Package ‘randomizeR’

March 11, 2016

Title Randomization for Clinical Trials

Version 1.2

Date 2016-03-11

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 (>= 3.2.3),
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'
'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'
'ebcSeq.R'
'gbcdPar.R'
'gbcdSeq.R'
'hadaPar.R'
'hadaSeq.R'
'mpPar.R'
'mpSeq.R'
'pbrPar.R'
'pbrSeq.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,
quantreg

VignetteBuilder knitr

RoxygenNote 5.0.1

R topics documented:

`randomizeR-package` *Randomization for Clinical Trials*

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 IDeAl 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 (2002) *Randomization in Clinical Trials*. Wiley.

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 Adjustable Biased Coin Design*

Description

Represents the randomization procedure Adjustable 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 my 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 my 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

`randSeq`	object of class `randSeq`.
`...`	at least one object of class `issue` or just a list of objects of the class `issue`.
`endp`	object of class `endpoint`, or `missing`.

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 treamtent 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 <- 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 my 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 my 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` *Function returning the block slot of an S4 object*

Description

Function returning the block slot of an S4 object

Usage

```
blocks(obj)
```

Arguments

`obj`	object of class pbrPAr

bsdPar — *Representing Big Stick Design*

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

`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).
`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 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 `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.

If `method="exact"`, the object represents the exact type-I-error proabability 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.

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 exlusion 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` *Function returning the coin slot of an S4 object*

Description

Function returning the coin slot of an S4 object

Usage

```
coin(obj)
```

Arguments

`obj`	object extending class randPar or randSeq

`combineBias` *Combined additive bias criterion*

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

`issue`	object of class `issue`.
`...`	at least one object of class `randSeq` or a list of objects of class `randSeq`.
`endp`	object of class `endpoint`, or `missing`.

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 treamtent 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 occured less often.

`"DS"` refers to "divergence strategy", i.e. the investigator predicts the treatment which has hitherto occured 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)
```

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,...,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.

Details

Dending on the input of the user, createParam creates an object representing a randomization procedures 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 *Query to create a randomization sequence of a particular randomization procedure*

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

`file`	A connection, or a character string naming the file to write to.

Value

an object `Param`, which is available

crPar *Representing Complete Randomization*

Description

Represents the randomization procedure Complete Randomization.

Usage

```
crPar(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 `sum(ratio)`.
`groups`	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`

---

doublyT — *Approximation of the distribution function of the doubly noncentral t-distribution*

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 — *Calculation of the biased type-one-error (resp. power) of Student‘s t-test*

Description

Computes the biased type-one-error (resp. power) of Student‘ts 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 *Representing Efron's Biased Coin Design*

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`

gbcdPar — *Representing Generalized Biased Coin Design*

Description

Represents the randomization procedure Generalized Biased Coin Design.

Usage

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

Arguments

N	integer for the total sample size of the trial.
rho	nonnegative parameter which my 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.
groups	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 — *Sampling algorithm for gbcd*

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 my 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` *Complete set of randomization sequences*

Description

Outputs all randomization sequences for the given randomization procedure along with the parameters belonging to the 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 rows.

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, i.e. an object of a class.

Details

`getAllSeq` is a generic function which dispatches different methods depending on the type of input.

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 a randomization sequences for 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, i.e. an object of a class.
`r`	numeric indicating the number of random sequences to be generated at random or missing.
`seed`	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`.

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 noncentraility 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)
```

getCorGuesses — *Matrix of the guesses of the investigator*

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)
```

---

getDesign *Design of a randomization procedure*

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`.

`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 treatement 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, i.e. an object of a class.

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` 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 represenation 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` `issue` *Issues in clinical trials*

Description

Summarizes the criteria for the assessment of randomization procedures.

Details

Randomization in clinical trials is supposed to control certain issues in clinical trials. Many of the issues are working in opposite direction, so it is crucial to decide for which of the issues is relevant in the present clinical trial. These issues include

- **Selection bias** can occur if future treatment allocations are predictable due to restricted randomization and unmasking of past treatment assigments. Selection bias is represented by the `selBias` class.
- **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.
- **Balance** is important in order to ensure proper power estimation properties of the treatments. However, a high degree of balance favours selection bias. A middle course seems optimal. Imbalance bias is represented by the `imbal` 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` *Function returning the total sample size slot of an S4 object*

Description

Function returning the total sample size slot of an S4 object

Usage

```
K(obj)
```

Arguments

`obj`	object of class randPar

`method` *Function returning the allocation ratio slot of an S4 object*

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 assigend 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` *Function returning the MTI slot of an S4 object*

Description

Function returning the MTI slot of an S4 object

Usage

```
mti(obj)
```

Arguments

`obj`	object of class bsdPar or mpPar

`mu` *Access the expectation value slot of a normEndp S4 object*

Description

Access the expectation value slot of a normEndp S4 object

Usage

```
mu(obj)
```

Arguments

`obj`	object of class normEndp

`N` *Function returning the sample size slot of an S4 object*

Description

Function returning the sample size slot of an S4 object

Usage

```
N(obj)
```

Arguments

`obj`	object inheriting from randPar

normEndp — *Representation of normally distributed endpoints*

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 — *Overview over the parameters used in the* `randomizeR` *package*

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,...,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
`ini`	integer representing the initial urn composition.
`compr`	factor of compression for the sigmoid-time trend.
`d`	effect size.

delta	first noncentrality parameter of the doubly noncentral t-distribution.
df	degrees of freedom (i.a. N-2).
eta	numeric specifying the magnitude of selection bias.
file	A connection, or a character string naming the file to write to.
filledBlock	logical whether the last block should be filled or not.
FTI	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.
groups	character vector of labels for the different treatments.
k	length of the block to be permuted. k should be divisible by the number of treatment arms.
K	number of treatment groups (e.g. K=2 if we compare one experimental against one control treatment).
lb	lower bound for the starting value of the poisson distribution.
lambda	(second) noncentrality parameter of the doubly noncentral t-distribution.
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.
mti	maximum tolerated imbalance in patient numbers during the trial.
N	integer for the total sample size of the trial.
name	name of a variable.
mu	vector of the expected responses of the treatment groups, should have length K (i.e. one entry for each treatment group).
obj	object specifying the randomization procedure, i.e. an object of a class.
object	any R object.
oject	any R object. Inheriting from randPar. See also createParam.
p	success probability of the biased coin (e.g. in Efron's Biased Coin Design).
pr	vector with patient responses, i.e. each patients resulting value after the treatment.
q	"cut-off" value in [0.5,1]. This is the ratio of patients up from which the experimenter imposes selection bias on the data.
r	numeric indicating the number of random sequences to be generated at random or missing.
ratio	vector of length K. The total sample number N and all used block lengths (bc) have to be divisible by sum(ratio).
rb	block lengths of the blocks that can be selected equiprobable at random.
rsob	randomization sequence (of one block).
rs	randomization sequence (of all blocks).
S	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	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 physican.
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 my 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 my 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,...,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 sum(ratio).
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 *Generic plotting of comparison objects*

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)
```

`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.

`randBlocks` *Function returning the block slot of an S4 object*

Description

Function returning the block slot of an S4 object

Usage

```
randBlocks(obj)
```

Arguments

`obj`	object of class pbrPAr

randPar *Settings for randomization procedures*

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)
- Efron's Biased Coin Design (ebcPar)
- Generalized Biased Coin Design (gbcdPar)
- Adjustable Biased Coin Design (abcdPar)
- Bayesian Biased Coin Design (bbcdPar)
- Hadamard Randomization (hadaPar)
- Maximal Procedure (mpPar)
- Permuted Block Randomization (pbrPar)
- Random Allocation Rule (rarPar)
- Permuted Block Randomization with random block length (rpbrPar)
- Truncated Binomial Design with random block length (rtbdPar)
- Truncated Binomial Design (tbdPar)
- Wei's Urn Design (udPar)
- Chen's Design (chenPar)

See Also

Generate randomization sequences genSeq. Calculate the the complete set of randomization sequences of a randomization procedure. getAllSeq.

randPar-class *Randomization paramters generic*

Description

Randomization paramters generic

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 `sum(ratio)`.
`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 — *Function returning the allocation ratio slot of an S4 object*

Description

Function returning the allocation ratio slot of an S4 object

Usage

```
ratio(obj)
```

Arguments

obj	object of class randPar

---

rho — *Function returning the adjusting parameter rho slot of an S4 object*

Description

Function returning the adjusting parameter rho slot of an S4 object

Usage

```
rho(obj)
```

Arguments

obj	object of class randPar

rpbrPar *Representing Randomized Permuted Block Randomization*

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 sum(ratio).
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 possbible 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, gbcdPar, 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 assigend 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

`obj`	object of class assessmentment, e.g. the output of the `assess` function.
`file`	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 — *Saving a randomization lists*

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` *Function returning the allocation seed slot of an object*

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, i.e. an object of a class.

`selBias` *Representing selection bias*

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"` 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 proabability 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 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` *Function returning the standard deviation slot of a normEndp S4 object*

Description

Function returning the standard deviation slot of a normEndp S4 object

Usage

```
sigma(obj)
```

Arguments

`obj`	object of class normEndp

`summary` *Summary of assessments of a randomization procedure*

Description

Summary of assessments of a randomization procedure

Usage

```
summary(object, ...)

## S4 method for signature 'assessment'
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.

Value

Data frame with a summary of the assessment object.

Examples

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

---

tbdPar *Representing Truncated Binomal Design*

Description

Represents the Truncated Binomial Design.

Usage

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

Arguments

bc	vector which contains the lengths k_1,...,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 toin 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

type — *Get type of an object*

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 assigend.

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.

`$,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.

`$,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.

`$,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