Package 'crmPack'

December 30, 2014

Maintainer Daniel Sabanes Bove <sabanesd@roche.com>

License GPL (>= 2)

Title Object-oriented implementation of CRM designs

LinkingTo Rcpp, RcppArmadillo

LazyLoad yes

Author Daniel Sabanes Bove <sabanesd@roche.com>

Description Object-oriented implementation of CRM designs

Version 0.0-21

Date 2014-12-30

Depends R (>= 3.0.0),
    Rcpp (>= 0.10.6),
    RcppArmadillo (>= 0.3.920),
    ggplot2,
    graphics

Imports methods,
    grid,
    gridExtra,
    GenSA,
    mvtnorm,
    parallel,
    BayesLogit,
    rjags,
    utils

Suggests ggmcmc,
    R2WinBUGS

Collate 'Data-class.R'
    'Data-methods.R'
    'helpers.R'
    'Rules-class.R'
    'Model-class.R'
    'Design-class.R'
    'McmcOptions-class.R'
    'McmcOptions-methods.R'
    'Samples-class.R'
    'Model-methods.R'
    'Rules-methods.R'

'fromQuantiles.R'
'Samples-methods.R'
'Simulations-class.R'
'Simulations-methods.R'
'crmPack-package.R'
'writeModel.R'
'mcmc.R'
'simulate.R'

R topics documented:

`crmPack-package` *Object-oriented implementation of CRM designs*

Description

Object-oriented implementation of CRM designs

Author(s)

Daniel Sabanes Bove <sabanesd@roche.com>

---

approximate *Approximate posterior with (log) normal distribution*

Description

Approximate posterior with (log) normal distribution

Usage

```
approximate(object, ...)
```

Arguments

object	the object
...	unused

Value

the approximation model

---

approximate,Samples-method *Approximate posterior with (log) normal distribution*

Description

It is recommended to use set.seed before, in order to be able to reproduce the resulting approximating model exactly.

Usage

```
## S4 method for signature 'Samples'
approximate(object, model, data, points = seq(from =
  min(data@doseGrid), to = max(data@doseGrid), length = 5L),
  refDose = median(points), logNormal = FALSE, verbose = TRUE, ...)
```

Arguments

object	the Samples object
model	the Model object
data	the Data object
points	optional parameter, which gives the dose values at which the approximation should rely on (default: 5 values equally spaced from minimum to maximum of the dose grid)
refDose	the reference dose to be used (default: median of points)

`logNormal`	use the log-normal prior? (not default) otherwise, the normal prior for the logistic regression coefficients is used
`verbose`	be verbose (progress statements and plot)? (default)
`...`	additional arguments for `Quantiles2LogisticNormal`, e.g. in order to control the approximation quality, etc.
`...`	unused

Value

the approximation `Model`

`as.list,Data-method` *as.list method for the "Data" class*

Description

as.list method for the "Data" class

Usage

```
## S4 method for signature 'Data'
as.list(x, ...)
```

Arguments

`x`	the `Data` object we want to convert
`...`	objects, possibly named.

Value

a list of all slots in `x`

`CohortSize-class` *The virtual class for cohort sizes*

Description

The virtual class for cohort sizes

See Also

`CohortSizeMax`, `CohortSizeMin`, `CohortSizeRange`, `CohortSizeDLT`, `CohortSizeConst`, `CohortSizeParts`

`CohortSizeConst-class` *Constant cohort size*

Description

This class is used when the cohort size should be kept constant.

Slots

`size` the constant integer size

`CohortSizeDLT-class` *Cohort size based on number of DLTs*

Description

Cohort size based on number of DLTs

Slots

`DLTintervals` a vector with the bounds of the relevant DLT intervals of length `n`

`cohortSize` an integer vector of length `n-1` with the cohort sizes in the `DLTintervals`

`CohortSizeMax-class` *Size based on maximum of multiple cohort size rules*

Description

This class can be used to combine multiple cohort size rules with the MAX operation.

Details

`cohortSizeList` contains all cohort size rules, which are again objects of class `CohortSize`. The maximum of these individual cohort sizes is taken to give the final cohort size.

Slots

`cohortSizeList` list of cohort size rules

`CohortSizeMin-class` *Size based on minimum of multiple cohort size rules*

Description

This class can be used to combine multiple cohort size rules with the MIN operation.

Details

`cohortSizeList` contains all cohort size rules, which are again objects of class `CohortSize`. The minimum of these individual cohort sizes is taken to give the final cohort size.

Slots

`cohortSizeList` list of cohort size rules

`CohortSizeParts-class` *Cohort size based on the parts*

Description

This class is used when the cohort size should change for the second part of the dose escalation. Only works in conjunction with `DataParts` objects.

Slots

`sizes` the two sizes for part 1 and part 2

`CohortSizeRange-class` *Cohort size based on dose range*

Description

Cohort size based on dose range

Slots

`intervals` a vector with the bounds of the relevant dose intervals of length `n`

`cohortSize` an integer vector of length `n-1` with the cohort sizes in the `intervals`

`crmPackExample` *Open the example pdf for crmPack*

Description

Calling this helper function should open the example.pdf document, residing in the doc subfolder of the package installation directory.

Usage

```
crmPackExample()
```

Value

nothing

Author(s)

Daniel Sabanes Bove <sabanesd@roche.com>

`crmPackHelp` *Open the browser with help pages for crmPack*

Description

This convenience function opens your browser with the help pages for crmPack.

Usage

```
crmPackHelp()
```

Value

nothing

Author(s)

Daniel Sabanes Bove <sabanesd@roche.com>

Data-class *Class for the data input*

Description

Class for the data input

Slots

- `x` the doses for the patients
- `y` the vector of toxicity events (0 or 1 integers)
- `ID` unique patient IDs (integer vector)
- `cohort` the cohort indices (sorted values from 0, 1, 2, ...)
- `doseGrid` the vector of all possible doses (sorted), i.e. the dose grid
- `nObs` number of observations
- `nGrid` number of gridpoints
- `xLevel` the levels for the doses the patients have been given

DataDual-class *Class for the dual endpoint data input*

Description

This is a subclass of `Data`, so contains all slots from `Data`, and in addition biomarker values.

Slots

- `w` the continuous vector of biomarker values

DataParts-class *Class for the data with two study parts*

Description

This is a subclass of `Data`, so contains all slots from `Data`, and in addition information on the two study parts.

Slots

- `part` integer vector; which part does each of the patients belong to?
- `nextPart` integer; what is the part for the next cohort?
- `part1Ladder` sorted numeric vector; what is the escalation ladder for part 1? This shall be a subset of the `doseGrid`.

Design-class *Class for the CRM design*

Description

Class for the CRM design

Slots

- `model` the model to be used, an object of class `Model`
- `nextBest` how to find the next best dose, an object of class `NextBest`
- `stopping` stopping rule(s) for the trial, an object of class `Stopping`
- `increments` how to control increments between dose levels, an object of class `Increments`
- `cohortSize` rules for the cohort sizes, an object of class `CohortSize`
- `data` what is the dose grid, any previous data, etc., contained in an object of class `Data`
- `startingDose` what is the starting dose? Must lie on the grid in `data`

dose *Compute the doses for a given probability, given model and samples*

Description

Compute the doses for a given probability, given model and samples

Usage

```
dose(prob, model, samples, ...)
```

Arguments

`prob`	the probability
`model`	the `Model`
`samples`	the `Samples`
`...`	unused

dose,numeric,Model,Samples-method

Compute the doses for a given probability, given model and samples

Description

Compute the doses for a given probability, given model and samples

Usage

```
## S4 method for signature 'numeric,Model,Samples'
dose(prob, model, samples, ...)
```

Arguments

prob	the probability
model	the Model
samples	the Samples
...	unused

DualEndpoint-class

Dual endpoint model

Description

todo: describe the model

Slots

mu For the probit toxicity model, mu contains the prior mean vector

Sigma For the probit toxicity model, contains the prior covariance matrix

sigma2betaW For the biomarker model, contains the prior variance factor of the random walk prior. If it is not a single number, it can also contain a vector with elements a and b for the inverse-gamma prior on sigma2betaW.

sigma2W Either a fixed value for the biomarker variance, or a vector with elements a and b for the inverse-gamma prior parameters.

rho Either a fixed value for the correlation (between -1 and 1), or a vector with elements a and b for the Beta prior on the transformation kappa = (rho + 1) / 2, which is in (0, 1). For example, a=1,b=1 leads to a uniform prior on rho.

useRW1 for specifying the random walk prior on the biomarker level: if TRUE, RW1 is used, otherwise RW2.

useFixed a list with logical value for each of the three parameters sigma2betaW, sigma2W and rho indicating whether a fixed value is used or not.

`DualEndpoint2-class` *Dual endpoint model version 2 which shall work with JAGS*

Description

Model class should be the same. Difference is in the initialize method. todo: describe the model

Slots

`mu` For the probit toxicity model, `mu` contains the prior mean vector

`Sigma` For the probit toxicity model, contains the prior covariance matrix

`sigma2betaW` For the biomarker model, contains the prior variance factor of the random walk prior. If it is not a single number, it can also contain a vector with elements `a` and `b` for the inverse-gamma prior on `sigma2betaW`.

`sigma2W` Either a fixed value for the biomarker variance, or a vector with elements `a` and `b` for the inverse-gamma prior parameters.

`rho` Either a fixed value for the correlation (between -1 and 1), or a vector with elements `a` and `b` for the Beta prior on the transformation kappa = (rho + 1) / 2, which is in (0, 1). For example, `a=1,b=1` leads to a uniform prior on rho.

`useRW1` for specifying the random walk prior on the biomarker level: if `TRUE`, RW1 is used, otherwise RW2.

`useFixed` a list with logical value for each of the three parameters `sigma2betaW`, `sigma2W` and `rho` indicating whether a fixed value is used or not.

`extract` *Extract something from an object and produce a data.frame*

Description

Extract something from an object and produce a data.frame

Usage

```
extract(object, ...)
```

Arguments

`object`	the object
`...`	unused

Value

the data frame

extract,Samples-method *Extract certain parameter from Samples object*

Description

Extract certain parameter from Samples object

Usage

```
## S4 method for signature 'Samples'
extract(object, parameter, ...)
```

Arguments

`object`	the `Samples` object
`parameter`	the name of the parameter
`...`	unused

Value

the data frame suitable for use with `ggmcmc`

fitted,Samples-method *Fit method for the Samples class*

Description

Fit method for the Samples class

Usage

```
## S4 method for signature 'Samples'
fitted(object, model, data, points = data@doseGrid,
  quantiles = c(0.025, 0.975), middle = mean, ...)
```

Arguments

`object`	the `Samples` object
`model`	the `Model` object
`data`	the `Data` object
`points`	at which dose levels is the fit requested? default is the dose grid
`quantiles`	the quantiles to be calculated (default: 0.025 and 0.975)
`middle`	the function for computing the middle point. Default: `mean`
`...`	other arguments.

Value

data frame with dose, middle, lower and upper quantiles

GeneralSimulations-class

General class for the simulations output

Description

This class captures trial simulations.

Details

Here also the random generator state before starting the simulation is saved, in order to be able to reproduce the outcome. For this just use `set.seed` with the seed as argument before running `simulate,Design-method`.

Slots

- `data` list of produced `Data` objects
- `doses` the vector of final dose recommendations
- `seed` random generator state before starting the simulation

GeneralSimulations-summary-class

Class for the summary of general simulations output

Description

Class for the summary of general simulations output

Slots

- `target` target toxicity interval
- `targetDoseInterval` corresponding target dose interval
- `nsim` number of simulations
- `propDLTs` proportions of DLTs in the trials
- `meanToxRisk` mean toxicity risks for the patients
- `doseSelected` doses selected as MTD
- `toxAtDosesSelected` true toxicity at doses selected
- `propAtTarget` Proportion of trials selecting target MTD
- `doseMostSelected` dose most often selected as MTD
- `obsToxRateAtDoseMostSelected` observed toxicity rate at dose most often selected
- `nObs` number of patients overall
- `nAboveTarget` number of patients treated above target tox interval
- `doseGrid` the dose grid that has been used

getThreePlusThreeDesign *Creates a new 3+3 design object from a dose grid*

Description

Creates a new 3+3 design object from a dose grid

Usage

```
getThreePlusThreeDesign(doseGrid)
```

Arguments

`doseGrid`	the dose grid to be used

Value

the object of class `RuleDesign` with the 3+3 design

Author(s)

Daniel Sabanes Bove `<sabanesd@roche.com>`

Increments-class *The virtual class for controlling increments*

Description

The virtual class for controlling increments

See Also

`IncrementsRelative`, `IncrementsRelativeDLT`, `IncrementsRelativeParts`

IncrementsRelative-class *Increments control based on relative differences in intervals*

Description

Note that `intervals` is to be read as follows. If for example, we want to specify three intervals: First 0 to less than 50, second at least 50 up to less than 100 mg, and third at least 100 mg, then we specify `intervals` to be `c(0, 50, 100, Inf)`. That means, the right bound of the intervals are exclusive to the interval.

Slots

`intervals` a vector with the bounds of the relevant intervals of length `n`.

`increments` a vector of length `n-1` with the maximum allowable relative increments in the `intervals`

IncrementsRelativeDLT-class

Increments control based on relative differences in terms of DLTs

Description

Note that `DLTintervals` is to be read as follows. If for example, we want to specify three intervals: First 0 DLTs, second 1 or 2 DLTs, and third at least 3 DLTs, then we specify `DLTintervals` to be `c(0, 1, 3,Inf)`. That means, the right bound of the intervals are exclusive to the interval.

Slots

`DLTintervals` a vector with the bounds of the relevant DLT intervals of length `n`

`increments` a vector of length `n-1` with the maximum allowable relative increments in the `DLTintervals`

IncrementsRelativeParts-class

Increments control based on relative differences in intervals, with special rules for part 1 and beginning of part 2

Description

Note that this only works in conjunction with `DataParts` objects. If the part 2 will just be started in the next cohort, then the next maximum dose will be either `dltStart` (e.g. -1) shift of the last part 1 dose in case of a DLT in part 1, or `cleanStart` shift (e.g. 0) in case of no DLTs in part 1. If part 1 will still be on in the next cohort, then the next dose level will be the next higher dose level in the `part1Ladder` of the data object. If part 2 has been started before, the usual relative increment rules apply, see `IncrementsRelative`.

Slots

`dltStart` integer giving the dose level increment for starting part 2 in case of a DLT in part 1

`cleanStart` integer giving the dose level increment for starting part 2 in case of a DLT in part 1. If this is less or equal to 0, then the part 1 ladder will be used to find the maximum next dose. If this is larger than 0, then the relative increment rules will be applied to find the next maximum dose level.

initialize,Data-method

Initialization method for the "Data" class

Description

This is the method for initializing a "Data" class object.

Usage

```
## S4 method for signature 'Data'
initialize(.Object, x = numeric(), y = integer(),
  ID = integer(), cohort = integer(), doseGrid = numeric(), ...)
```

Arguments

`.Object`	the `Data` we want to initialize
`x`	the doses for the patients
`y`	the vector of toxicity events (0 or 1 integers)
`ID`	unique patient IDs (integer vector)
`cohort`	the cohort indices (sorted values from 0, 1, 2, ...)
`doseGrid`	the vector of all possible doses
`...`	data to include in the new object. Named arguments correspond to slots in the class definition. Unnamed arguments must be objects from classes that this class extends.

Details

Note that `ID` and `cohort` can be missing, then a warning will be issued and the variables will be filled with default IDs and best guesses, respectively.

initialize,DualEndpoint-method

Initialization method for the "DualEndpoint" class

Description

Initialization method for the "DualEndpoint" class

Usage

```
## S4 method for signature 'DualEndpoint'
initialize(.Object, mu, Sigma, sigma2betaW, sigma2W,
  rho, smooth = c("RW1", "RW2"), ...)
```

Arguments

`.Object`	the `DualEndpoint` we want to initialize
`mu`	see `DualEndpoint`
`Sigma`	see `DualEndpoint`
`sigma2betaW`	see `DualEndpoint`
`sigma2W`	see `DualEndpoint`
`rho`	see `DualEndpoint`
`smooth`	either “RW1” (default) or “RW2”, for specifying the random walk prior on the biomarker level.
`...`	data to include in the new object. Named arguments correspond to slots in the class definition. Unnamed arguments must be objects from classes that this class extends.

`initialize,DualEndpoint2-method`

Initialization method for the "DualEndpoint2" class (JAGS version)

Description

Initialization method for the "DualEndpoint2" class (JAGS version)

Usage

```
## S4 method for signature 'DualEndpoint2'
initialize(.Object, mu, Sigma, sigma2betaW, sigma2W,
  rho, smooth = c("RW1", "RW2"), ...)
```

Arguments

`.Object`	the `DualEndpoint2` we want to initialize
`mu`	see `DualEndpoint2`
`Sigma`	see `DualEndpoint2`
`sigma2betaW`	see `DualEndpoint2`
`sigma2W`	see `DualEndpoint2`
`rho`	see `DualEndpoint2`
`smooth`	either “RW1” (default) or “RW2”, for specifying the random walk prior on the biomarker level.
`...`	data to include in the new object. Named arguments correspond to slots in the class definition. Unnamed arguments must be objects from classes that this class extends.

`initialize,LogisticKadane-method`

Initialization method for the "LogisticKadane" class

Description

Initialization method for the "LogisticKadane" class

Usage

```
## S4 method for signature 'LogisticKadane'
initialize(.Object, theta, xmin, xmax, ...)
```

Arguments

`.Object`	the `LogisticKadane` we want to initialize
`theta`	the target toxicity probability
`xmin`	the minimum of the dose range
`xmax`	the maximum of the dose range
`...`	data to include in the new object. Named arguments correspond to slots in the class definition. Unnamed arguments must be objects from classes that this class extends.

`initialize,LogisticLogNormal-method`

Initialization method for the "LogisticLogNormal" class

Description

Initialization method for the "LogisticLogNormal" class

Usage

```
## S4 method for signature 'LogisticLogNormal'
initialize(.Object, mean, cov, refDose, ...)
```

Arguments

`.Object`	the `LogisticLogNormal` we want to initialize
`mean`	the prior mean vector
`cov`	the prior covariance matrix
`refDose`	the reference dose
`...`	data to include in the new object. Named arguments correspond to slots in the class definition. Unnamed arguments must be objects from classes that this class extends.

`initialize,LogisticLogNormalSub-method`
Initialization method for the "LogisticLogNormalSub" class

Description

Initialization method for the "LogisticLogNormalSub" class

Usage

```
## S4 method for signature 'LogisticLogNormalSub'
initialize(.Object, mean, cov, refDose, ...)
```

Arguments

`.Object`	the `LogisticLogNormalSub` we want to initialize
`mean`	the prior mean vector
`cov`	the prior covariance matrix
`refDose`	the reference dose
`...`	data to include in the new object. Named arguments correspond to slots in the class definition. Unnamed arguments must be objects from classes that this class extends.

`initialize,LogisticNormal-method`
Initialization method for the "LogisticNormal" class

Description

Initialization method for the "LogisticNormal" class

Usage

```
## S4 method for signature 'LogisticNormal'
initialize(.Object, mean, cov, refDose, ...)
```

Arguments

`.Object`	the `LogisticNormal` we want to initialize
`mean`	the prior mean vector
`cov`	the prior covariance matrix
`refDose`	the reference dose
`...`	data to include in the new object. Named arguments correspond to slots in the class definition. Unnamed arguments must be objects from classes that this class extends.

`initialize,LogisticNormalFixedMixture-method` *Initialization method for the "LogisticNormalFixedMixture" class*

Description

Initialization method for the "LogisticNormalFixedMixture" class

Usage

```
## S4 method for signature 'LogisticNormalFixedMixture'
initialize(.Object, components, weights,
  refDose, logNormal = FALSE, ...)
```

Arguments

`.Object`	the `LogisticNormalFixedMixture` we want to initialize
`components`	the specifications of the mixture components: a list with one list of `mean` and `cov` for each bivariate (log) normal prior
`weights`	the weights of the components, these must be positive and will be normalized to sum to 1
`refDose`	the reference dose
`logNormal`	should a log normal prior be specified, such that the mean vectors and covariance matrices are valid for the intercept and log slope? (not default)
`...`	data to include in the new object. Named arguments correspond to slots in the class definition. Unnamed arguments must be objects from classes that this class extends.

`initialize,LogisticNormalMixture-method` *Initialization method for the "LogisticNormalMixture" class*

Description

Initialization method for the "LogisticNormalMixture" class

Usage

```
## S4 method for signature 'LogisticNormalMixture'
initialize(.Object, comp1, comp2, weightpar,
  refDose, ...)
```

Arguments

`.Object`	the `LogisticNormalMixture` we want to initialize
`comp1`	the specifications of the first component: a list with `mean` and `cov` for the first bivariate normal prior
`comp2`	the specifications of the second component
`weightpar`	the beta parameters for the weight of the first component
`refDose`	the reference dose
`...`	data to include in the new object. Named arguments correspond to slots in the class definition. Unnamed arguments must be objects from classes that this class extends.

`initialize,McmcOptions-method` *Initialization method for the "McmcOptions" class*

Description

Initialization method for the "McmcOptions" class

Usage

```
## S4 method for signature 'McmcOptions'
initialize(.Object, burnin = 10000L, step = 2L,
  samples = 10000L, ...)
```

Arguments

`.Object`	the `McmcOptions` we want to initialize
`burnin`	number of burn-in iterations which are not saved (default: `10,000`)
`step`	only every step-th iteration is saved after the burn-in (default: `2`)
`samples`	number of resulting samples (by default `10,000` will result)
`...`	data to include in the new object. Named arguments correspond to slots in the class definition. Unnamed arguments must be objects from classes that this class extends.

`LogisticKadane-class` *Reparametrized logistic model*

Description

This is the logistic model in the parametrization of Kadane et al. (1980).

Details

Let $\rho_0 = p(x_{min})$ be the probability of a DLT and the minimum dose x_{min}, and let γ be the dose with target toxicity probability θ, i.e. $p(\gamma) = \theta$. Then it can easily be shown that the logistic regression model has intercept

$$\frac{\gamma logit(\rho_0) - x_{min} logit(\theta)}{\gamma - x_{min}}$$

and slope

$$\frac{logit(theta) - logit(\rho_0)}{\gamma - x_{min}}$$

The prior is a uniform distribution for γ between x_{min} and x_{max}, and for ρ_0 as well a uniform distribution between 0 and θ.

The slots of this class, required for creating the model, are the target toxicity, as well as the minimum and maximum of the dose range. Note that these can be different from the minimum and maximum of the dose grid in the data later on.

Slots

`theta` the target toxicity probability θ

`xmin` the minimum of the dose range x_{min}

`xmax` the maximum of the dose range x_{max}

LogisticLogNormal-class

Standard logistic model with bivariate (log) normal prior

Description

This is the usual logistic regression model with a bivariate normal prior on the intercept and log slope.

Details

The covariate is the natural logarithm of the dose x divided by the reference dose x^*:

$$logit[p(x)] = \alpha + \beta \cdot \log(x/x^*)$$

where $p(x)$ is the probability of observing a DLT for a given dose x.

The prior is

$$(\alpha, \log(\beta)) \sim Normal(\mu, \Sigma)$$

The slots of this class contain the mean vector and the covariance matrix of the bivariate normal distribution, as well as the reference dose.

Slots

`mean` the prior mean vector μ

`cov` the prior covariance matrix Σ

`refDose` the reference dose x^*

LogisticLogNormalSub-class *Standard logistic model with bivariate (log) normal prior with substractive dose standardization*

Description

This is the usual logistic regression model with a bivariate normal prior on the intercept and log slope.

Details

The covariate is the dose x minus the reference dose x^*:

$$logit[p(x)] = \alpha + \beta \cdot (x - x^*)$$

where $p(x)$ is the probability of observing a DLT for a given dose x.

The prior is

$$(\alpha, \log(\beta)) \sim Normal(\mu, \Sigma)$$

The slots of this class contain the mean vector and the covariance matrix of the bivariate normal distribution, as well as the reference dose.

Slots

`mean` the prior mean vector μ

`cov` the prior covariance matrix Σ

`refDose` the reference dose x^*

LogisticNormal-class *Standard logistic model with bivariate normal prior*

Description

This is the usual logistic regression model with a bivariate normal prior on the intercept and slope.

Details

The covariate is the natural logarithm of the dose x divided by the reference dose x^*:

$$logit[p(x)] = \alpha + \beta \cdot \log(x/x^*)$$

where $p(x)$ is the probability of observing a DLT for a given dose x.

The prior is

$$(\alpha, \beta) \sim Normal(\mu, \Sigma)$$

The slots of this class contain the mean vector, the covariance and precision matrices of the bivariate normal distribution, as well as the reference dose.

Slots

`mean` the prior mean vector μ

`cov` the prior covariance matrix Σ

`prec` the prior precision matrix Σ^{-1}

`refDose` the reference dose x^*

`LogisticNormalFixedMixture-class` *Standard logistic model with fixed mixture of multiple bivariate (log) normal priors*

Description

This is standard logistic regression model with a mixture of multiple bivariate (log) normal priors on the intercept and slope parameters. The weights of the normal priors are fixed, hence no additional model parameters are introduced. This type of prior is often used to better approximate a given posterior distribution, or when the information is given in terms of a mixture.

Details

The covariate is the natural logarithm of the dose x divided by the reference dose x^*:

$$logit[p(x)] = \alpha + \beta \cdot \log(x/x^*)$$

where $p(x)$ is the probability of observing a DLT for a given dose x.

The prior is

$$(\alpha, \beta) \sim \sum_{j=1}^{K} w_j Normal(\mu_j, \Sigma_j)$$

if a normal prior is used and

$$(\alpha, \log(\beta)) \sim \sum_{j=1}^{K} w_j Normal(\mu_j, \Sigma_j)$$

if a log normal prior is used.

The weight w_j of the components are fixed and sum to 1.

The (additional) slots of this class comprise two lists, containing the mean vector, the covariance and precision matrices of the two bivariate normal distributions each, the parameters of the beta prior for the first component weight, as well as the reference dose. Moreover, a slot specifies whether a log normal prior is used.

Slots

`components` a list with one entry per component of the mixture. Each entry is a list with `mean`, `cov` and `prec` for the bivariate normal prior

`weights` the weights of the components, these must be positive and sum to 1

`refDose` the reference dose x^*

`logNormal` is a log normal prior specified for each of the components?

`LogisticNormalMixture-class` *Standard logistic model with flexible mixture of two bivariate normal priors*

Description

This is standard logistic regression model with a mixture of two bivariate normal priors on the intercept and slope parameters. The weight of the two normal priors is a model parameter, hence it is a flexible mixture. This type of prior is often used with a mixture of a minimal informative and an informative component, in order to make the CRM more robust to data deviations from the informative component.

Details

The covariate is the natural logarithm of the dose x divided by the reference dose x^*:

$$logit[p(x)] = \alpha + \beta \cdot \log(x/x^*)$$

where $p(x)$ is the probability of observing a DLT for a given dose x.

The prior is

$$(\alpha, \beta) \sim w * Normal(\mu_1, \Sigma_1) + (1 - w) * Normal(\mu_2, \Sigma_2)$$

The weight w for the first component is assigned a beta prior B(a, b).

The slots of this class comprise two lists, containing the mean vector, the covariance and precision matrices of the two bivariate normal distributions each, the parameters of the beta prior for the first component weight, as well as the reference dose.

Slots

`comp1` the specifications of the first component: a list with `mean`, `cov` and `prec` for the first bivariate normal prior

`comp2` the specifications of the second component

`weightpar` the beta parameters for the weight of the first component

`refDose` the reference dose x^*

`logit` *Shorthand for logit function*

Description

Shorthand for logit function

Usage

```
logit(x)
```

Arguments

`x`	the function argument

Value

the logit(x)

`maxDose` *Determine the maximum possible next dose*

Description

Determine the upper limit of the next dose based on the increments rule.

Usage

```
maxDose(increments, data, ...)
```

Arguments

`increments`	The rule, an object of class `Increments`
`data`	The data input, an object of class `Data`
`...`	further arguments

Details

This function outputs the maximum possible next dose, based on the corresponding rule `increments` and the `data`.

Value

the maximum possible next dose

`maxDose,IncrementsRelative,Data-method` *Determine the maximum possible next dose based on relative increments*

Description

Determine the maximum possible next dose based on relative increments

Usage

```
## S4 method for signature 'IncrementsRelative,Data'
maxDose(increments, data, ...)
```

Arguments

`increments`	The rule, an object of class `Increments`
`data`	The data input, an object of class `Data`
`...`	further arguments

maxDose,IncrementsRelativeDLT,Data-method

Determine the maximum possible next dose based on relative increments determined by DLTs so far

Description

Determine the maximum possible next dose based on relative increments determined by DLTs so far

Usage

```
## S4 method for signature 'IncrementsRelativeDLT,Data'
maxDose(increments, data, ...)
```

Arguments

increments	The rule, an object of class Increments
data	The data input, an object of class Data
...	further arguments

maxDose,IncrementsRelativeParts,DataParts-method

Determine the maximum possible next dose based on relative increments and part 1 and 2

Description

Determine the maximum possible next dose based on relative increments and part 1 and 2

Usage

```
## S4 method for signature 'IncrementsRelativeParts,DataParts'
maxDose(increments, data, ...)
```

Arguments

increments	The rule, an object of class Increments
data	The data input, an object of class Data
...	further arguments

`maxSize` *"MAX" combination of cohort size rules*

Description

This function combines cohort size rules by taking the maximum of all sizes.

Usage

```
maxSize(...)
```

Arguments

`...`	Objects of class `CohortSize`

Value

the combination as an object of class `CohortSizeMax`

See Also

`minSize`

`maxSize,CohortSize-method` *The method combining cohort size rules by taking maximum*

Description

The method combining cohort size rules by taking maximum

Usage

```
## S4 method for signature 'CohortSize'
maxSize(...)
```

Arguments

`...`	Objects of class `CohortSize`

mcmc *Obtain posterior samples for all model parameters*

Description

Obtain posterior samples for all model parameters

Usage

```
mcmc(data, model, options, ...)
```

Arguments

`data`	The data input, an object of class `Data`
`model`	The model input, an object of class `Model`
`options`	MCMC options, an object of class `McmcOptions`
`...`	unused

Details

This is the function to actually run the MCMC machinery to produce posterior samples from all model parameters and required derived values. It is a generic function, so that customized versions may be conveniently defined for specific subclasses of Data, Model, and McmcOptions input.

Value

The posterior samples, an object of class `Samples`.

mcmc,Data,LogisticNormal,McmcOptions-method *The fast method for the LogisticNormal class*

Description

The fast method for the LogisticNormal class

Usage

```
## S4 method for signature 'Data,LogisticNormal,McmcOptions'
mcmc(data, model, options,
  verbose = FALSE, ...)
```

Arguments

`verbose`	shall messages be printed? (not default)
`data`	The data input, an object of class `Data`
`model`	The model input, an object of class `Model`
`options`	MCMC options, an object of class `McmcOptions`
`...`	unused

mcmc,Data,Model,McmcOptions-method *Standard method which uses JAGS/BUGS*

Description

Standard method which uses JAGS/BUGS

Usage

```
## S4 method for signature 'Data,Model,McmcOptions'
mcmc(data, model, options,
  program = c("JAGS", "OpenBUGS", "WinBUGS"), verbose = FALSE, ...)
```

Arguments

`program`	the program which shall be used: either “JAGS” (default), “OpenBUGS” or “WinBUGS”
`verbose`	shall messages be printed? (not default)
`data`	The data input, an object of class `Data`
`model`	The model input, an object of class `Model`
`options`	MCMC options, an object of class `McmcOptions`
`...`	unused

McmcOptions-class *Class for the three canonical MCMC options*

Description

Class for the three canonical MCMC options

Slots

`iterations` number of MCMC iterations

`burnin` number of burn-in iterations which are not saved

`step` only every step-th iteration is saved after the burn-in

MinimalInformative *Construct a minimally informative prior*

Description

This function constructs a minimally informative prior, which is captured in a `LogisticNormal` object.

Usage

```
MinimalInformative(dosegrid, refDose, threshmin = 0.2, threshmax = 0.3, ...)
```

Arguments

`dosegrid`	the dose grid
`refDose`	the reference dose
`threshmin`	Any toxicity probability above this threshold would be very unlikely (5%) at the minimum dose (default: 0.2)
`threshmax`	Any toxicity probability below this threshold would be very unlikely (5%) at the maximum dose (default: 0.3)
`...`	additional arguments for computations, see `Quantiles2LogisticNormal`

Details

Based on the proposal by Neuenschwander et al (2008, Statistics in Medicine), a minimally informative prior distribution is constructed. The required key input is the minimum (d_1 in the notation of the Appendix A.1 of that paper) and the maximum value (d_J) of the dose grid supplied to this function. Then `threshmin` is the probability threshold q_1, such that any probability of DLT larger than q_1 has only 5% probability. Likewise, `threshmax` is the probability threshold q_J, such that any probability of DLT smaller than q_J has only 5% probability. Subsequently, for all doses supplied in the `dosegrid` argument, Beta distributions are set up, and `Quantiles2LogisticNormal` is used to transform the resulting quantiles into an approximating `LogisticNormal` model.

Value

see `Quantiles2LogisticNormal`

minSize *"MIN" combination of cohort size rules*

Description

This function combines cohort size rules by taking the minimum of all sizes.

Usage

```
minSize(...)
```

Arguments

`...` Objects of class `CohortSize`

Value

the combination as an object of class `CohortSizeMin`

See Also

`maxSize`

`minSize,CohortSize-method` *The method combining cohort size rules by taking minimum*

Description

The method combining cohort size rules by taking minimum

Usage

```
## S4 method for signature 'CohortSize'
minSize(...)
```

Arguments

`...` Objects of class `CohortSize`

`Model-class` *Class for the model input*

Description

This is the general model class, from which all other specific models inherit.

Details

The `datamodel` must obey the convention that the data input is called exactly as in the `Data` class. All prior distributions for parameters should be contained in the model function `priormodel`. The background is that this can be used to simulate from the prior distribution, before obtaining any data.

The `dose` function has as first argument `prob`, a scalar toxicity probability which is targeted. Additional arguments are model parameters. Then it computes, using model parameter(s) (samples), the resulting dose. Note that the model parameters are called exactly as in the `model` and must be included in the `sample` vector. The vectors of all samples for these parameters will then be supplied to the function. So your function must be able to process vectors of the model parameters, i.e. it must vectorize over them.

The prob function has as first argument dose, which is a scalar dose. Additional arguments are model parameters. Then it computes, using model parameter(s) (samples), the resulting probability of toxicity at that dose. Again here, the function must vectorize over the model parameters.

If you work with multivariate parameters, then please assume that your the two functions receive either one parameter value as a row vector, or a samples matrix where the rows correspond to the sampling index, i.e. the layout is then nSamples x dimParameter.

Note that dose and prob are the inverse functions of each other.

Slots

datamodel a function representing the BUGS data model specification (see the details above)

priormodel a function representing the BUGS prior specification (see the details above)

datanames The names of all Data slots that are used in the datamodel and/or priormodel definition. Note that you cannot specify more variables than those that are really used in the model!

modelspecs a function computing the list of the data model and prior model specifications that are required for fully specifying them (e.g. prior parameters, reference dose, etc.), based on the Data slots that are then required as arguments of this function. This will then be passed to BUGS for the computations.

dose a function computing the dose reaching a specific target probability, based on the model parameters and additional prior settings (see the details above)

prob a function computing the probability of toxicity for a specific dose, based on the model parameters and additional prior settings (see the details above)

init a function computing the list of starting values for parameters required to be initialized in the MCMC sampler, based on the Data slots that are then required as arguments of this function

sample names of all parameters from which you would like to save the MCMC samples. These must include the ones required by the dose and prob functions.

See Also

LogisticNormal, LogisticLogNormal, LogisticKadane, DualEndpoint

---

nextBest *Find the next best dose*

Description

Compute the recommended next best dose.

Usage

```
nextBest(nextBest, doselimit, samples, model, data, ...)
```

Arguments

nextBest	The rule, an object of class NextBest
doselimit	The maximum allowed next dose
samples	the Samples object
model	The model input, an object of class Model
data	The data input, an object of class Data
...	possible additional arguments without method dispatch

Details

This function outputs the next best dose recommendation based on the corresponding rule `nextBest`, the posterior `samples` from the `model` and the underlying `data`.

Value

a list with the next best dose (element `value`) on the grid defined in `data`, and a plot depicting this recommendation (element `plot`)

---

nextBest,NextBestDualEndpoint,numeric,Samples,DualEndpoint,Data-method

Find the next best dose based on the dual endpoint model

Description

Find the next best dose based on the dual endpoint model

Usage

```
  ## S4 method for signature
## 'NextBestDualEndpoint,numeric,Samples,DualEndpoint,Data'
nextBest(nextBest,
  doselimit, samples, model, data, ...)
```

Arguments

`nextBest`	The rule, an object of class `NextBest`
`doselimit`	The maximum allowed next dose
`samples`	the `Samples` object
`model`	The model input, an object of class `Model`
`data`	The data input, an object of class `Data`
`...`	possible additional arguments without method dispatch

---

nextBest,NextBestMTD,numeric,Samples,Model,Data-method

Find the next best dose based on the MTD rule

Description

Find the next best dose based on the MTD rule

Usage

```
## S4 method for signature 'NextBestMTD,numeric,Samples,Model,Data'
nextBest(nextBest, doselimit,
  samples, model, data, ...)
```

Arguments

`nextBest`	The rule, an object of class `NextBest`
`doselimit`	The maximum allowed next dose
`samples`	the `Samples` object
`model`	The model input, an object of class `Model`
`data`	The data input, an object of class `Data`
`...`	possible additional arguments without method dispatch

`nextBest,NextBestNCRM,numeric,Samples,Model,Data-method`
Find the next best dose based on the NCRM method

Description

Find the next best dose based on the NCRM method

Usage

```
## S4 method for signature 'NextBestNCRM,numeric,Samples,Model,Data'
nextBest(nextBest,
  doselimit, samples, model, data, ...)
```

Arguments

`nextBest`	The rule, an object of class `NextBest`
`doselimit`	The maximum allowed next dose
`samples`	the `Samples` object
`model`	The model input, an object of class `Model`
`data`	The data input, an object of class `Data`
`...`	possible additional arguments without method dispatch

`nextBest,NextBestNCRM,numeric,Samples,Model,DataParts-method`
Find the next best dose based on the NCRM method when two parts trial is used

---

Description

Find the next best dose based on the NCRM method when two parts trial is used

Usage

```
## S4 method for signature 'NextBestNCRM,numeric,Samples,Model,DataParts'
nextBest(nextBest,
  doselimit, samples, model, data, ...)
```

Arguments

nextBest	The rule, an object of class NextBest
doselimit	The maximum allowed next dose
samples	the Samples object
model	The model input, an object of class Model
data	The data input, an object of class Data
...	possible additional arguments without method dispatch

nextBest,NextBestThreePlusThree,missing,missing,missing,Data-method
Find the next best dose based on the 3+3 method

Description

Find the next best dose based on the 3+3 method

Usage

```
  ## S4 method for signature
## 'NextBestThreePlusThree,missing,missing,missing,Data'
nextBest(nextBest,
  doselimit, samples, model, data, ...)
```

Arguments

nextBest	The rule, an object of class NextBest
doselimit	The maximum allowed next dose
samples	the Samples object
model	The model input, an object of class Model
data	The data input, an object of class Data
...	possible additional arguments without method dispatch

NextBest-class *The virtual class for finding next best dose*

Description

The virtual class for finding next best dose

See Also

NextBestMTD, NextBestNCRM, NextBestDualEndpoint, NextBestThreePlusThree

`NextBestDualEndpoint-class` *The class with the input for finding the next dose based on the dual endpoint model*

Description

The class with the input for finding the next dose based on the dual endpoint model

Slots

- `target` the biomarker level, relative to the maximum, that needs to be reached. For example, 0.9 means that a dose with 90 of the maximum biomarker level is considered as having reached sufficient biomarker level.
- `overdose` the overdose toxicity interval
- `maxOverdoseProb` maximum overdose probability that is allowed

`NextBestMTD-class` *The class with the input for finding the next best MTD estimate*

Description

The class with the input for finding the next best MTD estimate

Slots

- `target` the target toxicity probability
- `derive` the function which derives from the input, a vector of posterior MTD samples called `mtdSamples`, the final next best MTD estimate.

`NextBestNCRM-class` *The class with the input for finding the next dose in target interval*

Description

Note that to avoid numerical problems, the dose selection algorithm has been implemented as follows: First admissible doses are found, which are those with probability to fall in `overdose` category being below `maxOverdoseProb`. Next, within the admissible doses, the maximum probability to fall in the `target` category is calculated. If that is above 5% (i.e., it is not just numerical error), then the corresponding dose is the next recommended dose. Otherwise, the highest admissible dose is the next recommended dose.

Slots

- `target` the target toxicity interval
- `overdose` the overdose toxicity interval
- `maxOverdoseProb` maximum overdose probability that is allowed

`NextBestThreePlusThree-class` *The class with the input for finding the next dose in target interval*

Description

Implements the classical 3+3 dose recommendation. No input is required, hence this class has no slots.

`or-Stopping-Stopping` *The method combining two atomic stopping rules*

Description

The method combining two atomic stopping rules

Usage

```
## S4 method for signature 'Stopping,Stopping'
e1 | e2
```

Arguments

`e1`	First `Stopping` object
`e2`	Second `Stopping` object

Value

The `StoppingAny` object

`or-Stopping-StoppingAny` *The method combining a stopping list and an atomic*

Description

The method combining a stopping list and an atomic

Usage

```
## S4 method for signature 'StoppingAny,Stopping'
e1 | e2
```

Arguments

`e1`	`StoppingAny` object
`e2`	`Stopping` object

Value

The modified `StoppingAny` object

or-StoppingAny-Stopping

The method combining an atomic and a stopping list

Description

The method combining an atomic and a stopping list

Usage

```
## S4 method for signature 'Stopping,StoppingAny'
e1 | e2
```

Arguments

`e1`	`Stopping` object
`e2`	`StoppingAny` object

Value

The modified `StoppingAny` object

plot,Data,missing-method

Plot method for the "Data" class

Description

Plot method for the "Data" class

Usage

```
## S4 method for signature 'Data,missing'
plot(x, y, ...)
```

Arguments

`x` the `Data` object we want to plot

`y` the y coordinates of points in the plot, *optional* if `x` is an appropriate structure.

`...` Arguments to be passed to methods, such as graphical parameters (see `par`). Many methods will accept the following arguments:

`type` what type of plot should be drawn. Possible types are

- `"p"` for **p**oints,
- `"l"` for **l**ines,
- `"b"` for **b**oth,
- `"c"` for the lines part alone of `"b"`,
- `"o"` for both '**o**verplotted',

- "h" for '**h**istogram' like (or 'high-density') vertical lines,
- "s" for stair **s**teps,
- "S" for other **s**teps, see 'Details' below,
- "n" for no plotting.

All other types give a warning or an error; using, e.g., type = "punkte" being equivalent to type = "p" for S compatibility. Note that some methods, e.g.plot.factor, do not accept this.

main an overall title for the plot: see title.

sub a sub title for the plot: see title.

xlab a title for the x axis: see title.

ylab a title for the y axis: see title.

asp the y/x aspect ratio, see plot.window.

Value

the ggplot object

---

plot,DataDual,missing-method

Plot method for the "DataDual" class

Description

Plot method for the "DataDual" class

Usage

```
## S4 method for signature 'DataDual,missing'
plot(x, y, ...)
```

Arguments

x the DataDual object we want to plot

y the y coordinates of points in the plot, *optional* if x is an appropriate structure.

... Arguments to be passed to methods, such as graphical parameters (see par). Many methods will accept the following arguments:

type what type of plot should be drawn. Possible types are

- "p" for **p**oints,
- "l" for **l**ines,
- "b" for **b**oth,
- "c" for the lines part alone of "b",
- "o" for both '**o**verplotted',
- "h" for '**h**istogram' like (or 'high-density') vertical lines,
- "s" for stair **s**teps,
- "S" for other **s**teps, see 'Details' below,
- "n" for no plotting.

All other `types` give a warning or an error; using, e.g., `type = "punkte"` being equivalent to `type = "p"` for S compatibility. Note that some methods, e.g.`plot.factor`, do not accept this.

`main` an overall title for the plot: see `title`.

`sub` a sub title for the plot: see `title`.

`xlab` a title for the x axis: see `title`.

`ylab` a title for the y axis: see `title`.

`asp` the y/x aspect ratio, see `plot.window`.

Value

the `ggplot` object

`plot,GeneralSimulations,missing-method` *Plot simulations*

Description

Summarize the simulations with plots

Usage

```
## S4 method for signature 'GeneralSimulations,missing'
plot(x, y, type = c("trajectory",
  "dosesTried"), ...)
```

Arguments

`x`	the `GeneralSimulations` object we want to plot from
`type`	the type of plots you want to obtain.
`y`	the y coordinates of points in the plot, *optional* if `x` is an appropriate structure.
`...`	Arguments to be passed to methods, such as graphical parameters (see `par`). Many methods will accept the following arguments:

`type` what type of plot should be drawn. Possible types are

- `"p"` for **p**oints,
- `"l"` for **l**ines,
- `"b"` for **b**oth,
- `"c"` for the lines part alone of `"b"`,
- `"o"` for both ‘**o**verplotted’,
- `"h"` for ‘**h**istogram’ like (or ‘high-density’) vertical lines,
- `"s"` for stair **s**teps,
- `"S"` for other **s**teps, see ‘Details’ below,
- `"n"` for no plotting.

All other `types` give a warning or an error; using, e.g., `type = "punkte"` being equivalent to `type = "p"` for S compatibility. Note that some methods, e.g.`plot.factor`, do not accept this.

`main` an overall title for the plot: see `title`.

`sub` a sub title for the plot: see `title`.

`xlab` a title for the x axis: see `title`.

`ylab` a title for the y axis: see `title`.

`asp` the y/x aspect ratio, see `plot.window`.

Details

This plot method can be applied to `GeneralSimulations` objects in order to summarize them graphically. Possible `type`s of plots at the moment are:

trajectory Summary of the trajectory of the simulated trials

dosesTried Average proportions of the doses tested in patients

You can specify one or both of these in the `type` argument.

Value

A single `ggplot2` object if a single plot is asked for, otherwise a `gridExtra{gTree}` object.

`plot,GeneralSimulations-summary,missing-method`

Plot summaries of the general simulations

Description

Graphical display of the general simulation summary

Usage

```
## S4 method for signature 'GeneralSimulations-summary,missing'
plot(x, y, type = c("nObs",
  "doseSelected", "propDLTs", "nAboveTarget"), ...)
```

Arguments

`x` the `GeneralSimulations-summary` object we want to plot from

`type` the types of plots you want to obtain.

`y` the y coordinates of points in the plot, *optional* if `x` is an appropriate structure.

`...` Arguments to be passed to methods, such as graphical parameters (see `par`). Many methods will accept the following arguments:

`type` what type of plot should be drawn. Possible types are

- `"p"` for **p**oints,
- `"l"` for **l**ines,
- `"b"` for **b**oth,
- `"c"` for the lines part alone of `"b"`,
- `"o"` for both ‘**o**verplotted’,
- `"h"` for ‘**h**istogram’ like (or ‘high-density’) vertical lines,
- `"s"` for stair **s**teps,
- `"S"` for other **s**teps, see ‘Details’ below,

- "n" for no plotting.

All other types give a warning or an error; using, e.g., type = "punkte" being equivalent to type = "p" for S compatibility. Note that some methods, e.g.plot.factor, do not accept this.

main an overall title for the plot: see title.

sub a sub title for the plot: see title.

xlab a title for the x axis: see title.

ylab a title for the y axis: see title.

asp the y/x aspect ratio, see plot.window.

Details

This plot method can be applied to GeneralSimulations-summary objects in order to summarize them graphically. Possible types of plots at the moment are:

nObs Distribution of the number of patients in the simulated trials

doseSelected Distribution of the final selected doses in the trials. Note that this can include zero entries, meaning that the trial was stopped because all doses in the dose grid appeared too toxic.

propDLTs Distribution of the proportion of patients with DLTs in the trials

nAboveTarget Distribution of the number of patients treated at doses which are above the target toxicity interval (as specified by the truth and target arguments to summary,GeneralSimulations-method)

You can specify any subset of these in the type argument.

Value

A single ggplot2 object if a single plot is asked for, otherwise a gridExtra{gTree} object.

---

plot,Samples,DualEndpoint-method

Plot method for the "Samples" object, when we have the dual endpoint model

---

Description

Plot method for the "Samples" object, when we have the dual endpoint model

Usage

```
## S4 method for signature 'Samples,DualEndpoint'
plot(x, y, data, extrapolate = TRUE, ...)
```

Arguments

`x` the `Samples` object

`y` the `DualEndpoint` object

`data` the `DataDual` object

`extrapolate` should the biomarker fit be extrapolated to the whole dose grid? (default)

`...` Arguments to be passed to methods, such as graphical parameters (see `par`). Many methods will accept the following arguments:

`type` what type of plot should be drawn. Possible types are

- `"p"` for **p**oints,
- `"l"` for **l**ines,
- `"b"` for **b**oth,
- `"c"` for the lines part alone of `"b"`,
- `"o"` for both '**o**verplotted',
- `"h"` for '**h**istogram' like (or 'high-density') vertical lines,
- `"s"` for stair **s**teps,
- `"S"` for other **s**teps, see 'Details' below,
- `"n"` for no plotting.

All other `type`s give a warning or an error; using, e.g., `type = "punkte"` being equivalent to `type = "p"` for S compatibility. Note that some methods, e.g.`plot.factor`, do not accept this.

`main` an overall title for the plot: see `title`.

`sub` a sub title for the plot: see `title`.

`xlab` a title for the x axis: see `title`.

`ylab` a title for the y axis: see `title`.

`asp` the y/x aspect ratio, see `plot.window`.

Value

the `ggplot` object

`plot,Samples,Model-method`

Plot method for the "Samples" and "Model" object

Description

Plot method for the "Samples" and "Model" object

Usage

```
## S4 method for signature 'Samples,Model'
plot(x, y, data, ..., xlab = "Dose level",
  ylab = "Probability of DLT [%]")
```

Arguments

`x` the `Samples` object

`y` the `Model` object

`data` the `Data` object

`xlab` the x axis label

`ylab` the y axis label

`...` Arguments to be passed to methods, such as graphical parameters (see `par`). Many methods will accept the following arguments:

`type` what type of plot should be drawn. Possible types are

- `"p"` for **p**oints,
- `"l"` for **l**ines,
- `"b"` for **b**oth,
- `"c"` for the lines part alone of `"b"`,
- `"o"` for both '**o**verplotted',
- `"h"` for '**h**istogram' like (or 'high-density') vertical lines,
- `"s"` for stair **s**teps,
- `"S"` for other **s**teps, see 'Details' below,
- `"n"` for no plotting.

All other `type`s give a warning or an error; using, e.g., `type = "punkte"` being equivalent to `type = "p"` for S compatibility. Note that some methods, e.g.`plot.factor`, do not accept this.

`main` an overall title for the plot: see `title`.

`sub` a sub title for the plot: see `title`.

`xlab` a title for the x axis: see `title`.

`ylab` a title for the y axis: see `title`.

`asp` the y/x aspect ratio, see `plot.window`.

Value

the `ggplot` object

`plot,Simulations-summary,missing-method`

Plot summaries of the model-based design simulations

Description

Graphical display of the simulation summary

Usage

```
## S4 method for signature 'Simulations-summary,missing'
plot(x, y, type = c("nObs",
  "doseSelected", "propDLTs", "nAboveTarget", "meanFit"), ...)
```

Arguments

`x` the `Simulations-summary` object we want to plot from

`type` the types of plots you want to obtain.

`y` the y coordinates of points in the plot, *optional* if `x` is an appropriate structure.

`...` Arguments to be passed to methods, such as graphical parameters (see `par`). Many methods will accept the following arguments:

`type` what type of plot should be drawn. Possible types are

- `"p"` for **p**oints,
- `"l"` for **l**ines,
- `"b"` for **b**oth,
- `"c"` for the lines part alone of `"b"`,
- `"o"` for both '**o**verplotted',
- `"h"` for '**h**istogram' like (or 'high-density') vertical lines,
- `"s"` for stair **s**teps,
- `"S"` for other **s**teps, see 'Details' below,
- `"n"` for no plotting.

All other `types` give a warning or an error; using, e.g., `type = "punkte"` being equivalent to `type = "p"` for S compatibility. Note that some methods, e.g.`plot.factor`, do not accept this.

`main` an overall title for the plot: see `title`.

`sub` a sub title for the plot: see `title`.

`xlab` a title for the x axis: see `title`.

`ylab` a title for the y axis: see `title`.

`asp` the y/x aspect ratio, see `plot.window`.

Details

This plot method can be applied to `Simulations-summary` objects in order to summarize them graphically. Possible `type` of plots at the moment are those listed in `plot,GeneralSimulations-summary-method` plus:

meanFit Plot showing the average fitted dose-toxicity curve across the trials, together with 95% credible intervals, and comparison with the assumed truth (as specified by the `truth` argument to `summary,Simulations-method`)

You can specify any subset of these in the `type` argument.

Value

A single `ggplot2` object if a single plot is asked for, otherwise a `gridExtra{gTree}` object.

prob *Compute the probability for a given dose, given model and samples*

Description

Compute the probability for a given dose, given model and samples

Usage

```
prob(dose, model, samples, ...)
```

Arguments

`dose`	the dose
`model`	the `Model`
`samples`	the `Samples`
`...`	unused

prob,numeric,Model,Samples-method *Compute the probability for a given dose, given model and samples*

Description

Compute the probability for a given dose, given model and samples

Usage

```
## S4 method for signature 'numeric,Model,Samples'
prob(dose, model, samples, ...)
```

Arguments

`dose`	the dose
`model`	the `Model`
`samples`	the `Samples`
`...`	unused

Quantiles2LogisticNormal

Convert prior quantiles (lower, median, upper) to logistic (log) normal model

Description

This function uses generalised simulated annealing to optimise a LogisticNormal model to be as close as possible to the given prior quantiles.

Usage

```
Quantiles2LogisticNormal(dosegrid, refDose, lower, median, upper,
  level = 0.95, logNormal = FALSE, parstart = NULL, parlower = c(-10,
  -10, 0, 0, -0.95), parupper = c(10, 10, 10, 10, 0.95), verbose = TRUE,
  control = list(threshold.stop = 0.01, maxit = 50000, temperature = 50000,
  max.time = 600))
```

Arguments

dosegrid	the dose grid
refDose	the reference dose
lower	the lower quantiles
median	the medians
upper	the upper quantiles
level	the credible level of the (lower, upper) intervals (default: 0.95)
logNormal	use the log-normal prior? (not default) otherwise, the normal prior for the logistic regression coefficients is used
parstart	starting values for the parameters. By default, these are determined from the medians supplied.
parlower	lower bounds on the parameters (intercept alpha and the slope beta, the corresponding standard deviations and the correlation.)
parupper	upper bounds on the parameters
verbose	be verbose? (default)
control	additional options for the optimisation routine, see GenSA for more details

Value

a list with the best approximating `model` (LogisticNormal or LogisticLogNormal), the resulting `quantiles`, the `required` quantiles and the `distance` to the required quantiles, as well as the final `parameters` (which could be used for running the algorithm a second time)

Report

A Reference Class to represent sequentially updated reporting objects.

Description

A Reference Class to represent sequentially updated reporting objects.

`RuleDesign-class` *Class for rule-based designs*

Description

The difference to `Design` class is that model, stopping and increments slots are missing.

Slots

- `nextBest` how to find the next best dose, an object of class `NextBest`
- `cohortSize` rules for the cohort sizes, an object of class `CohortSize`
- `data` what is the dose grid, any previous data, etc., contained in an object of class `Data`
- `startingDose` what is the starting dose? Must lie on the grid in `data`

`Samples-class` *Class for the MCMC output*

Description

Class for the MCMC output

Slots

- `data` a list where each entry contains the samples of a (vector-valued) parameter in a vector/matrix in the format (number of samples) x (dimension of the parameter).
- `options` the `McmcOptions` which have been used

`sampleSize` *Compute the number of samples for a given MCMC options triple*

Description

Compute the number of samples for a given MCMC options triple

Usage

```
sampleSize(mcmcOptions)
```

Arguments

`mcmcOptions`	the `McmcOptions` object

Value

the resulting sample size

show,GeneralSimulations-summary-method

Show the summary of the simulations

Description

Show the summary of the simulations

Usage

```
## S4 method for signature 'GeneralSimulations-summary'
show(object)
```

Arguments

object	the GeneralSimulations-summary object we want to print

Value

invisibly returns a data frame of the results with one row and appropriate column names

show,Simulations-summary-method

Show the summary of the simulations

Description

Show the summary of the simulations

Usage

```
## S4 method for signature 'Simulations-summary'
show(object)
```

Arguments

object	the Simulations-summary object we want to print

Value

invisibly returns a data frame of the results with one row and appropriate column names `simulate,Design-method`

Simulate outcomes from a CRM design

Description

Simulate outcomes from a CRM design

Usage

```
## S4 method for signature 'Design'
simulate(object, truth, args = NULL,
  firstSeparate = FALSE, nsim = 1L, mcmcOptions = new("McmcOptions"),
  seed = NULL, parallel = FALSE, ...)
```

Arguments

`object`	the `Design` object we want to simulate data from
`truth`	a function which takes as input a dose (vector) and returns the true probability (vector) for toxicity. Additional arguments can be supplied in `args`.
`args`	data frame with arguments for the `truth` function. The column names correspond to the argument names, the rows to the values of the arguments. The rows are appropriately recycled in the `nsim` simulations. In order to produce outcomes from the posterior predictive distribution, e.g, pass an `object` that contains the data observed so far, `truth` contains the `prob` function from the model in `object`, and `args` contains posterior samples from the model.
`firstSeparate`	enroll the first patient separately from the rest of the cohort? (not default) If yes, the cohort will be closed if a DLT occurs in this patient.
`nsim`	the number of simulations (default: 1)
`mcmcOptions`	object of class `McmcOptions`, giving the MCMC options for each evaluation in the trial. By default, the standard options are used
`seed`	an object specifying if and how the random number generator should be initialized (“seeded”). Either `NULL` (default) or an integer that will be used in a call to `set.seed` before simulating the response vectors. If set, the value is saved as the `seed` slot of the returned object. The default, `NULL` will not change the random generator state, and `.Random.seed` will be saved.
`parallel`	should the simulation runs be parallelized across the clusters of the computer? (not default)
`...`	additional optional arguments.

Value

an object of class `Simulations`

simulate,RuleDesign-method *Simulate outcomes from a rule-based design*

Description

Simulate outcomes from a rule-based design

Usage

```
## S4 method for signature 'RuleDesign'
simulate(object, truth, args = NULL, nsim = 1L,
  seed = NULL, parallel = FALSE, ...)
```

Arguments

`object`	the RuleDesign object we want to simulate data from
`truth`	a function which takes as input a dose (vector) and returns the true probability (vector) for toxicity. Additional arguments can be supplied in `args`.
`args`	data frame with arguments for the `truth` function. The column names correspond to the argument names, the rows to the values of the arguments. The rows are appropriately recycled in the `nsim` simulations.
`nsim`	the number of simulations (default: 1)
`seed`	an object specifying if and how the random number generator should be initialized (“seeded”). Either `NULL` (default) or an integer that will be used in a call to `set.seed` before simulating the response vectors. If set, the value is saved as the `seed` slot of the returned object. The default, `NULL` will not change the random generator state, and `.Random.seed` will be saved.
`parallel`	should the simulation runs be parallelized across the clusters of the computer? (not default)
`...`	additional optional arguments.

Value

an object of class `GeneralSimulations`

Simulations-class *Class for the simulations output from model based designs*

Description

This class captures the trial simulations from model based designs. Additional slots fit and stopReasons compared to the general class.

Slots

`fit` list with the final fits

`stopReasons` list of stopping reasons for each simulation run

`Simulations-summary-class` *Class for the summary of model-based simulations output*

Description

Additional slots that require the model fits.

Slots

`fitAtDoseMostSelected` fitted toxicity rate at dose most often selected

`meanFit` list with the average, lower (2.5 quantiles of the mean fitted toxicity at each dose level

`size` *Determine the size of the next cohort*

Description

This function determines the size of the next cohort.

Usage

```
size(cohortSize, dose, data, ...)
```

Arguments

`cohortSize`	The rule, an object of class `CohortSize`
`dose`	the next dose
`data`	The data input, an object of class `Data`
`...`	additional arguments

Value

the size as integer value

size,CohortSizeConst,ANY,Data-method

Constant cohort size

Description

Constant cohort size

Usage

```
## S4 method for signature 'CohortSizeConst,ANY,Data'
size(cohortSize, dose, data, ...)
```

Arguments

cohortSize	The rule, an object of class CohortSize
dose	the next dose
data	The data input, an object of class Data
...	additional arguments

size,CohortSizeDLT,ANY,Data-method

Determine the cohort size based on the number of DLTs so far

Description

Determine the cohort size based on the number of DLTs so far

Usage

```
## S4 method for signature 'CohortSizeDLT,ANY,Data'
size(cohortSize, dose, data, ...)
```

Arguments

cohortSize	The rule, an object of class CohortSize
dose	the next dose
data	The data input, an object of class Data
...	additional arguments

`size,CohortSizeMax,ANY,Data-method` *Size based on maximum of multiple cohort size rules*

Description

Size based on maximum of multiple cohort size rules

Usage

```
## S4 method for signature 'CohortSizeMax,ANY,Data'
size(cohortSize, dose, data, ...)
```

Arguments

`cohortSize`	The rule, an object of class `CohortSize`
`dose`	the next dose
`data`	The data input, an object of class `Data`
`...`	additional arguments

`size,CohortSizeMin,ANY,Data-method` *Size based on minimum of multiple cohort size rules*

Description

Size based on minimum of multiple cohort size rules

Usage

```
## S4 method for signature 'CohortSizeMin,ANY,Data'
size(cohortSize, dose, data, ...)
```

Arguments

`cohortSize`	The rule, an object of class `CohortSize`
`dose`	the next dose
`data`	The data input, an object of class `Data`
`...`	additional arguments

size,CohortSizeParts,ANY,DataParts-method

Cohort size based on the parts

Description

Cohort size based on the parts

Usage

```
## S4 method for signature 'CohortSizeParts,ANY,DataParts'
size(cohortSize, dose, data, ...)
```

Arguments

cohortSize	The rule, an object of class CohortSize
dose	the next dose
data	The data input, an object of class Data
...	additional arguments

size,CohortSizeRange,ANY,Data-method

Determine the cohort size based on the range into which the next dose falls into

Description

Determine the cohort size based on the range into which the next dose falls into

Usage

```
## S4 method for signature 'CohortSizeRange,ANY,Data'
size(cohortSize, dose, data, ...)
```

Arguments

cohortSize	The rule, an object of class CohortSize
dose	the next dose
data	The data input, an object of class Data
...	additional arguments

Stopping-class *The virtual class for stopping rules*

Description

The virtual class for stopping rules

See Also

StoppingList, StoppingMaxPatients, StoppingCohortsNearDose, StoppingPatientsNearDose, StoppingMinCohorts, StoppingMinPatients, StoppingTargetProb StoppingMTDdistribution, StoppingTargetBiomarker

StoppingAll-class *Stop based on fullfillment of all multiple stopping rules*

Description

This class can be used to combine multiple stopping rules with an AND operator.

Details

stopList contains all stopping rules, which are again objects of class Stopping. All stopping rules must be fulfilled in order that the result of this rule is to stop.

Slots

stopList list of stopping rules of the stopping rules into a single result

StoppingAny-class *Stop based on fullfillment of any stopping rule*

Description

This class can be used to combine multiple stopping rules with an OR operator.

Details

stopList contains all stopping rules, which are again objects of class Stopping. Any of these rules must be fulfilled in order that the result of this rule is to stop.

Slots

stopList list of stopping rules of the stopping rules into a single result

StoppingCohortsNearDose-class *Stop based on number of cohorts near to next best dose*

Description

Stop based on number of cohorts near to next best dose

Slots

`nCohorts` number of required cohorts

`percentage` percentage (between 0 and 100) within the next best dose the cohorts must lie

StoppingList-class *Stop based on multiple stopping rules*

Description

This class can be used to combine multiple stopping rules.

Details

`stopList` contains all stopping rules, which are again objects of class `Stopping`, and the `summary` is a function taking a logical vector of the size of `stopList` and returning a single logical value. For example, if the function `all` is given as `summary` function, then this means that all stopping rules must be fulfilled in order that the result of this rule is to stop.

Slots

`stopList` list of stopping rules

`summary` the summary function to combine the results of the stopping rules into a single result

StoppingMaxPatients-class *Stop based on maximum number of patients*

Description

Stop based on maximum number of patients

Slots

`nPatients` maximum allowed number of patients

StoppingMinCohorts-class *Stop based on minimum number of cohorts*

Description

Stop based on minimum number of cohorts

Slots

`nCohorts` minimum required number of cohorts

StoppingMinPatients-class *Stop based on minimum number of patients*

Description

Stop based on minimum number of patients

Slots

`nPatients` minimum required number of patients

StoppingMTDdistribution-class *Stop based on MTD distribution*

Description

Has 90% probability above a threshold of 50% of the current MTD been reached? This class is used for this question.

Slots

`target` the target toxicity probability (e.g. 0.33) defining the MTD

`thresh` the threshold relative to the MTD (e.g. 0.5)

`prob` required probability (e.g. 0.9)

`StoppingPatientsNearDose-class` *Stop based on number of patients near to next best dose*

Description

Stop based on number of patients near to next best dose

Slots

`nPatients` number of required patients

`percentage` percentage (between 0 and 100) within the next best dose the patients must lie

`StoppingTargetBiomarker-class` *Stop based on probability of target biomarker*

Description

Stop based on probability of target biomarker

Slots

`target` the biomarker level, relative to the maximum, that needs to be reached

`prob` required target probability for reaching sufficient precision

`StoppingTargetProb-class` *Stop based on probability of target tox interval*

Description

Stop based on probability of target tox interval

Slots

`target` the target toxicity interval

`prob` required target toxicity probability for reaching sufficient precision

stopTrial *Stop the trial?*

Description

This function returns whether to stop the trial.

Usage

```
stopTrial(stopping, dose, samples, model, data, ...)
```

Arguments

`stopping`	The rule, an object of class `Stopping`
`dose`	the recommended next best dose
`samples`	the `Samples` object
`model`	The model input, an object of class `Model`
`data`	The data input, an object of class `Data`
`...`	additional arguments

Value

logical value: `TRUE` if the trial can be stopped, `FALSE` otherwise. It should have an attribute `message` which gives the reason for the decision.

stopTrial,StoppingAll,ANY,ANY,ANY,ANY-method *Stop based on fulfillment of all multiple stopping rules*

Description

Stop based on fulfillment of all multiple stopping rules

Usage

```
## S4 method for signature 'StoppingAll,ANY,ANY,ANY,ANY'
stopTrial(stopping, dose, samples,
  model, data, ...)
```

Arguments

`stopping`	The rule, an object of class `Stopping`
`dose`	the recommended next best dose
`samples`	the `Samples` object
`model`	The model input, an object of class `Model`
`data`	The data input, an object of class `Data`
`...`	additional arguments

stopTrial,StoppingAny,ANY,ANY,ANY,ANY-method

Stop based on fulfillment of any stopping rule

Description

Stop based on fulfillment of any stopping rule

Usage

```
## S4 method for signature 'StoppingAny,ANY,ANY,ANY,ANY'
stopTrial(stopping, dose, samples,
  model, data, ...)
```

Arguments

stopping	The rule, an object of class Stopping
dose	the recommended next best dose
samples	the Samples object
model	The model input, an object of class Model
data	The data input, an object of class Data
...	additional arguments

stopTrial,StoppingCohortsNearDose,numeric,ANY,ANY,Data-method

Stop based on number of cohorts near to next best dose

Description

Stop based on number of cohorts near to next best dose

Usage

```
## S4 method for signature 'StoppingCohortsNearDose,numeric,ANY,ANY,Data'
stopTrial(stopping,
  dose, samples, model, data, ...)
```

Arguments

stopping	The rule, an object of class Stopping
dose	the recommended next best dose
samples	the Samples object
model	The model input, an object of class Model
data	The data input, an object of class Data
...	additional arguments

stopTrial,StoppingList,ANY,ANY,ANY,ANY-method

Stop based on multiple stopping rules

Description

Stop based on multiple stopping rules

Usage

```
## S4 method for signature 'StoppingList,ANY,ANY,ANY,ANY'
stopTrial(stopping, dose, samples,
  model, data, ...)
```

Arguments

`stopping`	The rule, an object of class `Stopping`
`dose`	the recommended next best dose
`samples`	the `Samples` object
`model`	The model input, an object of class `Model`
`data`	The data input, an object of class `Data`
`...`	additional arguments

stopTrial,StoppingMaxPatients,ANY,ANY,ANY,Data-method

Stop based on maximum number of patients

Description

Stop based on maximum number of patients

Usage

```
## S4 method for signature 'StoppingMaxPatients,ANY,ANY,ANY,Data'
stopTrial(stopping, dose,
  samples, model, data, ...)
```

Arguments

`stopping`	The rule, an object of class `Stopping`
`dose`	the recommended next best dose
`samples`	the `Samples` object
`model`	The model input, an object of class `Model`
`data`	The data input, an object of class `Data`
`...`	additional arguments

stopTrial,StoppingMinCohorts,ANY,ANY,ANY,Data-method

Stop based on minimum number of cohorts

Description

Stop based on minimum number of cohorts

Usage

```
## S4 method for signature 'StoppingMinCohorts,ANY,ANY,ANY,Data'
stopTrial(stopping, dose,
  samples, model, data, ...)
```

Arguments

stopping	The rule, an object of class Stopping
dose	the recommended next best dose
samples	the Samples object
model	The model input, an object of class Model
data	The data input, an object of class Data
...	additional arguments

stopTrial,StoppingMinPatients,ANY,ANY,ANY,Data-method

Stop based on minimum number of patients

Description

Stop based on minimum number of patients

Usage

```
## S4 method for signature 'StoppingMinPatients,ANY,ANY,ANY,Data'
stopTrial(stopping, dose,
  samples, model, data, ...)
```

Arguments

stopping	The rule, an object of class Stopping
dose	the recommended next best dose
samples	the Samples object
model	The model input, an object of class Model
data	The data input, an object of class Data
...	additional arguments

stopTrial,StoppingMTDdistribution,numeric,Samples,Model,ANY-method

Stop based on MTD distribution

Description

Stop based on MTD distribution

Usage

```
 ## S4 method for signature 'StoppingMTDdistribution,numeric,Samples,Model,ANY'
stopTrial(stopping,
  dose, samples, model, data, ...)
```

Arguments

stopping	The rule, an object of class Stopping
dose	the recommended next best dose
samples	the Samples object
model	The model input, an object of class Model
data	The data input, an object of class Data
...	additional arguments

stopTrial,StoppingPatientsNearDose,numeric,ANY,ANY,Data-method

Stop based on number of patients near to next best dose

Description

Stop based on number of patients near to next best dose

Usage

```
## S4 method for signature 'StoppingPatientsNearDose,numeric,ANY,ANY,Data'
stopTrial(stopping,
  dose, samples, model, data, ...)
```

Arguments

stopping	The rule, an object of class Stopping
dose	the recommended next best dose
samples	the Samples object
model	The model input, an object of class Model
data	The data input, an object of class Data
...	additional arguments

stopTrial,StoppingTargetBiomarker,numeric,Samples,DualEndpoint,ANY-method

Stop based on probability of targeting biomarker

Description

Stop based on probability of targeting biomarker

Usage

```
  ## S4 method for signature
## 'StoppingTargetBiomarker,numeric,Samples,DualEndpoint,ANY'
stopTrial(stopping,
  dose, samples, model, data, ...)
```

Arguments

stopping	The rule, an object of class Stopping
dose	the recommended next best dose
samples	the Samples object
model	The model input, an object of class Model
data	The data input, an object of class Data
...	additional arguments

stopTrial,StoppingTargetProb,numeric,Samples,Model,ANY-method

Stop based on probability of target tox interval

Description

Stop based on probability of target tox interval

Usage

```
## S4 method for signature 'StoppingTargetProb,numeric,Samples,Model,ANY'
stopTrial(stopping,
  dose, samples, model, data, ...)
```

Arguments

stopping	The rule, an object of class Stopping
dose	the recommended next best dose
samples	the Samples object
model	The model input, an object of class Model
data	The data input, an object of class Data
...	additional arguments

summary,GeneralSimulations-method

Summarize the simulations, relative to a given truth

Description

Summarize the simulations, relative to a given truth

Usage

```
## S4 method for signature 'GeneralSimulations'
summary(object, truth, target = c(0.2, 0.35),
  ...)
```

Arguments

`object`	the `GeneralSimulations` object we want to summarize
`truth`	a function which takes as input a dose (vector) and returns the true probability (vector) for toxicity. Additional arguments can be supplied via ....
`target`	the target toxicity interval (default: 20-35%) used for the computations
`...`	additional arguments affecting the summary produced.

Value

an object of class `GeneralSimulations-summary`

summary,Simulations-method

Summarize the model-based design simulations, relative to a given truth

Description

Summarize the model-based design simulations, relative to a given truth

Usage

```
## S4 method for signature 'Simulations'
summary(object, truth, target = c(0.2, 0.35), ...)
```

Arguments

`object`	an object for which a summary is desired.
`...`	additional arguments affecting the summary produced.

Value

an object of class `Simulations-summary`

update,Data-method *Update method for the "Data" class*

Description

Add new data to the Data object

Usage

```
## S4 method for signature 'Data'
update(object, x, y, ID = (if (length(object@ID))
  max(object@ID) else 0L) + seq_along(y), ...)
```

Arguments

object	the old Data object
x	the dose level (one level only!)
y	the DLT vector (0/1 vector), for all patients in this cohort
ID	the patient IDs
...	Additional arguments to the call, or arguments with changed values. Use name = NULL to remove the argument name.

Value

the new Data object

update,DataParts-method *Update method for the "DataParts" class*

Description

Add new data to the DataParts object

Usage

```
## S4 method for signature 'DataParts'
update(object, x, y, ID = (if (length(object@ID))
  max(object@ID) else 0L) + seq_along(y), ...)
```

Arguments

object	the old DataParts object
x	the dose level (one level only!)
y	the DLT vector (0/1 vector), for all patients in this cohort
ID	the patient IDs
...	Additional arguments to the call, or arguments with changed values. Use name = NULL to remove the argument name.

Value

the new DataParts object

&,Stopping,Stopping-method

The method combining two atomic stopping rules

Description

The method combining two atomic stopping rules

Usage

```
## S4 method for signature 'Stopping,Stopping'
e1 & e2
```

Arguments

e1	First Stopping object
e2	Second Stopping object

Value

The StoppingAll object

&,Stopping,StoppingAll-method

The method combining an atomic and a stopping list

Description

The method combining an atomic and a stopping list

Usage

```
## S4 method for signature 'Stopping,StoppingAll'
e1 & e2
```

Arguments

e1	Stopping object
e2	StoppingAll object

Value

The modified StoppingAll object

&,StoppingAll,Stopping-method

The method combining a stopping list and an atomic

Description

The method combining a stopping list and an atomic

Usage

```
## S4 method for signature 'StoppingAll,Stopping'
e1 & e2
```

Arguments

e1	StoppingAll object
e2	Stopping object

Value

The modified StoppingAll object

Index