Package ‘randomizeR’

July 8, 2015

Title Randomization - Assessment and Evaluation of Randomization Procedures

Version 0.1

Description This package helps to find the optimal randomizaton procedure.
Afterwards, a randomization sequence can be generated from this procedure.
Finally, the outcome of the trial can be analyzed by a randomization test.

Depends R (>= 3.1.2),
methods

License GPL (>=2)

LazyData true

Collate 'util.R'
'randSeq.R'
'normEndp.R'
'getStat.R'
'endpoint.R'
'imbalance.R'
'corGuess.R'
'testDec.R'
'getExpectation.R'
'doublyT.R'
'selBias.R'
'chronBias.R'
'bias.R'
'issue.R'
'assess.R'
'randPar.R'
'ebcPar.R'
'bsdPar.R'
'bsdSeq.R'
'chronBiasStepT.R'
'crPar.R'
'crSeq.R'
'createParam.R'
'ebcSeq.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'
'saveRand.R'
'tbdSeq.R'
'udPar.R'
'udSeq.R'

Suggests testthat,
knitr

VignetteBuilder knitr

R topics documented:

`randomizeR-package` *Randomization Procedures - Assessment and Evaluation*

Description

Randomization Procedures - Assessment and Evaluation

Details

Package:	randomizeR
Type:	Package
Version:	1.0
Date:	2014-12-18
License:	GPL (>= 2)
LazyLoad:	yes

This package provides functionality for randomization in clinical trials. Clinical trials often are accrual. Hence, the patients get allocated into the treatment arms at the time they are admitted in the study. This is reflected in the randomization protocol. If the person electing the patients in the study can guess what treatment will be assigned next he might be able to bias the study. Making use of soft inclusion/exclusion criteria, he may decline a "weak" patient when he can guess that his favoured treatment is going to be next in the allocation list, alleging that the weak patient will have a smaller probability of success for the treatment (binary case) or smaller average treatment effect (continuous case). The bias introduced this way is called selection bias. This package contains functions for the computation of randomization sequences with various protocols, for the computation of selection bias and patient response for binary as well as continuous endpoints.

Author(s)

David Schindler <`dschindler@ukaachen.de`>, Diane Uschner <`duschner@ukaachen.de`>

References

Lievens Diss, Ivanova Paper, non-published research by the authors

---

`assess` *Assessment of a randomization sequence*

Description

Assessment of a randomization sequence

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 the class randSeq.
`...`	one/several object(s) of the class issue.
`endp`	object of the class endpoint (optional).

Examples

```
# assess the full set of PBR(2)
seq <- getAllSeq(pbrPar(4))
issue1 <- corGuess("CS")
issue2 <- corGuess("DS")
issue3 <- imbal("absImb")
issue4 <- imbal("imb")
assess(seq, issue1, issue2, issue3, issue4)
# a randomized sequence from the Big Stick Design
seq <- genSeq(bsdPar(10, 2), seed = 1909)
assess(seq, issue1)
# assess an object of class selBias
endp <- normEndp(c(2, 2), c(1, 1))
issue5 <- selBias("CS", 4, "exact")
assess(seq, issue1, issue5, endp = endp)
```

---

assessment-class *Randomization paramters generic*

Description

Randomization paramters generic

---

bias *Bias class*

Description

Bias class

---

blockRand *Permuted block randomization*

Description

Compute a permuted block randomization sequence for a clinical trial with several blocks.

Usage

```
blockRand(bc, K = 2, ratio = rep(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	The ratio of group A to the total sample size: ratio*N=N_A and N-N_A=N_B.

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).

`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

`blockSeq` *Permuted block*

Description

Compute a vector containing the treatment allocations for one permuted block.

Usage

```
blockSeq(k, K = 2, ratio = rep(1, K))
```

Arguments

`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).
`ratio`	The ratio of group A to the total sample size: `ratio*N=N_A` and `N-N_A=N_B`.

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).

bsdPar *Constructor functions for BSD parameters*

Description

Generates an object of the class mpPar

Usage

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

Arguments

`N`	numeric representing 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.

Value

object of type bsdPar with given parameters

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 paramter creators: `crPar`; `createParam`; `ebcPar`; `hadaPar`; `mpPar`; `pbrPar`; `randPar`; `rarPar`; `rpbrPar`; `rtbdPar`; `tbdPar`; `udPar`

bsdRand *Sampling algorigthm for BSD*

Description

Sampling algorigthm for BSD

Usage

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

Arguments

`N`	numeric representing 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.

`chronBias` *Generate a chronBias object*

Description

The function generates an object of the S4 class `chronBias`. The object contains the information of possible 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"`, `Using "linT" the following linear time trend function is used:` $$f(i) = i\theta$$ `Using "logT" the following logarithmic time trend function is used:` $$f(i) = \log(i)\theta$$ `Using "stepT" the following step function is used:` $$f(i) = 1_{i \geq c > 1}\theta$$
`theta`	factor of the time trend for further details see `type`.
`method`	if `exact` the exact p.value of a ranodmization sequence is calculated, otherwise one p.value for every randomization sequence is simulated.
`saltus`	saltus in the trial when a step time trend is present (for other types of time trends `missing`).
`alpha`	if method is `sim` the two-sided level of each each simulated t.test, otherwise the sum of the corresponding quantiles of the doubly-noncentral t-distribution.

Details

The generated object contains full information

- of the time trend function.
- of the strength of the time trend.
- whether one test decision should be simulated or the p.value should be calculated exact.
- of the alpha level of the two-sided test or of the quantiles of the corresponding distribution function used to determine an exact type-I-error probability of a given randomization sequence.

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: `corGuess`; `imbal`; `selBias`

`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

`completeRand` *Complete Randomization*

Description

This function implements a generalized version of Complete Randomization. In the original version Complete Randomization is equivalent to tossing a fair coin for a 1:1 allocation of subjects into two treatment groups. This version extends the original version to support more than two treatment groups and unequal allocation ratios.

Usage

```
completeRand(N, K = 2, ratio = rep(1, K))
```

Arguments

`N`	numeric representing 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`	The ratio of group A to the total sample size: `ratio*N=N_A` and `N-N_A=N_B`.

Value

A vector with the allocation sequence for a clinical trial. It will contain the number j-1 at position `i`, when patient `i` is allocated to treatment `j` (`j=1,...,K`).

`corGuess` *Generate an object of the class corGuess*

Description

The function generates an object of the S4 class `corGuess`. The object contains information of the biasing policy of the investigator.

Usage

```
corGuess(type)
```

Arguments

`type`	character vector indicating which biasing strategy the investigator is using (selection bias). Possible values: convergence strategy (`CS`) or divergence strategy (`DS`).

Details

The generated objects contains one argument whether the investigator is using the divergence or the convergence strategy for guessing future enrolled patients.

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`; `imbal`; `selBias`

`createParam` *Setting randomization parameters*

Description

Generates an object of a class inheriting from randPar for a tow-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`	numeric representing 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 at random.
`p`	success probability of the biased coin (e.g. in Efrons 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` calls a different function of the family randomization.parameter.creators, see also `randPar`.

Value

an object `Param`, which is available

See Also

Other randomization paramter creators: `bsdPar`; `crPar`; `ebcPar`; `hadaPar`; `mpPar`; `pbrPar`; `randPar`; `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 — *Constructor functions for the CR parameters*

Description

Generates an object of the class crPar

Usage

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

Arguments

`N`	numeric representing 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`	The ratio of group A to the total sample size: `ratio*N=N_A` and `N-N_A=N_B`.
`groups`	character vector of labels for the different treatments.

Value

object of type crPar with given parameters

References

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

See Also

Other randomization paramter creators: `bsdPar`; `createParam`; `ebcPar`; `hadaPar`; `mpPar`; `pbrPar`; `randPar`; `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` *Constructor functions for EBC parameters*

Description

Generates an object of the class ebcPar

Usage

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

Arguments

`N`	numeric representing the total sample size of the trial.
`p`	success probability of the biased coin (e.g. in Efrons Biased Coin Design).
`groups`	character vector of labels for the different treatments.

Value

object of type ebcPar with given parameters

See Also

Other randomization paramter creators: `bsdPar`; `crPar`; `createParam`; `hadaPar`; `mpPar`; `pbrPar`; `randPar`; `rarPar`; `rpbrPar`; `rtbdPar`; `tbdPar`; `udPar`

`efronRand` *Efrons Biased Coin and Big Stick Design*

Description

This procedure generalises efrons biased coin design. It permits a maximum tolerated imbalance `MTI` during the trial. In the setting with success probability p = 0.5 of the biased coin it thus yields the Big Stick Design.

Usage

```
efronRand(bc, p, mti, K = 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.
`p`	success probability of the biased coin (e.g. in Efrons Biased Coin Design).
`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

B. Efron (1971) Forcing a sequential experiment to be balanced. Biometrika, 58, 403-17. J. F. Soares and C. F. Jeff Wu (1983) Some Restricted Randomization Rules in Sequential Designs. Comm. in Stat., 12, 2017-34.

---

endpoint — *Common representation of the endpoints.*

Description

Common representation of the endpoints.

---

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)
```

genSeq,bsdPar,numeric,numeric-method

Generate random sequences

Description

Generates a randomization sequences for a given design.

Usage

```
## 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 '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 '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 '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)

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 '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 '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 '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 sequnces generated at random for the specified randomisatiom 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)
```

`getAllSeq,bsdPar-method`
Get the 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

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

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

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

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

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

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

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 randomisation 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)
```

`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)
```

---

getExpectation,randSeq,chronBias,normEndp-method
Get expectations of a randomization list

Description

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

Usage

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

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

getExpectation(randSeq, bias, endp)

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

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

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

Arguments

`randSeq`	object of the class randSeq.
`bias`	object of the class bias (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)
```

`getProb,bsdSeq-method` *Calculate theoretical probability for observed sequences*

Description

Calculate theoretical probability for observed sequences

Usage

```
## S4 method for signature 'bsdSeq'
getProb(obj)

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

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

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

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

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

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` *Constructor functions for the HADA parameters*

Description

Generates an object of the class hadaPar

Usage

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

Arguments

`N`	numeric representing the total sample size of the trial.
`groups`	character vector of labels for the different treatments.

Value

object of type hadaPar with given parameters

References

R.A. Bailey and P.R. Nelson (2003) Hadamard Randomization: A valid Restriction of Random Permuted Blocks. Biometrical Journal, 45, 554-60. 58, 403-17

See Also

Other randomization paramter creators: `bsdPar`; `crPar`; `createParam`; `ebcPar`; `mpPar`; `pbrPar`; `randPar`; `rarPar`; `rpbrPar`; `rtbdPar`; `tbdPar`; `udPar`

`hadaRand` *Hadamard Randomization*

Description

Computes a Hadamard Randomization sequence for a clinical trial with several blocks.

Usage

```
hadaRand(bc)
```

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.

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).

imbal — *Generate object representing the allocation imbalance*

Description

Balance of the treatment assignment of patients can be an issue in the design of a clinical trial. The `imbal` function generates an object that represents this demand.

Usage

```
imbal(type)
```

Arguments

`type`	character string, should be one of `"imb"`, `"absImb"`, `"loss"`, or `"maxImb"`, with `"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

Details

This is a constructor function for an S4 object of class `imbal`.

References

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

See Also

Other issues: `chronBias`; `corGuess`; `selBias`

Examples

```
issue <- imbal("maxImb")
issue
```

issue — *Issue class*

Description

Issue class

`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` *Constructor functions for MP parameters*

Description

Generates an object of the class mpPar

Usage

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

Arguments

`N`	numeric representing the total sample size of the trial.
`mti`	Maximum tolerated imbalance in patient numbers during the trial.
`ratio`	The ratio of group A to the total sample size: `ratio*N=N_A` and `N-N_A=N_B`.
`groups`	character vector of labels for the different treatments.

Value

object of type mpPar with given parameters

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 paramter creators: `bsdPar`; `crPar`; `createParam`; `ebcPar`; `hadaPar`; `pbrPar`; `randPar`; `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` *Generate normEndp object*

Description

Generate normEndp object

Usage

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

Arguments

`mu`	vector of expectations (length of `K`).
`sigma`	vector of standard deviations (length of `K`).

`normEndp-class` *Representation of the normal endpoints*

Description

Representation of the normal endpoints

`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.
`ini`	integer representing the initial urn composition.
`compr`	factor of compression for the sigmoid-time trend.
`delta`	(first) noncentrality parameter of the doubly noncentral t-distribution.
`df`	degrees of freedom (i.a. `N-2`).
`eta`	strength 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.
`gamma`	selection effect (eta divided by sigma).
`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.
`MTI`	The maximum tolerated imbalance during the trial (depricated).
`N`	numeric representing the total sample size of the trial.
`name`	name of a variable.
`mu`	vector of expectations (length of `K`).
`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 Efrons 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`	The ratio of group A to the total sample size: `ratio*N=N_A` and `N-N_A=N_B`.
`rb`	block lengths of the blocks that can be selected 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`	saltus in the trial when a step time trend is present (for other types of time trends `missing`).
`seed`	seed for the random number generation
`sigma`	vector of standard deviations (length of `K`).
`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.
`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`.

`paramErrors` *Function for errors requesting*

Description

This function is a query to make sure that the parameters are all in the right range.

Usage

```
paramErrors(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`	numeric representing 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 at random.
`p`	success probability of the biased coin (e.g. in Efrons 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.

Value

returns a `TRUE` if everything is fine, otherwise a `FALSE`

`pbrPar` *Constructor functions for the PBR parameters*

Description

Generates an object of the class pbrPar

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`	The ratio of group A to the total sample size: `ratio*N=N_A` and `N-N_A=N_B`.
`groups`	character vector of labels for the different treatments.

Value

object of type pbrPar with given parameters

References

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

See Also

Other randomization paramter creators: `bsdPar`; `crPar`; `createParam`; `ebcPar`; `hadaPar`; `mpPar`; `randPar`; `rarPar`; `rpbrPar`; `rtbdPar`; `tbdPar`; `udPar`

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 — *Setting randomization parameters*

Description

This is a set of classes and generics that provide functionality for generating randomization parameter lists and randomization sequences

Usage

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

Arguments

N	numeric representing 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	The ratio of group A to the total sample size: `ratio*N=N_A` and `N-N_A=N_B`.
groups	character vector of labels for the different treatments.

See Also

Other randomization paramter creators: `bsdPar`; `crPar`; `createParam`; `ebcPar`; `hadaPar`; `mpPar`; `pbrPar`; `rarPar`; `rpbrPar`; `rtbdPar`; `tbdPar`; `udPar`

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 *Constructor functions for the RAR parameters*

Description

Generates an object of the class rarPar

Usage

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

Arguments

`N`	numeric representing 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`	The ratio of group A to the total sample size: `ratio*N=N_A` and `N-N_A=N_B`.
`groups`	character vector of labels for the different treatments.

Value

object of type rarPar with given parameters

References

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

See Also

Other randomization paramter creators: `bsdPar`; `crPar`; `createParam`; `ebcPar`; `hadaPar`; `mpPar`; `pbrPar`; `randPar`; `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

`rpbrPar` *Constructor functions for the PBR parameters*

Description

Generates an object of the class pbrPar

Usage

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

Arguments

`rb`	block lengths of the blocks that can be selected at random.
`N`	numeric representing 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`	The ratio of group A to the total sample size: `ratio*N=N_A` and `N-N_A=N_B`.
`groups`	character vector of labels for the different treatments.
`filledBlock`	`logical` whether the last block should be filled or not.

Value

object of type pbrPar with given parameters

References

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

See Also

Other randomization paramter creators: `bsdPar`; `crPar`; `createParam`; `ebcPar`; `hadaPar`; `mpPar`; `pbrPar`; `randPar`; `rarPar`; `rtbdPar`; `tbdPar`; `udPar`

rtbdPar *Constructor functions for TBD parameters*

Description

Generates an object of the class tbdPar

Usage

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

Arguments

`N`	numeric representing the total sample size of the trial.
`rb`	block lengths of the blocks that can be selected at random.
`K`	number of treatment groups (e.g. K=2 if we compare one experimental against one control treatment).
`ratio`	The ratio of group A to the total sample size: `ratio*N=N_A` and `N-N_A=N_B`.
`groups`	character vector of labels for the different treatments.
`filledBlock`	`logical` whether the last block should be filled or not.

Value

object of type tbdPar with given parameters

References

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

See Also

Other randomization paramter creators: `bsdPar`; `crPar`; `createParam`; `ebcPar`; `hadaPar`; `mpPar`; `pbrPar`; `randPar`; `rarPar`; `rpbrPar`; `tbdPar`; `udPar`

saveRand *Function for saving the randomization list*

Description

Function for saving the parameters of an randSeq object

Usage

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

Arguments

`obj`	object specifying the randomization procedure, i.e. an object of a class.
`file`	A connection, or a character string naming the file to write to.

Value

An object in the home folder.

`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` *Generate a selBias object*

Description

The function generates an object of the S4 class `selBias`. The object contains the information of possible selection bias in a clinical trial.

Usage

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

Arguments

`type`	character string indicating which biasing strategy the investigator is using (selection bias). Possible values: convergence strategy (`CS`) or divergence strategy (`DS`).
`eta`	strength of selection bias.
`method`	if `sim` one test decision for every randomization sequence is calculated, otherewise the exact p.value of a randomization sequence is computed.
`alpha`	if method is `sim` the two-sided level of each each simulated t.test, otherwise the sum of the corresponding quantiles of the doubly-noncentral t-distribution.

Details

The generated object contains full information

- of the used biasing strategy of the investigator.
- of the strength of selection bias.
- whether one test decision should be simulated or the p.value should be calculated exact.
- of the alpha level of the two-sided test or of the quantiles of the corresponding distribution function used to determine an exact type-I-error probability of a given randomization sequence.

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

Other issues: `chronBias`; `corGuess`; `imbal`

`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 a randomization procedure*

Description

Summary of a randomization procedure

Usage

```
summary(object, ...)

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

Arguments

`object`	object of the class assessment.
`...`	additional arguments affecting the summary produced.

Examples

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

tbdPar — *Constructor functions for TBD parameters*

Description

Generates an object of the class tbdPar

Usage

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

Arguments

`bc`	block constellation used in the trial
`K`	number of treatment groups (e.g. K=2 if we compare one experimental against one control treatment).
`ratio`	The ratio of group A to the total sample size: `ratio*N=N_A` and `N-N_A=N_B`.
`groups`	character vector of labels for the different treatments.

Value

object of type tbdPar with given parameters

References

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

See Also

Other randomization paramter creators: `bsdPar`; `crPar`; `createParam`; `ebcPar`; `hadaPar`; `mpPar`; `pbrPar`; `randPar`; `rarPar`; `rpbrPar`; `rtbdPar`; `udPar`

`tbdRand` *Truncated Binomial Design*

Description

This procedure generalises the Truncated Binomial Design

Usage

```
tbdRand(N, bc = N, K = 2, ratio = rep(1, K))
```

Arguments

`N`	numeric representing the total sample size of 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.
`K`	number of treatment groups (e.g. K=2 if we compare one experimental against one control treatment).
`ratio`	The ratio of group A to the total sample size: `ratio*N=N_A` and `N-N_A=N_B`.

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

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

`type` *Function returning the type of a slot of a S4 object*

Description

Function returning the type of a slot of a S4 object

Usage

```
type(obj)
```

Arguments

`obj`	object specifying the randomization procedure, i.e. an object of a class.

udPar — *Constructor functions for UD parameters*

Description

Generates an object of the class udPar

Usage

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

Arguments

`N`	numeric representing 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.

Value

object of type udPar with given parameters.

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 paramter creators: `bsdPar`; `crPar`; `createParam`; `ebcPar`; `hadaPar`; `mpPar`; `pbrPar`; `randPar`; `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.

`$,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