Package 'BIGL'

May 5, 2022

Type Package

Title Biochemically Intuitive Generalized Loewe Model

Version 1.6.6

Date 2022-05-04

Author Heather Turner, Annelies Tourny, Olivier Thas, Maxim Nazarov, Rytis Bagdziunas, Stijn Hawinkel

Maintainer Maxim Nazarov <maxim.nazarov@openanalytics.eu>

Description Response surface methods for drug synergy analysis. Available methods include generalized and classical Loewe formulations as well as Highest Single Agent methodology. Response surfaces can be plotted in an interactive 3-D plot and formal statistical tests for presence of synergistic effects are available. Implemented methods and tests are described in the article ``BIGL: Biochemically Intuitive Generalized Loewe null model for prediction of the expected combined effect compatible with partial agonism and antagonism" by Koen Van der Borght, Annelies Tourny, Rytis Bagdziunas, Olivier Thas, Maxim Nazarov, Heather Turner, Bie Verbist & Hugo Ceulemans (2017) <doi:10.1038/s41598-017-18068-5>.

License GPL-3

Depends R (>= 3.5)

Imports ggplot2, MASS, methods, minpack.lm, numDeriv, parallel, progress, rgl, robustbase, scales, nleqslv

Suggests knitr, rmarkdown, testthat, shiny, DT

VignetteBuilder knitr, rmarkdown

LazyData true

URL https://github.com/openanalytics/BIGL

BugReports https://github.com/openanalytics/BIGL/issues

Encoding UTF-8

RoxygenNote 7.1.2

NeedsCompilation no

Repository CRAN

Date/Publication 2022-05-05 07:00:06 UTC

R topics documented:

`addResids` *Add residuals by adding to mean effects*

Description

Add residuals by adding to mean effects

Usage

```
addResids(means, ...)
```

Arguments

`means`	a vector of means
`...`	passed on to predictVar

backscaleResids *Backscale residuals*

Description

Backscale residuals

Usage

```
backscaleResids(scaledResids, ...)
```

Arguments

scaledResids	scaled residuals
...	passed on to predictVar

Blissindependence *Bliss Independence Model*

Description

This function returns fractional response levels for when these are based on Bliss Independence Model.

Usage

```
Blissindependence(doseInput, parmInput, ...)
```

Arguments

doseInput	Dose-response dataframe containing "d1" and "d2" columns
parmInput	Numeric vector or list with appropriately named parameter inputs. Typically, it will be coefficients from a MarginalFit object.
...	Further arguments that are currently unused

`bootConfInt` *Obtain confidence intervals for the raw effect sizes on every off-axis point and overall*

Description

Obtain confidence intervals for the raw effect sizes on every off-axis point and overall

Usage

```
bootConfInt(
  Total,
  idUnique,
  bootStraps,
  transforms,
  respS,
  B.B,
  method,
  CP,
  reps,
  n1,
  cutoff,
  R,
  fitResult,
  bootRS,
  data_off,
  posEffect = all(Total$effect >= 0),
  transFun,
  invTransFun,
  model,
  rescaleResids,
  ...
)
```

Arguments

`Total`	data frame with all effects and mean effects
`idUnique`	unique combinations of on-axis points, a character vector
`bootStraps`	precomputed bootstrap objects
`transforms`	Transformation functions. If non-null, `transforms` is a list containing 5 elements, namely biological and power transformations along with their inverse functions and `compositeArgs` which is a list with argument values shared across the 4 functions. See vignette for more information.
`respS`	the observed response surface
`B.B`	Number of iterations to use in bootstrapping null distribution for either meanR or maxR statistics.

method	What assumption should be used for the variance of on- and off-axis points. This argument can take one of the values from c("equal", "model", "unequal"). With the value "equal" as the default. "equal" assumes that both on- and off-axis points have the same variance, "unequal" estimates a different parameter for on- and off-axis points and "model" predicts variance based on the average effect of an off-axis point. If no transformations are used the "model" method is recommended. If transformations are used, only the "equal" method can be chosen.
CP	Prediction covariance matrix. If not specified, it will be estimated by bootstrap using B.CP iterations.
reps	Numeric vector containing number of replicates for each off-axis dose combination. If missing, it will be calculated automatically from output of predictOffAxis function.
n1	the number of off-axis points
cutoff	Cut-off to use in maxR procedure for declaring non-additivity (default is 0.95).
R	Numeric vector containing mean deviation of predicted response surface from the observed one at each of the off-axis points. If missing, it will be calculated automatically from output of predictOffAxis function.
fitResult	Monotherapy (on-axis) model fit, e.g. produced by fitMarginals. It has to be a "MarginalFit" object or a list containing df, sigma, coef, shared_asymptote and method elements for, respectively, marginal model degrees of freedom, residual standard deviation, named vector of coefficient estimates, logical value of whether shared asymptote is imposed and method for estimating marginal models during bootstrapping (see fitMarginals). If biological and power transformations were used in marginal model estimation, fitResult should contain transforms elements with these transformations. Alternatively, these can also be specified via transforms argument.
bootRS	a boolean, should bootstrapped response surfaces be used in the calculation of the confidence intervals?
data_off	data frame with off -axis information
posEffect	a boolean, are effects restricted to be positive
transFun	the transformation and inverse transformation functions for the variance
invTransFun	the transformation and inverse transformation functions for the variance
model	The mean-variance model
rescaleResids	a boolean indicating whether to rescale residuals, or else normality of the residuals is assumed.
...	Further arguments that will be later passed to generateData function during bootstrapping

Value

A list with components

offAxis	The off-axis bootstrapped confidence intervals
single	A mean effect and percentile and studentized boostrap intervals

boxcox.transformation *Apply two-parameter Box-Cox transformation*

Description

Apply two-parameter Box-Cox transformation

Usage

```
boxcox.transformation(y, lambda, alpha = 0)
```

Arguments

y	Numeric vector
lambda	Power parameter in power transform
alpha	Shift paramater in 2-parameter power transform. Defaults to 0 which implies a 1-parameter Box-Cox transform.

Value

Power-transformed data

coef.MarginalFit *Coefficients from marginal model estimation*

Description

Coefficients from marginal model estimation

Usage

```
## S3 method for class 'MarginalFit'
coef(object, ...)
```

Arguments

object	Output of fitMarginals function
...	Further arguments

`constructFormula` *Construct a model formula from parameter constraint matrix*

Description

For parameter names defined in `naming` vector, formula is constructed so that `consMatrix %*% naming = consVector` is satisfied. Constraint coefficients are normalized and convert into fractions.

Usage

```
constructFormula(
  consMatrix = NULL,
  consVector = NULL,
  naming = c("h1", "h2", "b", "m1", "m2", "e1", "e2"),
  extraVars = c("d1", "d2"),
  formulaArgs = c("effect", "fn")
)
```

Arguments

`consMatrix`	Constraint matrix
`consVector`	Constraint vector
`naming`	Parameter names
`extraVars`	Non-parameter variables used in the formula and function evaluation. These will be appended to the formula.
`formulaArgs`	Character vector of length two. First element indicates name for the response variable. Second element indicates name of the function.

Value

This function returns a model construct appropriate for `fitMarginals` function. It also separates variables into those that are free and those which are constrained.

Examples

```
constM <- rbind(c(0, 0, 1, 0, 0, 0, 0),
                c(0, 0, 0, -1, 1, 0, 0))
constV <- c(0.9, 0)
constructFormula(constM, constV)
```

`contour.ResponseSurface` *Method for plotting of contours based on maxR statistics*

Description

Method for plotting of contours based on maxR statistics

Usage

```
## S3 method for class 'ResponseSurface'
contour(x, ...)
```

Arguments

`x`	Output of `fitSurface`
`...`	Further parameters passed to `plot.maxR`

`df.residual.MarginalFit` *Residual degrees of freedom in marginal model estimation*

Description

Residual degrees of freedom in marginal model estimation

Usage

```
## S3 method for class 'MarginalFit'
df.residual(object, ...)
```

Arguments

`object`	Output of `fitMarginals` function
`...`	Further arguments

directAntivirals *Partial data with combination experiments of direct-acting antivirals*

Description

A dataset containing 11 combination experiments of direct-acting antivirals.

Format

A data frame with 3520 rows and 6 variables:

- experiment: ID of experiment (1-11)
- cpd1: name of the first compound (4 different compounds)
- cpd2: name of the second compound (11 different compounds)
- effect: observed effect (cell count)
- d1: dose of the first compound
- d2: dose of the second compound

directAntivirals_ALL *Full data with combination experiments of direct-acting antivirals*

Description

A dataset containing 11 combination experiments of direct-acting antivirals. This dataset is larger than `directAntivirals` dataset as it includes concentrations at levels of `1e6` which can render plots visually unappealing.

Format

A data frame with 4224 rows and 6 variables:

- experiment: ID of experiment (1-11)
- cpd1: name of the first compound (4 different compounds)
- cpd2: name of the second compound (11 different compounds)
- effect: observed effect (cell count)
- d1: dose of the first compound
- d2: dose of the second compound

`fitMarginals` *Fit two 4-parameter log-logistic functions for a synergy experiment*

Description

This function uses dose-response data for two compounds and estimates coefficients for monotherapy models of both of these compounds such that they share a common baseline. Currently, these coefficients are estimated by default using a non-linear least squares approximation. Although entire dose-response data can be provided, estimation will subset the part of data where at least one of the compounds is dosed at zero, i.e. on-axis data.

Usage

```
fitMarginals(
  data,
  transforms = NULL,
  start = NULL,
  constraints = NULL,
  fixed = NULL,
  method = c("nlslm", "nls", "optim"),
  names = NULL,
  ...
)
```

Arguments

`data`	Dose-response dataframe. Marginal data will be extracted from it automatically.
`transforms`	Transformation functions. If non-null, `transforms` is a list containing 5 elements, namely biological and power transformations along with their inverse functions and `compositeArgs` which is a list with argument values shared across the 4 functions. See vignette for more information.
`start`	Starting parameter values. If not specified, they will be obtained from `initialMarginal`.
`constraints`	List of constraint matrix and vector which will be passed to `constructFormula`. If `constraints = NULL`, no constraints on parameter estimation will be imposed.
`fixed`	This arguments provides a user-friendly alternative to impose a fixed value for marginal parameters. It must be a named vector with names contained in `c("h1", "h2", "b", "m1", "m2", "e1", "e2")`. For example, `fixed = c("m1" = 1, "h1" = 1)` will automatically generate appropriate constraint matrix and vector to set the maximal response and the Hill coefficient of the first compound to 1. If both `constraints` and `fixed` arguments are passed, then only `fixed` will be used.
`method`	Which estimation method should be used to obtain the estimates. If `method = "nls"`, simple non-linear least squares `nls` will be used. If `method = "nlslm"` Levenberg-Marquardt non-linear least squares `nlsLM` is used instead (default). If `method = "optim"`, residual sum of squares will be minimized using general purpose optimization based on Nelder-Mean algorithm in `optim`. This method can be noticeably slower than the non-linear least squares methods.

names	Compound names to be used on the plot labels.
...	Further arguments that are passed to the optimizer function, such as lower or upper (for the "nlslm" method), or control.

Details

Model formula is specified as effect ~ fn(h1, h2, ...) where fn is a hard-coded function which fits two 4-parameter log-logistic functions simultaneously so that the baseline can be shared. If transformation functions are provided, fn is consequently adjusted to account for them.

Value

This function returns a MarginalFit object with monotherapy coefficient estimates and diverse information regarding monotherapy estimation. MarginalFit object is essentially a list with appropriately named elements.

Among these list elements, "coef" is a named vector with parameter estimates. h1 and h2 are Hill's slope coefficients for each of the compounds, m1 and m2 are their maximal response levels whereas b is the shared baseline. Lastly, e1 and e2 are log-transformed EC50 values.

"sigma" is standard deviation of residuals for the estimated monotherapy model and "df" is the degrees of freedom for the residuals. "vcov" is the variance-covariance matrix of the estimated parameters.

Return object also contains information regarding data, biological and power transformations used in this estimation as well as model construct and method of estimation.

Examples

```
data <- subset(directAntivirals, experiment == 1)
## Data must contain d1, d2 and effect columns
transforms <- getTransformations(data)
fitMarginals(data, transforms)
```

---

fitSurface *Fit response surface model and compute meanR and maxR statistics*

Description

This function computes predictions for off-axis dose combinations according to the BIGL or HSA null model and, if required, computes appropriate meanR and maxR statistics. Function requires as input dose-response dataframe and output of fitMarginals containing estimates for the monotherapy model. If transformation functions were used in monotherapy estimation, these should also be provided.

Usage

```
fitSurface(
  data,
  fitResult,
  transforms = fitResult$transforms,
  null_model = c("loewe", "hsa", "bliss", "loewe2"),
  effect = "effect",
  d1 = "d1",
  d2 = "d2",
  statistic = c("none", "meanR", "maxR", "both"),
  CP = NULL,
  B.CP = 50,
  B.B = NULL,
  nested_bootstrap = FALSE,
  error = 4,
  sampling_errors = NULL,
  wild_bootstrap = FALSE,
  cutoff = 0.95,
  parallel = FALSE,
  progressBar = TRUE,
  method = c("equal", "model", "unequal"),
  confInt = TRUE,
  bootRS = TRUE,
  trans = "identity",
  rescaleResids = FALSE,
  invtrans = switch(trans, identity = "identity", log = "exp"),
  newtonRaphson = FALSE,
  asymptotes = 2,
  bootmethod = method
)
```

Arguments

`data`	Dose-response dataframe.
`fitResult`	Monotherapy (on-axis) model fit, e.g. produced by `fitMarginals`. It has to be a `"MarginalFit"` object or a list containing `df`, `sigma`, `coef`, `shared_asymptote` and `method` elements for, respectively, marginal model degrees of freedom, residual standard deviation, named vector of coefficient estimates, logical value of whether shared asymptote is imposed and method for estimating marginal models during bootstrapping (see `fitMarginals`). If biological and power transformations were used in marginal model estimation, `fitResult` should contain `transforms` elements with these transformations. Alternatively, these can also be specified via `transforms` argument.
`transforms`	Transformation functions. If non-null, `transforms` is a list containing 5 elements, namely biological and power transformations along with their inverse functions and `compositeArgs` which is a list with argument values shared across the 4 functions. See vignette for more information.

`null_model` Specified null model for the expected response surface. Currently, allowed options are `"loewe"` for generalized Loewe model, `"hsa"` for Highest Single Agent model, `"bliss"` for Bliss additivity, and `"loewe2"` for the alternative Loewe generalization.

`effect` Name of the response column in the data ("effect")

`d1` Name of the column with doses of the first compound ("d1")

`d2` Name of the column with doses of the second compound ("d2")

`statistic` Which statistics should be computed. This argument can take one of the values from `c("none", "meanR", "maxR", "both")`.

`CP` Prediction covariance matrix. If not specified, it will be estimated by bootstrap using `B.CP` iterations.

`B.CP` Number of bootstrap iterations to use for CP matrix estimation

`B.B` Number of iterations to use in bootstrapping null distribution for either meanR or maxR statistics.

`nested_bootstrap` When statistics are calculated, if `nested_bootstrap = TRUE`, `CP` matrix is recalculated at each bootstrap iteration of `B.B` using `B.CP` iterations. Using such nested bootstrap may however significantly increase computational time. If `nested_bootstrap = FALSE`, `CP` bootstrapped data reuses `CP` matrix calculated from the original data.

`error` Type of error for resampling in the bootstrapping procedure. This argument will be passed to `generateData`. If `error = 4` (default), the error terms for generating distribution of the null will be resampled from the vector specified in `sampling_errors`. If `error = 1`, normal errors are added. If `error = 2`, errors are sampled from a mixture of two normal distributions. If `error = 3`, errors are generated from a rescaled chi-square distribution.

`sampling_errors` Sampling vector to resample errors from. Used only if `error` is 4 and is passed as argument to `generateData`. If `sampling_errors = NULL` (default), mean residuals at off-axis points between observed and predicted response are taken.

`wild_bootstrap` Whether special bootstrap to correct for heteroskedasticity should be used. If `wild_bootstrap = TRUE`, errors are generated from `sampling_errors` multiplied by a random variable following Rademacher distribution. Argument is used only if `error = 4`.

`cutoff` Cut-off to use in maxR procedure for declaring non-additivity (default is 0.95).

`parallel` Whether parallel computing should be used for bootstrap. This parameter can take either integer value to specify the number of threads to be used or logical `TRUE/FALSE`. If `parallel = TRUE`, then `max(1, detectCores()-1)` is set to be the number of threads. If `parallel = FALSE`, then a single thread is used and cluster object is not created.

`progressBar` A boolean, should progress of bootstraps be shown?

`method` What assumption should be used for the variance of on- and off-axis points. This argument can take one of the values from `c("equal", "model", "unequal")`. With the value `"equal"` as the default. `"equal"` assumes that both on- and off-axis points have the same variance, `"unequal"` estimates a different parameter

	for on- and off-axis points and "model" predicts variance based on the average effect of an off-axis point. If no transformations are used the "model" method is recommended. If transformations are used, only the "equal" method can be chosen.
confInt	a boolean, should confidence intervals be returned?
bootRS	a boolean, should bootstrapped response surfaces be used in the calculation of the confidence intervals?
trans, invtrans	the transformation function for the variance and its inverse, possibly as strings
rescaleResids	a boolean indicating whether to rescale residuals, or else normality of the residuals is assumed.
newtonRaphson	A boolean, should Newton-Raphson be used to find Loewe response surfaces? May be faster but also less stable to switch on
asymptotes	Number of asymptotes. It can be either 1 as in standard Loewe model or 2 as in generalized Loewe model.
bootmethod	The resampling method to be used in the bootstraps. Defaults to the same as method

Details

Please see the example vignette `vignette("analysis", package = "BIGL")` and the report "Lack of fit test for detecting synergy" included in the `papers` folder for further details on the test statistics used: `system.file("papers", "newStatistics.pdf", package = "BIGL")`

Value

This function returns a `ResponseSurface` object with estimates of the predicted surface. `ResponseSurface` object is essentially a list with appropriately named elements.

Elements of the list include input data, monotherapy model coefficients and transformation functions, null model used to construct the surface as well as estimated CP matrix, occupancy level at each dose combination according to the generalized Loewe model and `"offAxisTable"` element which contains observed and predicted effects as well as estimated z-scores for each dose combination.

If statistical testing was done, returned object contains `"meanR"` and `"maxR"` elements with output from `meanR` and `maxR` respectively.

Examples

```
## Not run:
  data <- subset(directAntivirals, experiment == 4)
  ## Data should contain d1, d2 and effect columns
  transforms <- list("PowerT" = function(x, args) with(args, log(x)),
                     "InvPowerT" = function(y, args) with(args, exp(y)),
                     "BiolT" = function(x, args) with(args, N0 * exp(x * time.hours)),
                    "InvBiolT" = function(y, args) with(args, 1/time.hours * log(y/N0)),
                     "compositeArgs" = list(N0 = 1, time.hours = 72))
  fitResult <- fitMarginals(data, transforms)
```

```
  surf <- fitSurface(data, fitResult, statistic = "meanR")
  summary(surf)

## End(Not run)
```

`fitted.MarginalFit` *Compute fitted values from monotherapy estimation*

Description

Compute fitted values from monotherapy estimation

Usage

```
## S3 method for class 'MarginalFit'
fitted(object, ...)
```

Arguments

`object`	Output of `fitMarginals` function
`...`	Further arguments

`fitted.ResponseSurface` *Predicted values of the response surface according to the given null model*

Description

Predicted values of the response surface according to the given null model

Usage

```
## S3 method for class 'ResponseSurface'
fitted(object, ...)
```

Arguments

`object`	Output of `fitSurface`
`...`	Further parameters

generalizedLoewe *Compute combined predicted response from drug doses according to standard or generalized Loewe model.*

Description

Compute combined predicted response from drug doses according to standard or generalized Loewe model.

Usage

```
generalizedLoewe(
  doseInput,
  parmInput,
  asymptotes = 2,
  startvalues = NULL,
  newtonRaphson = FALSE,
  ...
)
```

Arguments

doseInput	Dose-response dataframe containing "d1" and "d2" columns
parmInput	Numeric vector or list with appropriately named parameter inputs. Typically, it will be coefficients from a MarginalFit object.
asymptotes	Number of asymptotes. It can be either 1 as in standard Loewe model or 2 as in generalized Loewe model.
startvalues	Starting values for the non-linear equation, from the observed data
newtonRaphson	a boolean, is Newton raphson used for finding the response surface? May be faster but also less stable
...	Further arguments that are currently unused

generateData *Generate data from parameters of marginal monotherapy model*

Description

This function is used to generate data for bootstrapping of the null distribution for various estimates. Optional arguments such as specific choice of sampling vector or corrections for heteroskedasticity can be specified in the function arguments.

Usage

```
generateData(
  pars,
  sigma,
  data = NULL,
  transforms = NULL,
  null_model = c("loewe", "hsa", "bliss", "loewe2"),
  error = 1,
  sampling_errors = NULL,
  means = NULL,
  model = NULL,
  method = "equal",
  wild_bootstrap = FALSE,
  rescaleResids,
  invTransFun,
  newtonRaphson = FALSE,
  bootmethod = method,
  ...
)
```

Arguments

pars	Coefficients of the marginal model along with their appropriate naming scheme. These will typically be estimated using fitMarginals. Futhermore, pars can simply be a MarginalFit object and transforms object will be automatically extracted.
sigma	Standard deviation to use for randomly generated error terms. This argument is unused if error = 4 so that sampling error vector is provided.
data	Data frame with dose columns ("d1", "d2") to generate the effect for. Only "d1" and "d2" columns of the dose-response dataframe should be passed to this argument. "effect" column should not be passed and if it is, the column will be replaced by simulated data.
transforms	Transformation functions. If non-null, transforms is a list containing 5 elements, namely biological and power transformations along with their inverse functions and compositeArgs which is a list with argument values shared across the 4 functions. See vignette for more information.
null_model	Specified null model for the expected response surface. Currently, allowed options are "loewe" for generalized Loewe model, "hsa" for Highest Single Agent model, "bliss" for Bliss additivity, and "loewe2" for the alternative Loewe generalization.
error	Type of error for resampling. error = 1 (Default) adds normal errors to the simulated effects, error = 2 adds errors sampled from a mixture of two normal distributions, error = 3 generates errors from a rescaled chi-square distribution. error = 4 will use bootstrap. Choosing this option, the error terms will be resampled from the vector specified in sampling_errors.
sampling_errors	Sampling vector to resample errors from. Used only if error = 4.

means	The vector of mean values of the response surface, for variance modelling
model	The mean-variance model
method	What assumption should be used for the variance of on- and off-axis points. This argument can take one of the values from c("equal", "model", "unequal"). With the value "equal" as the default. "equal" assumes that both on- and off-axis points have the same variance, "unequal" estimates a different parameter for on- and off-axis points and "model" predicts variance based on the average effect of an off-axis point. If no transformations are used the "model" method is recommended. If transformations are used, only the "equal" method can be chosen.
wild_bootstrap	Whether special bootstrap to correct for heteroskedasticity should be used. If wild_bootstrap = TRUE, errors are generated from sampling_errors multiplied by a random variable following Rademacher distribution. Argument is used only if error = 4.
rescaleResids	a boolean indicating whether to rescale residuals, or else normality of the residuals is assumed.
invTransFun	the inverse transformation function, back to the variance domain
newtonRaphson	A boolean, should Newton-Raphson be used to find Loewe response surfaces? May be faster but also less stable to switch on
bootmethod	The resampling method to be used in the bootstraps. Defaults to the same as method
...	Further arguments

Value

Dose-response dataframe with generated data including "effect" as well as "d1" and "d2" columns.

Examples

```
coefs <- c("h1" = 1, "h2" = 1.5, "b" = 0,
           "m1" = 1, "m2" = 2, "e1" = 0.5, "e2" = 0.1)

## Dose levels are set to be integers from 0 to 10
generateData(coefs, sigma = 1)

## Dose levels are taken from existing dataset with d1 and d2 columns
data <- subset(directAntivirals, experiment == 1)
generateData(data = data[, c("d1", "d2")], pars = coefs, sigma = 1)
```

---

get.abs_tval *Return absolute t-value, used in optimization call in* optim.boxcox

---

Description

Return absolute t-value, used in optimization call in optim.boxcox

Usage

```
get.abs_tval(value, fac, lambda, zero.add2 = 0)
```

Arguments

value	data
fac	factor
lambda	box-cox parameter
zero.add2	2nd box-cox parameter

---

get.summ.data *Summarize data by factor*

Description

Summarize data by factor

Usage

```
get.summ.data(value, fac)
```

Arguments

value	data to sumamrize
fac	factor to summarize by

---

getCP *Estimate CP matrix from bootstraps*

Description

This function is generally called from within fitSurface.

Usage

```
getCP(bootStraps, null_model, transforms, sigma0, doseGrid)
```

Arguments

bootStraps	the bootstraps carried out already
null_model	Specified null model for the expected response surface. Currently, allowed options are "loewe" for generalized Loewe model, "hsa" for Highest Single Agent model, "bliss" for Bliss additivity, and "loewe2" for the alternative Loewe generalization.
transforms	Transformation functions. If non-null, transforms is a list containing 5 elements, namely biological and power transformations along with their inverse functions and compositeArgs which is a list with argument values shared across the 4 functions. See vignette for more information.
sigma0	standard deviation of the null model on the real data
doseGrid	a grid of dose combinations

Value

Estimated CP matrix

---

getd1d2 *A function to get the d1d2 identifier*

Description

A function to get the d1d2 identifier

Usage

```
getd1d2(dat)
```

Arguments

dat	the data frame containing d1 and d2 entries

Value

a vector of d1d2 identifiers

getR *Helper functions for the test statistics*

Description

Helper functions for the test statistics

Usage

```
getR(data, idUnique, transforms, respS)
```

Arguments

data	the datasets
idUnique	id of unique off axis points
transforms	Transformation functions. If non-null, transforms is a list containing 5 elements, namely biological and power transformations along with their inverse functions and compositeArgs which is a list with argument values shared across the 4 functions. See vignette for more information.
respS	the evaluated response surface

GetStartGuess *Estimate initial values for dose-response curve fit*

Description

Estimate initial values for dose-response curve fit

Usage

```
GetStartGuess(df, transforms = NULL)
```

Arguments

df	Dose-response dataframe containing "dose" and "effect" columns
transforms	Transformation functions. If non-null, transforms is a list containing 5 elements, namely biological and power transformations along with their inverse functions and compositeArgs which is a list with argument values shared across the 4 functions. See vignette for more information.

getTransformations *Return a list with transformation functions*

Description

This function takes in response data from a dose-response model and attempts to find an optimal Box-Cox power transform based on optim.boxcox function. It then returns a list of transformation functions which contains this power transform and its inverse which can be subsequently used in fitMarginals and fitSurface.

Usage

```
getTransformations(data, shift = FALSE, args = list(N0 = 1, time.hours = 1))
```

Arguments

data	Dose-response dataframe.
shift	If TRUE or is a numeric value, then a two-parameter Box-Cox transformation is assumed. This parameter will be passed on to optim.boxcox function.
args	List with elements that are added to the list of transformation function and which can be used by these functions. In particular, this list should be of type args = list("N0" = 1, "time.hours" = 1) where N0 and time.hours are arguments used for the biological transform.

Details

Additionally, returned list contains biological transform and its inverse based on a simple exponential growth model, especially useful when response data is provided in cell counts. User can additionally provide arguments for these biological transforms where N0 stands for initial cell count and time.hours indicates number in hours after which response data was measured.

getTransformations relies on optim.boxcox to obtain the optimal Box-Cox transformation parameters. However, optim.boxcox optimizes for the power parameter only within the interval (0.1, 0.9). Hence, if obtained power parameter is close to 0.1, then a logarithmic transformation is applied instead.

Value

This function returns a list with transformation functions. These include power transformation ("PowerT") and its inverse ("InvPowerT") as well as biological transformation ("BiolT") and its inverse ("InvBiolT").

Power transformation is a 1-parameter Box-Cox transformation. If shift = TRUE, then power transformation is a 2-parameter Box-Cox transformation. Optimal values for power and shift operators are selected by means of optim.boxcox function.

Biological transformation y = N0 * exp(x * t) where N0 is the initial cell count and t is the incubation time. If response/effect variable (y) is given in terms of cell counts, biological transformation ensures that modelisation is done for the growth rate instead (x).

Returned list also contains "compositeArgs" elements shared by all the transformation functions. These arguments include initial cell count ("N0") and incubation time ("time.hours").

Examples

```
data <- subset(directAntivirals, experiment == 1)
## Data must contain d1, d2 and effect columns
getTransformations(data)
```

---

harbronLoewe *Alternative Loewe generalization*

Description

Alternative Loewe generalization

Usage

```
harbronLoewe(
  doseInput,
  parmInput,
  asymptotes = 2,
  startvalues = NULL,
  newtonRaphson = FALSE,
  ...
)
```

Arguments

doseInput	Dose-response dataframe containing "d1" and "d2" columns
parmInput	Numeric vector or list with appropriately named parameter inputs. Typically, it will be coefficients from a MarginalFit object.
asymptotes	Number of asymptotes. It can be either 1 as in standard Loewe model or 2 as in generalized Loewe model.
startvalues	Starting values for the non-linear equation, from the observed data
newtonRaphson	a boolean, is Newton raphson used for finding the response surface? May be faster but also less stable
...	Further arguments that are currently unused

hsa *Highest Single Agent model*

Description

This function returns response levels for when these are based on Highest Single Agent (HSA) model.

Usage

```
hsa(doseInput, parmInput, ...)
```

Arguments

doseInput	Dose-response dataframe containing "d1" and "d2" columns
parmInput	Numeric vector or list with appropriately named parameter inputs. Typically, it will be coefficients from a MarginalFit object.
...	Further arguments that are currently unused

initialMarginal *Estimate initial values for fitting marginal dose-response curves*

Description

This is a wrapper function which, when a dose-response dataframe is provided, returns start value estimates for both compounds that could be supplied to fitMarginals function. This function is also used by fitMarginals if no initials values were supplied.

Usage

```
initialMarginal(data, transforms = NULL, ...)
```

Arguments

data	Dose-response dataframe. Marginal data will be extracted from it automatically.
transforms	Transformation functions. If non-null, transforms is a list containing 5 elements, namely biological and power transformations along with their inverse functions and compositeArgs which is a list with argument values shared across the 4 functions. See vignette for more information.
...	Further parameters that are currently not used

Details

Note that this function returns e1 and 2 which are log-transformed inflection points for respective compounds.

Value

Named vector with parameter estimates. Parameter names are consistent with parameter names in `fitMarginals`. `h1` and `h2` are Hill's slope coefficients for each of the compounds, `m1` and `m2` are their maximal response levels whereas `b` is the shared baseline. Lastly, `e1` and `e2` are log-transformed EC50 values.

Note

Returns starting value for e = log(EC50).

Examples

```
data <- subset(directAntivirals, experiment == 1)
## Data must contain d1, d2 and effect columns
transforms <- getTransformations(data)
initialMarginal(data, transforms)
```

`isobologram` *Isobologram of the response surface predicted by the null model*

Description

If transformation functions are used, then the isobologram response levels will be plotted on the transformed scale.

Usage

```
isobologram(x, grid.len = 100, logScale = TRUE, ...)
```

Arguments

`x`	Output of `fitSurface`
`grid.len`	Number of concentrations to plot for each compound in the contour plot. An evenly spaced grid of doses will be generated for each compound given its respective observed minimum and maximum doses. Note that `grid.len^2` computations will be needed later so this number should stay reasonably low.
`logScale`	If `logScale = TRUE`, then grid of doses is evenly spaced in the logarithmic scale.
`...`	Further parameters that are not used at this moment.

L4 *4-parameter logistic dose-response function*

Description

4-parameter logistic dose-response function

Usage

```
L4(dose, b, L, U, logEC50)
```

Arguments

dose	Dose level
b	Hill's coefficient (slope of the curve)
L	Baseline effect (at zero dose)
U	Asymptote effect (at infinite dose)
logEC50	Point of inflection (in logarithmic terms)

marginalNLS *Fit two 4-parameter log-logistic functions with non-linear least squares*

Description

This function does not automatically extract marginal data and requires model input obtained from constructFormula.

Usage

```
marginalNLS(data, transforms = NULL, start, model, nlsfn = nls, ...)
```

Arguments

data	Dose-response dataframe. Marginal data will be extracted from it automatically.
transforms	Transformation functions. If non-null, transforms is a list containing 5 elements, namely biological and power transformations along with their inverse functions and compositeArgs which is a list with argument values shared across the 4 functions. See vignette for more information.
start	Starting parameter values. If not specified, they will be obtained from initialMarginal.
model	List with model parameters. Typically, this is an output from constructFormula.
nlsfn	Non-linear least-squares optimizer function
...	Further arguments that are passed to the optimizer function, such as lower or upper (for the "nlslm" method), or control.

marginalOptim *Fit two 4-parameter log-logistic functions with common baseline*

Description

This function is an alternative to non-linear least squares and provides optimization framework with `optim` function. It is however noticeably slower than NLS methods and can be especially time consuming in large datasets, in particular if bootstrap statistics are calculated.

Usage

```
marginalOptim(data, transforms = NULL, start, model, ...)
```

Arguments

`data`	Dose-response dataframe. Marginal data will be extracted from it automatically.
`transforms`	Transformation functions. If non-null, `transforms` is a list containing 5 elements, namely biological and power transformations along with their inverse functions and `compositeArgs` which is a list with argument values shared across the 4 functions. See vignette for more information.
`start`	Starting parameter values. If not specified, they will be obtained from `initialMarginal`.
`model`	List with model parameters. Typically, this is an output from `constructFormula`.
`...`	Further parameters passed to `optim` function

Value

Variance-covariance matrix which is returned by `optim` is based on the fact that minimization of sum-of-squared residuals leads essentially to a maximum likelihood estimator and so variance-covariance matrix can be estimated using inverse Hessian evaluated at the optimal parameters. In some cases, so obtained variance-covariance matrix might not be positive-definite which probably means that estimates are unstable because of either a poor choice of initial values or poor properties of the data itself.

maxR *Compute maxR statistic for each off-axis dose combination*

Description

`maxR` computes maxR statistics for each off-axis dose combination given the data provided. It provides a summary with results indicating whether a given point is estimated to be synergetic or antagonistic. These can be based either on normal approximation or a fully bootstrapped distribution of the statistics.

Usage

```
maxR(
  data_off,
  fitResult,
  transforms = fitResult$transforms,
  null_model = c("loewe", "hsa", "bliss", "loewe2"),
  R,
  CP,
  reps,
  nested_bootstrap = FALSE,
  B.B = NULL,
  cutoff = 0.95,
  cl = NULL,
  B.CP = NULL,
  method = c("equal", "model", "unequal"),
  bootStraps,
  idUnique,
  n1,
  doseGridOff,
  transFun,
  invTransFun,
  ...
)
```

Arguments

data_off	data frame with off -axis information
fitResult	Monotherapy (on-axis) model fit, e.g. produced by fitMarginals. It has to be a "MarginalFit" object or a list containing df, sigma, coef, shared_asymptote and method elements for, respectively, marginal model degrees of freedom, residual standard deviation, named vector of coefficient estimates, logical value of whether shared asymptote is imposed and method for estimating marginal models during bootstrapping (see fitMarginals). If biological and power transformations were used in marginal model estimation, fitResult should contain transforms elements with these transformations. Alternatively, these can also be specified via transforms argument.
transforms	Transformation functions. If non-null, transforms is a list containing 5 elements, namely biological and power transformations along with their inverse functions and compositeArgs which is a list with argument values shared across the 4 functions. See vignette for more information.
null_model	Specified null model for the expected response surface. Currently, allowed options are "loewe" for generalized Loewe model, "hsa" for Highest Single Agent model, "bliss" for Bliss additivity, and "loewe2" for the alternative Loewe generalization.
R	Numeric vector containing mean deviation of predicted response surface from the observed one at each of the off-axis points. If missing, it will be calculated automatically from output of predictOffAxis function.

CP	Prediction covariance matrix. If not specified, it will be estimated by bootstrap using B.CP iterations.
reps	Numeric vector containing number of replicates for each off-axis dose combination. If missing, it will be calculated automatically from output of predictOffAxis function.
nested_bootstrap	When statistics are calculated, if nested_bootstrap = TRUE, CP matrix is recalculated at each bootstrap iteration of B.B using B.CP iterations. Using such nested bootstrap may however significantly increase computational time. If nested_bootstrap = FALSE, CP bootstrapped data reuses CP matrix calculated from the original data.
B.B	Number of iterations to use in bootstrapping null distribution for either meanR or maxR statistics.
cutoff	Cut-off to use in maxR procedure for declaring non-additivity (default is 0.95).
cl	If parallel computations are desired, cl should be a cluster object created by makeCluster. If parallel computing is active, progress reporting messages are not necessarily ordered as it should be expected.
B.CP	Number of bootstrap iterations to use for CP matrix estimation
method	What assumption should be used for the variance of on- and off-axis points. This argument can take one of the values from c("equal", "model", "unequal"). With the value "equal" as the default. "equal" assumes that both on- and off-axis points have the same variance, "unequal" estimates a different parameter for on- and off-axis points and "model" predicts variance based on the average effect of an off-axis point. If no transformations are used the "model" method is recommended. If transformations are used, only the "equal" method can be chosen.
bootStraps	precomputed bootstrap objects
idUnique	unique combinations of on-axis points, a character vector
n1	the number of off-axis points
doseGridOff	dose grid for off-axis points
transFun	the transformation and inverse transformation functions for the variance
invTransFun	the transformation and inverse transformation functions for the variance
...	Further arguments that will be later passed to generateData function during bootstrapping

Value

This function returns a maxR object with estimates for the maxR statistical test. maxR object is essentially a list with appropriately named elements.

In particular, maxR object contains "Ymean" element which is a summary table of maxR test results for each dose combination. This table contains mean deviation from the predicted surface, normalized deviation ("absR") as well as a statistical call whether this deviation is significant. Distributional information on which these calls are made can be retrieved from the attributes of the "Ymean" dataframe.

Also, maxR object contains "Call" element which indicates the general direction of the deviation of the observed surface from the null. This call is based on the strongest local deviation in the "Ymean" table. 4 values are available here: "Syn", "Ant", "None", "Undefined". If one compound acts as an agonist while another one is an antagonist, then a deviation from the null is classified as "Undefined". If both compounds act in the same direction, then a stronger than individual effect is classified as synergy while a weaker effect would be classified as antagonism.

---

meanR *Compute meanR statistic for the estimated model*

Description

meanR computes the meanR statistic for the provided model and returns the computed F-statistic and the estimated p-value. p-value can be calculated either by assuming an exact distribution or using bootstrapping procedure. In the latter case, null distribution of bootstrapped F-statistics is also returned.

Usage

```
meanR(
  data_off,
  fitResult,
  transforms = fitResult$transforms,
  null_model = c("loewe", "hsa", "bliss", "loewe2"),
  R,
  CP,
  reps,
  nested_bootstrap = FALSE,
  B.B = NULL,
  B.CP = NULL,
  cl = NULL,
  method = c("equal", "model", "unequal"),
  bootStraps,
  paramsBootstrap,
  idUnique,
  n1,
  transFun,
  invTransFun,
  ...
)
```

Arguments

data_off	data frame with off -axis information
fitResult	Monotherapy (on-axis) model fit, e.g. produced by fitMarginals. It has to be a "MarginalFit" object or a list containing df, sigma, coef, shared_asymptote and method elements for, respectively, marginal model degrees of freedom,

residual standard deviation, named vector of coefficient estimates, logical value of whether shared asymptote is imposed and method for estimating marginal models during bootstrapping (see `fitMarginals`). If biological and power transformations were used in marginal model estimation, `fitResult` should contain `transforms` elements with these transformations. Alternatively, these can also be specified via `transforms` argument.

`transforms` Transformation functions. If non-null, `transforms` is a list containing 5 elements, namely biological and power transformations along with their inverse functions and `compositeArgs` which is a list with argument values shared across the 4 functions. See vignette for more information.

`null_model` Specified null model for the expected response surface. Currently, allowed options are `"loewe"` for generalized Loewe model, `"hsa"` for Highest Single Agent model, `"bliss"` for Bliss additivity, and `"loewe2"` for the alternative Loewe generalization.

`R` Numeric vector containing mean deviation of predicted response surface from the observed one at each of the off-axis points. If missing, it will be calculated automatically from output of `predictOffAxis` function.

`CP` Matrix which is part of covariance matrix for the `R` argument

`reps` Numeric vector containing number of replicates for each off-axis dose combination. If missing, it will be calculated automatically from output of `predictOffAxis` function.

`nested_bootstrap`
When statistics are calculated, if `nested_bootstrap = TRUE`, `CP` matrix is recalculated at each bootstrap iteration of `B.B` using `B.CP` iterations. Using such nested bootstrap may however significantly increase computational time. If `nested_bootstrap = FALSE`, `CP` bootstrapped data reuses `CP` matrix calculated from the original data.

`B.B` Number of iterations to use in bootstrapping null distribution for either meanR or maxR statistics.

`B.CP` Number of bootstrap iterations to use for CP matrix estimation

`cl` If parallel computations are desired, `cl` should be a cluster object created by `makeCluster`. If parallel computing is active, progress reporting messages are not necessarily ordered as it should be expected.

`method` What assumption should be used for the variance of on- and off-axis points. This argument can take one of the values from `c("equal", "model", "unequal")`. With the value `"equal"` as the default. `"equal"` assumes that both on- and off-axis points have the same variance, `"unequal"` estimates a different parameter for on- and off-axis points and `"model"` predicts variance based on the average effect of an off-axis point. If no transformations are used the `"model"` method is recommended. If transformations are used, only the `"equal"` method can be chosen.

`bootStraps` precomputed bootstrap objects

`paramsBootstrap`
parameters for the nested bootstrap

`idUnique` unique combinations of on-axis points, a character vector

`n1` the number of off-axis points

`transFun, invTransFun`
the transformation and inverse transformation functions for the variance

`...` Further arguments that will be later passed to `generateData` function during bootstrapping

Value

This function returns a `meanR` object with estimates for the meanR statistical test. `meanR` object is essentially a list with appropriately named elements.

`meanR` object list includes notably the calculated F-statistic, p-value and degrees of freedom (`"n1"` and `"df0"` respectively) used to find the critical value of the F-distribution under the null.

If `meanR` test is run with bootstrapping, then p-value estimate is based on bootstrapped null distribution of test statistic and an additional element `"FDist"` (of class `"ecdf"`) is returned.

`modelVar` *Calculate model variance, assuming variance increases linearly with mean*

Description

Calculate model variance, assuming variance increases linearly with mean

Usage

```
modelVar(dat_off, transFun, invTransFun)
```

Arguments

`dat_off` off-axis points data

`transFun, invTransFun`
the transformation and inverse transformation functions for the variance

Value

the predicted model variance

optim.boxcox *Find optimal Box-Cox transformation parameters*

Description

Find optimal Box-Cox transformation parameters

Usage

```
optim.boxcox(value, fac, shift = FALSE)
```

Arguments

value	Response variable in the data, e.g. "effect" column
fac	Factor indicating groups of replicates, e.g. interaction(d1,d2)
shift	Whether to use 2-parameter Box-Cox transformation. Input may be TRUE/FALSE or a numeric value indicating the shift parameter to use. If FALSE, shift parameter is set to zero.

Value

Numeric vector with power and shift parameter in that order.

Examples

```
data <- subset(directAntivirals, experiment == 1)
optim.boxcox(data$effect, interaction(data$d1, data$d2))
```

outsidePoints *List non-additive points*

Description

List all points with corresponding p-values declared non-additive by the maxR statistical test.

Usage

```
outsidePoints(maxR, B = 10000)
```

Arguments

maxR	maxR statistics table returned by Ymean component from the output of maxR function. This can also be "maxR" element in the output of fitSurface function.
B	Iterations to use for the distribution of the maxR statistic. This is only used if Ymean dataframe does not have a "distr" attribute attached as is normally done when using fitSurface or maxR function.

Value

Returns a dataframe listing only dose combinations that exhibit significant deviations from the expected response surface.

Examples

```
data <- subset(directAntivirals, experiment == 2)
## Data must contain d1, d2 and effect columns
fitResult <- fitMarginals(data)
surf <- fitSurface(data, fitResult, statistic = "maxR")
outsidePoints(surf$maxR$Ymean)
```

---

plot.BIGLconfInt *Plot confidence intervals in a contour plot*

Description

Plot confidence intervals in a contour plot

Usage

```
## S3 method for class 'BIGLconfInt'
plot(x, color = "effect-size", showAll = TRUE, ...)
```

Arguments

x	off axis confidence intervals, a data frame
color	analysis with which to colour cells, either effect-size or maxR
showAll	show all intervals in the plot or only significant ones, logical defaulting to TRUE
...	additional arguments, currently ignored

Note

written after the contour() function in the drugCombo package

plot.MarginalFit *Plot monotherapy curve estimates*

Description

Plot monotherapy curve estimates

Usage

```
## S3 method for class 'MarginalFit'
plot(x, ncol = 2, logScale = TRUE, smooth = TRUE, dataScale = FALSE, ...)
```

Arguments

x	Output of fitMarginals function or a "MarginalFit" object
ncol	Number of plots per row
logScale	Whether x-axis should be plotted on a logarithmic scale
smooth	Whether to draw a smooth fitted curve (deafult), or line segments connecting predicted points only
dataScale	Whether to draw plot on original data scale in case when transformations were used for fitting. Default (FALSE) is to plot on the coef(x) scale
...	Further arguments

Value

Returns a ggplot object. It can be consequently modified by using standard operations on ggplot objects (if ggplot2 package is loaded).

plot.maxR *Plot of maxR object*

Description

Plot of maxR object

Usage

```
## S3 method for class 'maxR'
plot(
  x,
  main = "Contour plot for maxR",
  xlab = "Dose (Compound 1)",
  ylab = "Dose (Compound 2)",
  colorPalette = c("blue", "white", "red"),
```

```
  logScale = TRUE,
  zTransform = function(z) {     z },
  plevels = c(0.7, 0.8, 0.9, 0.95, 0.99, 0.999),
  cutoff = max(plevels),
  maxshow = NULL,
  ...
)
```

Arguments

x	Output of maxR. This can also be "maxR" element in the output of fitSurface.
main	Fixed non-moving title for the 3D plot
xlab	X axis label using font, size and color par(c("font.lab", "cex.lab", "col.lab")).
ylab	Y axis label, same font attributes as xlab.
colorPalette	Vector of color names for surface
logScale	Draw doses on log-scale (setting zeroes to be finite constant)
zTransform	Optional transformation function for z-axis. By default, identity function is used.
plevels	Probability levels used to generate a color scale
cutoff	Probability cutoff to use for range of colors
maxshow	Forced value for range of colors
...	Further arguments that are passed to format function for formatting of axis labels

---

plot.meanR *Plot bootstrapped cumulative distribution function of meanR null distribution*

Description

Plot bootstrapped cumulative distribution function of meanR null distribution

Usage

```
## S3 method for class 'meanR'
plot(x, ...)
```

Arguments

x	Output from meanR
...	Further arguments

plot.ResponseSurface *Method for plotting response surface objects*

Description

Method for plotting response surface objects

Usage

```
## S3 method for class 'ResponseSurface'
plot(x, color = c("z-score", "maxR", "occupancy", "confInt"), ...)
```

Arguments

x	Output of fitSurface
color	Character indicating on what values surface coloring will be based. If color = "z-score", surface coloring will be based on median of standartized off-axis Z-scores. Median function can be replaced by other function using an optional colorfun argument which will be passed to plotResponseSurface. Color breaks are determined here by standard deviation of off-axis Z-scores. For color = "maxR", coloring will be based on values of maxR statistic and the quantile of its distribution (bootstrapped or not). If color = "occupancy", coloring will be based on calculated occupancy rate for the respective dose combination.
...	Further parameters passed to plotResponseSurface. colorBy argument in this method is computed automatically and thus cannot be passed to plotResponseSurface.

plotConfInt *Plot confidence intervals from BIGL object in a contour plot*

Description

Plot confidence intervals from BIGL object in a contour plot

Usage

```
plotConfInt(BIGLobj, ...)
```

Arguments

BIGLobj	Output from fitSurface
...	passed on to plot.BIGLconfInt

plotMeanVarFit *Make a mean-variance plot*

Description

Make a mean-variance plot

Usage

```
plotMeanVarFit(
  data,
  trans = "identity",
  invtrans = switch(trans, identity = "identity", log = "exp"),
  main = paste(switch(trans, identity = "No", log = "log"), "transformation"),
  log = switch(trans, identity = "", log = "y", ""),
  ...
)
```

Arguments

data	a dataset or matrix with d1, d2 and effect column
trans, invtrans	the transformation function for the variance and its inverse, possibly as strings
main	the title of the plot
log	log-transform of the axes, as in plot()
...	passed on to plot()

Details

This is a crucial graphical check for deciding on the

Value

Plots the mean-variance trend

plotResponseSurface *Plot response surface*

Description

Plot the 3-dimensional response surface predicted by one of the null models. This plot allows for a visual comparison between the null model prediction and observed points. This function is mainly used as the workhorse of `plot.ResponseSurface` method.

Usage

```
plotResponseSurface(
  data,
  fitResult = NULL,
  transforms = fitResult$transforms,
  predSurface = NULL,
  null_model = c("loewe", "hsa", "bliss", "loewe2"),
  colorPalette = c("blue", "grey70", "red"),
  colorBy = "none",
  colorPoints = c("black", "sandybrown", "brown", "white"),
  breaks = c(-Inf, 0, Inf),
  radius = NULL,
  logScale = TRUE,
  colorfun = median,
  zTransform = function(x) x,
  add = FALSE,
  main = "",
  legend = TRUE,
  xat = "actual",
  yat = "actual",
  plotfun = NULL,
  ...
)
```

Arguments

`data`	Dose-response dataframe.
`fitResult`	Monotherapy (on-axis) model fit, e.g. produced by `fitMarginals`. It has to be a `"MarginalFit"` object or a list containing `df`, `sigma`, `coef`, `shared_asymptote` and `method` elements for, respectively, marginal model degrees of freedom, residual standard deviation, named vector of coefficient estimates, logical value of whether shared asymptote is imposed and method for estimating marginal models during bootstrapping (see `fitMarginals`). If biological and power transformations were used in marginal model estimation, `fitResult` should contain `transforms` elements with these transformations. Alternatively, these can also be specified via `transforms` argument.
`transforms`	Transformation functions. If non-null, `transforms` is a list containing 5 elements, namely biological and power transformations along with their inverse functions and `compositeArgs` which is a list with argument values shared across the 4 functions. See vignette for more information.
`predSurface`	Vector of all predicted responses based on `expand.grid(uniqueDoses)`. If not supplied, it will be computed with `predictOffAxis` function.
`null_model`	If `predSurface` is not supplied, it is computed using one of the available null models, i.e. `"loewe"`, `"hsa"`, `"bliss"` and `"loewe2"`. See also `fitSurface`.
`colorPalette`	Vector of color names for surface

colorBy	This parameter determines values on which coloring is based for the 3-dimensional surface. If matrix or a data frame with d1 and d2 columns is supplied, dose combinations from colorBy will be matched automatically to the appropriate dose combinations in data. Unmatched dose combinations will be set to 0. This is especially useful for plotting results for off-axis estimates only, e.g. off-axis Z-scores or maxR test statistics. If colorBy = "colors", surface will be colored using colors in colorPalette argument.
colorPoints	Colors for off-axis and on-axis points. Character vector of length four with colors for 1) off-axis points; 2) on-axis points of the first drug (i.e. second drug is dosed at zero); 3) on-axis points of the second drug; 4) on-axis points where both drugs are dosed at zero.
breaks	Numeric vector with numerical breaks. To be used in conjunction with colorPalette argument.
radius	Radius of spheres. If missing, an educated guess based on number of digits in average effect will be made.
logScale	Draw doses on log-scale (setting zeroes to be finite constant)
colorfun	If replicates in colorBy variable are present, these will be aggregated using colorfun function. This can also be a custom function returning a scalar.
zTransform	Optional transformation function for z-axis. By default, identity function is used.
add	Add the predicted response surface to an existing plot. Will not draw any points, just the surface. Must be called after another call to plotResponseSurface.
main	Fixed non-moving title for the 3D plot
legend	Whether legend should be added
xat	x-axis ticks: "pretty", "actual" or a numeric vector
yat	y-axis ticks: "pretty", "actual" or a numeric vector
plotfun	If replicates for dose combinations in data are available, points can be aggregated using plotfun function. Typically, it will be mean, median, min or max but a custom-defined function returning a scalar from a vector is also possible.
...	Further arguments to format axis labels

Details

Title for the plot and legend are drawn as bitmaps and do not rotate with the rest of the plot. Since they are bitmaps, they do not scale properly, hence resizing window will result in unappealing visuals. For them to look properly, it suffices to set the appropriate RGL window size and rerun the plotting command.

Value

Plot is shown on a rgl device.

Examples

```
## Not run:
  data <- subset(directAntivirals, experiment == 1)
  ## Data must contain d1, d2 and effect columns
  fitResult <- fitMarginals(data)
  data_mean <- aggregate(effect ~ d1 + d2, data = data[, c("d1", "d2", "effect")],
                         FUN = mean)

  ## Construct the surface from marginal fit estimates based on HSA
  ## model and color it by mean effect level
  plotResponseSurface(data, fitResult, null_model = "hsa",
                      colorBy = data_mean, breaks = 10^(c(0, 3, 4, 6)),
                      colorPalette = c("grey", "blue", "green"))

  ## Response surface based on Loewe additivity model and colored with
  ## rainbow colors. Legend will not be displayed in any case.
  plotResponseSurface(data, fitResult, null_model = "loewe",
                      colorBy = "colors", colorPalette = rainbow(6))

## End(Not run)
```

`predict.MarginalFit` *Predict values on the dose-response curve*

Description

Predict values on the dose-response curve

Usage

```
## S3 method for class 'MarginalFit'
predict(object, newdata, ...)
```

Arguments

`object`	Output of `fitMarginals` function
`newdata`	An optional data frame in which to look for `d1` and `d2` variables with which to predict. If omitted, the fitted values are used. Doses that are passed to this function must correspond to marginal data, i.e. at least one of the doses must be zero.
`...`	Further arguments

predictOffAxis *Compute off-axis predictions*

Description

Given a dataframe with dose-response data, this function uses coefficient estimates from the marginal (on-axis) monotherapy model to compute the expected values of response at off-axis dose combinations using a provided null model.

Usage

```
predictOffAxis(
  doseGrid,
  fitResult,
  transforms = fitResult$transforms,
  null_model = c("loewe", "hsa", "bliss", "loewe2"),
  fit = NULL,
  ...
)
```

Arguments

doseGrid	A dose grid with unique combination of doses
fitResult	Monotherapy (on-axis) model fit, e.g. produced by fitMarginals. It has to be a "MarginalFit" object or a list containing df, sigma, coef, shared_asymptote and method elements for, respectively, marginal model degrees of freedom, residual standard deviation, named vector of coefficient estimates, logical value of whether shared asymptote is imposed and method for estimating marginal models during bootstrapping (see fitMarginals). If biological and power transformations were used in marginal model estimation, fitResult should contain transforms elements with these transformations. Alternatively, these can also be specified via transforms argument.
transforms	Transformation functions. If non-null, transforms is a list containing 5 elements, namely biological and power transformations along with their inverse functions and compositeArgs which is a list with argument values shared across the 4 functions. See vignette for more information.
null_model	Specified null model for the expected response surface. Currently, allowed options are "loewe" for generalized Loewe model, "hsa" for Highest Single Agent model, "bliss" for Bliss additivity, and "loewe2" for the alternative Loewe generalization.
fit	a pre-calculated off-axis fit
...	Further arguments passed on to the Loewe fitters

Value

This functions returns a named vector with predicted off-axis points

Examples

```
data <- subset(directAntivirals, experiment == 1)
## Data must contain d1, d2 and effect columns
transforms <- getTransformations(data)
fitResult <- fitMarginals(data, transforms)
  uniqueDoses <- with(data, list("d1" = sort(unique(data$d1)),
  "d2" = sort(unique(data$d2))))
  doseGrid <- expand.grid(uniqueDoses)
predictOffAxis(fitResult, null_model = "hsa", doseGrid = doseGrid)
```

`predictResponseSurface` *Predict the entire response surface, so including on-axis points, and return the result as a matrix. For plotting purposes.*

Description

Predict the entire response surface, so including on-axis points, and return the result as a matrix. For plotting purposes.

Usage

```
predictResponseSurface(
  doseGrid,
  fitResult,
  null_model,
  transforms = fitResult$transforms
)
```

Arguments

`doseGrid`	A dose grid with unique combination of doses
`fitResult`	Monotherapy (on-axis) model fit, e.g. produced by `fitMarginals`. It has to be a `"MarginalFit"` object or a list containing `df`, `sigma`, `coef`, `shared_asymptote` and `method` elements for, respectively, marginal model degrees of freedom, residual standard deviation, named vector of coefficient estimates, logical value of whether shared asymptote is imposed and method for estimating marginal models during bootstrapping (see `fitMarginals`). If biological and power transformations were used in marginal model estimation, `fitResult` should contain `transforms` elements with these transformations. Alternatively, these can also be specified via `transforms` argument.
`null_model`	Specified null model for the expected response surface. Currently, allowed options are `"loewe"` for generalized Loewe model, `"hsa"` for Highest Single Agent model, `"bliss"` for Bliss additivity, and `"loewe2"` for the alternative Loewe generalization.

transforms	Transformation functions. If non-null, transforms is a list containing 5 elements, namely biological and power transformations along with their inverse functions and compositeArgs which is a list with argument values shared across the 4 functions. See vignette for more information.

---

predictVar *Predict variance*

Description

Predict variance

Usage

```
predictVar(means, model, invTransFun)
```

Arguments

means	a vector of means
model	The mean-variance model
invTransFun	the inverse transformation function, back to the variance domain

---

print.summary.BIGLconfInt
Print summary of BIGLconfInt object

Description

Print summary of BIGLconfInt object

Usage

```
## S3 method for class 'summary.BIGLconfInt'
print(x, ...)
```

Arguments

x	Summary of BIGLconfInt object
...	Further arguments

print.summary.MarginalFit *Print method for summary of* MarginalFit *object*

Description

Print method for summary of MarginalFit object

Usage

```
## S3 method for class 'summary.MarginalFit'
print(x, ...)
```

Arguments

x	Summary of MarginalFit object
...	Further arguments

print.summary.maxR *Print summary of maxR object*

Description

Print summary of maxR object

Usage

```
## S3 method for class 'summary.maxR'
print(x, ...)
```

Arguments

x	Summary of "maxR" object
...	Further arguments

print.summary.meanR *Print summary of meanR object*

Description

Print summary of meanR object

Usage

```
## S3 method for class 'summary.meanR'
print(x, ...)
```

Arguments

x	Summary of meanR object
...	Further arguments

print.summary.ResponseSurface *Print method for the summary function of* ResponseSurface *object*

Description

Print method for the summary function of ResponseSurface object

Usage

```
## S3 method for class 'summary.ResponseSurface'
print(x, ...)
```

Arguments

x	Summary of ResponseSurface object
...	Further parameters

residuals.MarginalFit *Residuals from marginal model estimation*

Description

Residuals from marginal model estimation

Usage

```
## S3 method for class 'MarginalFit'
residuals(object, ...)
```

Arguments

object	Output of fitMarginals function
...	Further arguments

runBIGL *Run the BIGL application for demonstrating response surfaces*

Description

Run the BIGL application for demonstrating response surfaces

Usage

```
runBIGL(...)
```

Arguments

...	Pass parameters to runApp

Examples

```
## Not run:
  runBIGL()

## End(Not run)
```

sampleResids *Sample residuals according to a new model*

Description

Sample residuals according to a new model

Usage

```
sampleResids(means, sampling_errors, method, rescaleResids, ...)
```

Arguments

means	a vector of means
sampling_errors	Sampling vector to resample errors from. Used only if error is 4 and is passed as argument to generateData. If sampling_errors = NULL (default), mean residuals at off-axis points between observed and predicted response are taken.
method	What assumption should be used for the variance of on- and off-axis points. This argument can take one of the values from c("equal", "model", "unequal"). With the value "equal" as the default. "equal" assumes that both on- and off-axis points have the same variance, "unequal" estimates a different parameter for on- and off-axis points and "model" predicts variance based on the average effect of an off-axis point. If no transformations are used the "model" method is recommended. If transformations are used, only the "equal" method can be chosen.
rescaleResids	a boolean indicating whether to rescale residuals, or else normality of the residuals is assumed.
...	passed on to predictVar

Value

sampled residuals

scaleResids *Functions for scaling, and rescaling residuals. May lead to unstable behaviour in practice*

Description

Functions for scaling, and rescaling residuals. May lead to unstable behaviour in practice

Usage

```
scaleResids(sampling_errors, ...)
```

Arguments

sampling_errors	A vector of raw residuals
...	passed on to predictVar

Details

Residuals are calculated with respect to the average observation on the off-axis point, so replicates are required!

---

simulateNull *Simulate data from a given null model and monotherapy coefficients*

Description

Simulate data from a given null model and monotherapy coefficients

Usage

```
simulateNull(
  data,
  fitResult,
  doseGrid,
  transforms = fitResult$transforms,
  startvalues,
  null_model = c("loewe", "hsa", "bliss", "loewe2"),
  ...
)
```

Arguments

data	Dose-response dataframe.
fitResult	Monotherapy (on-axis) model fit, e.g. produced by fitMarginals. It has to be a "MarginalFit" object or a list containing df, sigma, coef, shared_asymptote and method elements for, respectively, marginal model degrees of freedom, residual standard deviation, named vector of coefficient estimates, logical value of whether shared asymptote is imposed and method for estimating marginal models during bootstrapping (see fitMarginals). If biological and power transformations were used in marginal model estimation, fitResult should contain transforms elements with these transformations. Alternatively, these can also be specified via transforms argument.
doseGrid	A grid of dose combinations
transforms	Transformation functions. If non-null, transforms is a list containing 5 elements, namely biological and power transformations along with their inverse functions and compositeArgs which is a list with argument values shared across the 4 functions. See vignette for more information.

startvalues	Starting values for the non-linear equation, from the observed data
null_model	Specified null model for the expected response surface. Currently, allowed options are "loewe" for generalized Loewe model, "hsa" for Highest Single Agent model, "bliss" for Bliss additivity, and "loewe2" for the alternative Loewe generalization.
...	Further parameters that will be passed to generateData

Value

List with data element containing simulated data and fitResult element containing marginal fit on the simulated data.

Examples

```
data <- subset(directAntivirals, experiment == 1)
## Data must contain d1, d2 and effect columns
fitResult <- fitMarginals(data)
simDat <- simulateNull(data, fitResult, expand.grid(d1 = data$d1, d2 = data$d2),
null_model = "hsa")
```

---

summary.BIGLconfInt *Summary of confidence intervals object*

Description

Summary of confidence intervals object

Usage

```
## S3 method for class 'BIGLconfInt'
summary(object, ...)
```

Arguments

object	Output from bootConfInt
...	Further arguments

summary.MarginalFit *Summary of* MarginalFit *object*

Description

Summary of MarginalFit object

Usage

```
## S3 method for class 'MarginalFit'
summary(object, ...)
```

Arguments

object	Output of fitMarginals function
...	Further arguments

summary.maxR *Summary of maxR object*

Description

Summary of maxR object

Usage

```
## S3 method for class 'maxR'
summary(object, ...)
```

Arguments

object	Object of "maxR" class
...	Further arguments

summary.meanR *Summary of meanR object*

Description

Summary of meanR object

Usage

```
## S3 method for class 'meanR'
summary(object, ...)
```

Arguments

object	Output from meanR
...	Further arguments

summary.ResponseSurface *Summary of* ResponseSurface *object*

Description

Summary of ResponseSurface object

Usage

```
## S3 method for class 'ResponseSurface'
summary(object, ...)
```

Arguments

object	Output of fitSurface
...	Further parameters

`vcov.MarginalFit` *Estimate of coefficient variance-covariance matrix*

Description

Estimate of coefficient variance-covariance matrix

Usage

```
## S3 method for class 'MarginalFit'
vcov(object, ...)
```

Arguments

`object`	Output of `fitMarginals` function
`...`	Further arguments

Index