall summary.

For each issue the desirability score of the sequences is summarized to permit a design-based desirability score of the randomization procedure. This approach uses the sequence-wise values of the desirability and the probabilities in order to give an overall summary.

Value

Data frame with a summary of the assessment object.

Data frame with a summary of the desirability scores object.

Examples

```
# assess the full set of PBR(4)
seq <- getAllSeq(pbrPar(4))
issue <- corGuess("CS")
A <- assess(seq, issue)
summary(A)

# Compute the desirability scores of the full set of PBR(4)
seq <- getAllSeq(pbrPar(4))
issue1 <- corGuess("CS")
issue2 <- corGuess("DS")
A <- assess(seq, issue1, issue2)
d1 <- derFunc(0.5, c(0.1, 0.8), c(1, 1))
d2 <- derFunc(0.1, 0.7, 2)
D <- getDesScores(A, d1, d2, weights = c(5/6, 1/6))
summary(D)
```

tbdPar

Representing Truncated Binomial Design

Description

Represents the Truncated Binomial Design.

Usage

```
tbdPar(bc = N, groups = LETTERS[1:2])
```

Arguments

bc vector which contains the lengths k_1, \dots, k_l of each block. This means that the vector **bc** will have one entry for each block.

groups character vector of labels for the different treatments.

Details

A fair coin is tossed until half of the patients have been assigned to one of the treatment arms. Afterwards, the randomization list is filled with the other treatment.

Value

S4 object of the class `tbdPar`.

References

W. F. Rosenberger and J. M. Lachin (2002) *Randomization in Clinical Trials*. Wiley.

See Also

Other randomization procedures: [abcdPar](#), [bbcdPar](#), [bsdPar](#), [chenPar](#), [crPar](#), [createParam](#), [ebcPar](#), [gbcdPar](#), [hadaPar](#), [mpPar](#), [pbrPar](#), [rarPar](#), [rpbrPar](#), [rtbdPar](#), [udPar](#)

TV	<i>Function returning the sample size slot of an S4 object</i>
----	--

Description

Function returning the sample size slot of an S4 object

Usage

TV(obj)

Arguments

obj object inheriting from `derFunc`

type	<i>Get type of an object</i>
------	------------------------------

Description

Accesses the type slot of an S4 object

Usage

type(obj)

Arguments

obj a bias object (i.e. S4 object inheriting from `bias`)

Value

Character string specifying the type of bias obj represents, e.g. "linT" in case of chronological bias.

 udPar

Representing Wei's Urn Design

Description

Represents Wei's Urn Design.

Usage

```
udPar(N, ini, add, groups = LETTERS[1:2])
```

Arguments

N	integer for the total sample size of the trial.
ini	integer representing the initial urn composition.
add	integer representing the number of balls that are added to the urn in each step.
groups	character vector of labels for the different treatments.

Details

An urn is filled with a number of `ini` balls of both of the treatments. Afterwards, a ball is drawn randomly from the urn. Finally, `add` balls are added to the urn from the opposite treatment. This procedure is repeated until `N` patients are assigned.

Value

S4 object of the class `udPar`.

References

L.J. Wei (1977) A Class of Designs for Sequential Clinical Trials. *Journal of the American Statistical Association*, **72**, 382-6.

See Also

Other randomization procedures: [abcdPar](#), [bbcdPar](#), [bsdPar](#), [chenPar](#), [crPar](#), [createParam](#), [ebcPar](#), [gbcdPar](#), [hadaPar](#), [mpPar](#), [pbrPar](#), [rarPar](#), [rpbrPar](#), [rtbdPar](#), [tbdPar](#)

\$,assessment-method *Method defining the \$ operator for the assessemnt class*

Description

Method defining the \$ operator for the assessemnt class

Usage

```
## S4 method for signature 'assessment'
x$name
```

Arguments

x	a variable x.
name	name of a variable.

\$,comparison-method *Method defining the \$ operator for the assessemnt class*

Description

Method defining the \$ operator for the assessemnt class

Usage

```
## S4 method for signature 'comparison'
x$name
```

Arguments

x	a variable x.
name	name of a variable.

\$,derFunc-method *Method defining the \$ operator for the derFunc class*

Description

Method defining the \$ operator for the derFunc class

Usage

```
## S4 method for signature 'derFunc'  
x$name
```

Arguments

x a variable x.
name name of a variable.

\$,desFunc-method *Method defining the \$ operator for the desFunc class*

Description

Method defining the \$ operator for the desFunc class

Usage

```
## S4 method for signature 'desFunc'  
x$name
```

Arguments

x a variable x.
name name of a variable.

\$,desScores-method *Method defining the \$ operator for the assessemnt class*

Description

Method defining the \$ operator for the assessemnt class

Usage

```
## S4 method for signature 'desScores'
x$name
```

Arguments

x	a variable x.
name	name of a variable.

\$,endpoint-method *Method defining the \$ operator for the endpoint class*

Description

Method defining the \$ operator for the endpoint class

Usage

```
## S4 method for signature 'endpoint'
x$name
```

Arguments

x	a variable x.
name	name of a variable.

\$,evaluation-method Method defining the \$ operator for the evaluation class

Description

Method defining the \$ operator for the evaluation class

Usage

```
## S4 method for signature 'evaluation'  
x$name
```

Arguments

x	a variable x.
name	name of a variable.

\$,issue-method Method defining the \$ operator for the issue class

Description

Method defining the \$ operator for the issue class

Usage

```
## S4 method for signature 'issue'  
x$name
```

Arguments

x	a variable x.
name	name of a variable.

\$,probUnDesirable-method

Method defining the \$ operator for the probUnDesirable class

Description

Method defining the \$ operator for the probUnDesirable class

Usage

```
## S4 method for signature 'probUnDesirable'
x$name
```

Arguments

x	a variable x.
name	name of a variable.

\$,randPar-method

Method defining the \$ operator for the randPar class

Description

Method defining the \$ operator for the randPar class

Usage

```
## S4 method for signature 'randPar'
x$name
```

Arguments

x	a variable x.
name	name of a variable.

`$.randSeq-method` *Method defining the \$ operator for the randSeq class*

Description

Method defining the \$ operator for the randSeq class

Usage

```
## S4 method for signature 'randSeq'  
x$name
```

Arguments

x	a variable x.
name	name of a variable.

Index

- \$, assessment-method, 78
- \$, comparison-method, 78
- \$, derFunc-method, 79
- \$, desFunc-method, 79
- \$, desScores-method, 80
- \$, endpoint-method, 80
- \$, evaluation-method, 81
- \$, issue-method, 81
- \$, probUnDesirable-method, 82
- \$, randPar-method, 82
- \$, randSeq-method, 83

- a, 5
- abcdPar, 5, 9, 11, 12, 18, 20, 26, 28, 48, 55, 60, 66, 67, 69, 70, 76, 77
- abcdRand, 6
- assess, 5, 7, 44, 51, 57, 70, 72, 73
- assess, randSeq, endpoint-method (assess), 7
- assess, randSeq, missing-method (assess), 7

- bbcdPar, 6, 8, 11, 12, 18, 20, 26, 28, 48, 55, 60, 66, 67, 69, 70, 76, 77
- bbcdRand, 9
- blocks, 10
- bsdPar, 6, 9, 10, 12, 18, 20, 26, 28, 48, 55, 60, 65, 67, 69, 70, 76, 77
- bsdRand, 11

- calcProb (getProbabilities), 46
- calculateProbabilities (getProbabilities), 46
- chenPar, 6, 9, 11, 12, 18, 20, 26, 28, 48, 55, 60, 66, 67, 69, 70, 76, 77
- chenRand, 13
- chronBias, 13, 15, 18, 45, 50, 51, 73
- coin, 15
- combineBias, 15, 15, 18, 50, 51, 73
- compare, 5, 16, 51, 61
- compare, issue, endpoint-method (compare), 16
- compare, issue, missing-method (compare), 16
- corGuess, 15, 17, 50, 51, 73
- createParam, 6, 9, 11, 12, 18, 20, 26, 28, 30, 31, 36, 47, 48, 55, 58, 60, 65, 67, 69–71, 76, 77
- createSeq, 19
- crPar, 6, 9, 11, 12, 18, 19, 26, 28, 48, 55, 60, 65, 67, 69, 70, 76, 77

- derFunc, 20, 21, 27, 43, 44, 62, 64
- derFunc-class, 21
- desirability, 21
- doublyF_opt, 22
- doublyF_value, 23
- doublyF_values, 24
- doublyT, 24
- doublyTValues, 25

- ebcPar, 6, 9, 11, 12, 18, 20, 25, 28, 48, 55, 60, 65, 67, 69, 70, 76, 77
- evaluate, 21, 26, 44, 62, 64
- evaluate, character-method (evaluate), 26
- evaluate, missing-method (evaluate), 26

- gbcdPar, 6, 9, 11, 12, 18, 20, 26, 28, 48, 55, 60, 66, 67, 69, 70, 76, 77
- gbcdRand, 29
- generateAllSequences, 29
- generateRandomSequences, 32
- genNcps, 38
- genSeq, 5, 7, 16, 27, 44, 51, 64, 66
- genSeq (generateRandomSequences), 32
- genSeq, abcdPar, missing, missing-method (generateRandomSequences), 32
- genSeq, abcdPar, missing, numeric-method (generateRandomSequences), 32

- genSeq, rtbdPar, numeric, missing-method
(generateRandomSequences), 32
- genSeq, rtbdPar, numeric, numeric-method
(generateRandomSequences), 32
- genSeq, tbdPar, missing, missing-method
(generateRandomSequences), 32
- genSeq, tbdPar, missing, numeric-method
(generateRandomSequences), 32
- genSeq, tbdPar, numeric, missing-method
(generateRandomSequences), 32
- genSeq, tbdPar, numeric, numeric-method
(generateRandomSequences), 32
- genSeq, udPar, missing, missing-method
(generateRandomSequences), 32
- genSeq, udPar, missing, numeric-method
(generateRandomSequences), 32
- genSeq, udPar, numeric, missing-method
(generateRandomSequences), 32
- genSeq, udPar, numeric, numeric-method
(generateRandomSequences), 32
- getAllSeq, 7, 66
- getAllSeq (generateAllSequences), 29
- getAllSeq, abcdPar-method
(generateAllSequences), 29
- getAllSeq, bbcdPar-method
(generateAllSequences), 29
- getAllSeq, bsdPar-method
(generateAllSequences), 29
- getAllSeq, chenPar-method
(generateAllSequences), 29
- getAllSeq, crPar-method
(generateAllSequences), 29
- getAllSeq, ebcPar-method
(generateAllSequences), 29
- getAllSeq, gbcdPar-method
(generateAllSequences), 29
- getAllSeq, hadaPar-method
(generateAllSequences), 29
- getAllSeq, mpPar-method
(generateAllSequences), 29
- getAllSeq, pbrPar-method
(generateAllSequences), 29
- getAllSeq, rarPar-method
(generateAllSequences), 29
- getAllSeq, tbdPar-method
(generateAllSequences), 29
- getAllSeq, udPar-method
(generateAllSequences), 29
- getbiasCS1, 39
- getbiasCS2, 39
- getCorGuesses, 40
- getDesFunc, 40
- getDesFunc, derringerLs-method
(getDesFunc), 40
- getDesFunc, derringerRs-method
(getDesFunc), 40
- getDesFunc, derringerTs-method
(getDesFunc), 40
- getDesign, 41
- getDesign, abcdPar-method (getDesign), 41
- getDesign, abcdSeq-method (getDesign), 41
- getDesign, bbcdPar-method (getDesign), 41
- getDesign, bbcdSeq-method (getDesign), 41
- getDesign, bsdPar-method (getDesign), 41
- getDesign, bsdSeq-method (getDesign), 41
- getDesign, chenPar-method (getDesign), 41
- getDesign, chenSeq-method (getDesign), 41
- getDesign, crPar-method (getDesign), 41
- getDesign, crSeq-method (getDesign), 41
- getDesign, ebcPar-method (getDesign), 41
- getDesign, ebcSeq-method (getDesign), 41
- getDesign, gbcdPar-method (getDesign), 41
- getDesign, gbcdSeq-method (getDesign), 41
- getDesign, hadaPar-method (getDesign), 41
- getDesign, hadaSeq-method (getDesign), 41
- getDesign, mpPar-method (getDesign), 41
- getDesign, mpSeq-method (getDesign), 41
- getDesign, pbrPar-method (getDesign), 41
- getDesign, pbrSeq-method (getDesign), 41
- getDesign, rarPar-method (getDesign), 41
- getDesign, rarSeq-method (getDesign), 41
- getDesign, rpbrPar-method (getDesign), 41
- getDesign, rRpbrSeq-method (getDesign),
41
- getDesign, rRtbdSeq-method (getDesign),
41
- getDesign, rtbdPar-method (getDesign), 41
- getDesign, tbdPar-method (getDesign), 41
- getDesign, tbdSeq-method (getDesign), 41
- getDesign, udPar-method (getDesign), 41
- getDesign, udSeq-method (getDesign), 41
- getDesScores, 21, 27, 43, 62, 64
- getDesScores, assessment, missing-method
(getDesScores), 43
- getDesScores, assessment, numeric-method
(getDesScores), 43

- getExpectation, 44
- getExpectation, randSeq, chronBias, missing-method (getExpectation), 44
- getExpectation, randSeq, chronBias, normEndp-method (getExpectation), 44
- getExpectation, randSeq, combinedBias, normEndp-method (getExpectation), 44
- getExpectation, randSeq, combinedBiasStepTrend, normEndp-method (getExpectation), 44
- getExpectation, randSeq, missing, normEndp-method (getExpectation), 44
- getExpectation, randSeq, power, normEndp-method (getExpectation), 44
- getExpectation, randSeq, selBias, missing-method (getExpectation), 44
- getExpectation, randSeq, selBias, normEndp-method (getExpectation), 44
- getProb (getProbabilities), 46
- getProb, abcdSeq-method (getProbabilities), 46
- getProb, bbcdSeq-method (getProbabilities), 46
- getProb, bsdSeq-method (getProbabilities), 46
- getProb, chenSeq-method (getProbabilities), 46
- getProb, crSeq-method (getProbabilities), 46
- getProb, ebcSeq-method (getProbabilities), 46
- getProb, gbcdSeq-method (getProbabilities), 46
- getProb, hadaSeq-method (getProbabilities), 46
- getProb, mpSeq-method (getProbabilities), 46
- getProb, pbrSeq-method (getProbabilities), 46
- getProb, rarSeq-method (getProbabilities), 46
- getProb, tbdSeq-method (getProbabilities), 46
- getProb, udSeq-method (getProbabilities), 46
- getProbabilities, 46
- getRandList (getRandomizationList), 47
- getRandomizationList, 47
- hadaPar, 6, 9, 11, 12, 18, 20, 26, 28, 48, 55, 60, 66, 67, 69, 70, 76, 77
- hasAllGroups, 49
- hatMatrix, 49
- imbal, 15, 18, 50, 51, 73
- issues, 5, 7, 16, 27, 44, 64
- issue, 15, 18, 50, 51, 73
- isSupSet, 41
- lambda1, 52
- lambda2, 53
- makeBiasedExpectation, 53
- makeDesignMatrix, 54
- method, 54
- mpPar, 6, 9, 11, 12, 18, 20, 26, 28, 48, 55, 60, 65, 67, 69, 70, 76, 77
- mti, 56
- mu, 56
- N, 56
- normEndp, 45, 57
- overview, 57
- pbrPar, 6, 9, 11, 12, 18, 20, 26, 28, 48, 55, 59, 65, 67, 69, 70, 76, 77
- plot, 60
- plot, comparison, character-method (plot), 60
- plot, comparison, missing-method (plot), 60
- plotDes, 21, 27, 44, 61, 62, 64
- plotEv, 21, 27, 44, 62, 62, 64
- plotSeq, 63
- probUnDes, 21, 27, 44, 62, 64
- probUnDes, desScores-method (probUnDes), 64
- randBlocks, 65
- randomizeR (randomizeR-package), 4
- randomizeR-package, 4
- randPar, 5, 7, 16, 18, 27, 30, 36, 44, 47, 51, 58, 64, 65, 71
- randPar-class, 66
- randSeq, 31, 36
- randSeq-class, 66

rarPar, [6](#), [9](#), [11](#), [12](#), [18](#), [20](#), [26](#), [28](#), [48](#), [55](#), [60](#),
[65](#), [67](#), [69](#), [70](#), [76](#), [77](#)

ratio, [68](#)

rho, [68](#)

rpbrPar, [6](#), [9](#), [11](#), [12](#), [18](#), [20](#), [26](#), [28](#), [48](#), [55](#),
[60](#), [65](#), [67](#), [68](#), [70](#), [76](#), [77](#)

rtbdPar, [6](#), [9](#), [11](#), [12](#), [18](#), [20](#), [26](#), [28](#), [48](#), [55](#),
[60](#), [65](#), [67](#), [69](#), [69](#), [76](#), [77](#)

saveAssess, [70](#), [71](#)

saveRand, [71](#), [71](#)

seed, [71](#)

selBias, [15](#), [18](#), [45](#), [50](#), [51](#), [72](#), [73](#)

setPower, [15](#), [18](#), [50](#), [51](#), [73](#), [73](#)

sigma, [74](#)

summary, [74](#)

summary, assessment-method (summary), [74](#)

summary, desScores-method (summary), [74](#)

tbdPar, [6](#), [9](#), [11](#), [12](#), [18](#), [20](#), [26](#), [28](#), [48](#), [55](#), [60](#),
[65](#), [67](#), [69](#), [70](#), [75](#), [77](#)

TV, [76](#)

type, [76](#)

udPar, [6](#), [9](#), [11](#), [12](#), [18](#), [20](#), [26](#), [28](#), [48](#), [55](#), [60](#),
[66](#), [67](#), [69](#), [70](#), [76](#), [77](#)