Package 'DoseFinding'

May 7, 2026

Type Package

Title Planning and Analyzing Dose Finding Experiments

Version 1.4-1

Date 2025-07-08

Imports ggplot2, lattice, mvtnorm

Suggests emmeans, numDeriv, Rsolnp, quadprog, parallel, mmrm, multcomp, knitr, rmarkdown, MASS, testthat (>= 3.0.0), tibble, RBesT, nlme, dplyr, tidyr

Maintainer Marius Thomas <marius.thomas@novartis.com>

Description The DoseFinding package provides functions for the design and analysis of dose-finding experiments (with focus on pharmaceutical Phase II clinical trials). It provides functions for: multiple contrast tests, fitting non-linear dose-response models (using Bayesian and non-Bayesian estimation), calculating optimal designs and an implementation of the MCPMod methodology (Pinheiro et al. (2014) <doi:10.1002/sim.6052>).

VignetteBuilder knitr

License GPL-3

LazyLoad yes

RoxygenNote 7.3.2

Encoding UTF-8

URL https://github.com/openpharma/DoseFinding, https://openpharma.github.io/DoseFinding/

BugReports https://github.com/openpharma/DoseFinding/issues

Config/testthat/edition 3

NeedsCompilation yes

Author Bjoern Bornkamp [aut] (ORCID: <https://orcid.org/0000-0002-6294-8185>), Jose Pinheiro [aut], Frank Bretz [aut], Ludger Sandig [aut],

Marius Thomas [aut, cre],
Daniel Sabanes Bove [aut] (ORCID:
<https://orcid.org/0000-0002-0176-9239>),
Novartis Pharma AG [cph, fnd]

Repository CRAN

Date/Publication 2025-07-09 18:00:06 UTC

Contents

`bFitMod` *Fit a dose-response model using Bayesian or bootstrap methods.*

Description

For 'type = "Bayes"', MCMC sampling from the posterior distribution of the dose-response model is done. The function assumes a multivariate normal distribution for resp with covariance matrix S, and this is taken as likelihood function and combined with the prior distributions specified in prior to form the posterior distribution.

For 'type = "bootstrap"', a multivariate normal distribution for resp with covariance matrix S is assumed, and a large number of samples is drawn from this distribution. For each draw the fitMod function with 'type = "general"' is used to fit the draws from the multivariate normal distribution.

Usage

```
bFitMod(dose, resp, model, S, placAdj = FALSE,
        type = c("Bayes", "bootstrap"),
        start = NULL, prior = NULL, nSim = 1000,
        MCMCcontrol = list(), control = NULL, bnds,
        addArgs = NULL)

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

## S3 method for class 'bFitMod'
predict(object, predType = c("full-model", "effect-curve"),
     summaryFct = function(x) quantile(x, probs = c(0.025, 0.25, 0.5, 0.75, 0.975)),
        doseSeq = NULL, lenSeq = 101, ...)

## S3 method for class 'bFitMod'
plot(x, plotType = c("dr-curve", "effect-curve"),
        quant = c(0.025, 0.5, 0.975),
        plotData = c("means", "meansCI", "none"),
        level = 0.95, lenDose = 201, ...)
```

Arguments

dose	Numeric specifying the dose variable.
resp	Numeric specifying the response estimate corresponding to the doses in dose
S	Covariance matrix associated with the dose-response estimate specified via resp
model	Dose-response model to fit, possible models are "linlog", "linear", "quadratic", "emax", "exponential", "sigEmax", "betaMod" and "logistic", see drmodels.
placAdj	Whether or not estimates in "placAdj" are placebo-adjusted (note that the linear in log and the logistic model cannot be fitted for placebo-adjusted data)
type	Character with allowed values "Bayes" and "bootstrap", Determining whether samples are drawn from the posterior, or the bootstrap distribution.
start	Optional starting values for the dose-response parameters in the MCMC algorithm.

`prior` List containing the information regarding the prior distributions for '`type = "Bayes"`'. The list needs to have as many entries as there are model parameters. The ordering of the list entries should be the same as in the arguments list of the model see (see `drmodels`). For example for the Emax model the first entry determines the prior for e0, the second to eMax and the third to ed50.

For each list entry the user has the choice to choose from 4 possible distributions:

- `norm`: Vector of length 2 giving mean and standard deviation.
- `t`: Vector of length 3 giving median, scale and degrees of freedom of the t-distribution.
- `lnorm`: Vector of length 2 giving mean and standard deviation on log scale.
- `beta`: Vector of length 4 giving lower and upper bound of the beta prior as well as the alpha and beta parameters of the beta distribution

`nSim` Desired number of samples to produce with the algorithm

`MCMCcontrol` List of control parameters for the MCMC algorithm

- `thin` Thinning rate. Must be a positive integer.
- `w` Numeric of same length as number of parameters in the model, specifies the width parameters of the slice sampler.
- `adapt` Logical whether to adapt the `w` (width) parameter of the slice sampler in a short trial run. The widths are chosen as IQR/1.3 of the trial run.

`control` Same as the control argument in `fitMod`.

`bnds` Bounds for non-linear parameters, in case '`type = "bootstrap"`'. If missing the the default bounds from `defBnds` is used.

`addArgs` List containing two entries named "scal" and "off" for the "betaMod" and "linlog" model. When addArgs is NULL the following defaults are used '`list(scal = 1.2*max(doses), off = 0.01*max(doses))`'

`x, object` A bFitMod object

`predType, summaryFct, doseSeq, lenSeq`

Arguments for the predict method.

'`predType`': predType determines whether predictions are returned for the dose-response curve or the effect curve (difference to placebo).

'`summaryFct`': If equal to NULL predictions are calculated for each sampled parameter value. Otherwise a summary function is applied to the dose-response predictions for each parameter value. The default is to calculate 0.025, 0.25, 0.5, 0.75, 0.975 quantiles of the predictions for each dose.

'`doseSeq`': Where to calculate predictions. If not specified predictions are calculated on a grid of length '`lenSeq`' between minimum and maximum dose.

'`lenSeq`': Length of the default grid where to calculate predictions.

`plotType, quant, plotData, level, lenDose`

Arguments for plot method.

'`plotType`': Determining whether the dose-response curve or the effect curve should be plotted.

'`quant`': Vector of quantiles to display in plot

'`plotData`': Determines how the original data are plotted: Either as means or as means with CI or not. The level of the CI is determined by the argument '`level`'.

'level': Level for CI, when plotData is equal to 'meansCI'.
'lenDose': Number of grid values to use for display.

... Additional arguments are ignored.

Details

Componentwise univariate slice samplers are implemented (see Neal, 2003) to sample from the posterior distribution.

Value

An object of class bFitMod, which is a list containing the matrix of posterior simulations plus some additional information on the fitted model.

Author(s)

Bjoern Bornkamp

References

Neal, R. M. (2003), Slice sampling, Annals of Statistics, 31, 705-767

See Also

fitMod

Examples

```
data(biom)
## produce first stage fit (using dose as factor)
anMod <- lm(resp~factor(dose)-1, data=biom)
drFit <- coef(anMod)
S <- vcov(anMod)
dose <- sort(unique(biom$dose))
## define prior list
## normal prior for E0 (mean=0 and sdev=10)
## normal prior for Emax (mean=0 and sdev=100)
## beta prior for ED50: bounds: [0,1.5] parameters shape1=0.45, shape2=1.7
prior <- list(norm = c(0, 10), norm = c(0,100), beta=c(0,1.5,0.45,1.7))
## now fit an emax model
gsample <- bFitMod(dose, drFit, S, model = "emax",
                   start = c(0, 1, 0.1), nSim = 1000, prior = prior)
## summary information
gsample
## samples are stored in
head(gsample$samples)
## predict 0.025, 0.25, 0.5, 0.75, 0.975 Quantile at 0, 0.5 and 1
predict(gsample, doseSeq = c(0, 0.5, 1))
## simple plot function
plot(gsample)
```

```
## now look at bootstrap distribution
gsample <- bFitMod(dose, drFit, S, model = "emax", type = "bootstrap",
                   nSim = 100, bnds = defBnds(1)$emax)
plot(gsample)

## now fit linear interpolation
prior <- list(norm = c(0,1000), norm = c(0,1000),
              norm = c(0,1000), norm = c(0,1000), norm = c(0,100))
gsample <- bFitMod(dose, drFit, S, model = "linInt",
                   start = rep(1,5), nSim = 1000, prior = prior)
gsample <- bFitMod(dose, drFit, S, model = "linInt", type = "bootstrap",
                   nSim = 100)

## data fitted on placebo adjusted scale
data(IBScovars)
anovaMod <- lm(resp~factor(dose)+gender, data=IBScovars)
drFit <- coef(anovaMod)[2:5] # placebo adjusted estimates at doses
vCov <- vcov(anovaMod)[2:5,2:5]
dose <- sort(unique(IBScovars$dose))[-1]
prior <- list(norm = c(0,100), beta=c(0,6,0.45,1.7))
## Bayes fit
gsample <- bFitMod(dose, drFit, vCov, model = "emax", placAdj=TRUE,
                   start = c(1, 0.1), nSim = 1000, prior = prior)
## bootstrap fit
gsample <- bFitMod(dose, drFit, vCov, model = "emax", placAdj=TRUE,
                   type = "bootstrap", start = c(1, 0.1),
                   nSim = 100, prior = prior, bnds = c(0.01,6))
## calculate target dose estimate
TD(gsample, Delta = 0.2)
## now fit linear interpolation
prior <- list(norm = c(0,1000), norm = c(0,1000), norm = c(0,1000), norm = c(0,100))
gsample <- bFitMod(dose, drFit, vCov, model = "linInt", placAdj=TRUE,
                   start = rep(1,4), nSim = 1000, prior = prior)
gsample <- bFitMod(dose, drFit, vCov, model = "linInt", type = "bootstrap",
                   placAdj = TRUE, nSim = 100)
```

`biom` *Biometrics Dose Response data*

Description

An example data set for dose response studies. This data set was used in Bretz et al. (2005) to illustrate the MCPMod methodology.

Usage

```
data(biom)
```

Format

A data frame with 100 observations on the following 2 variables.

`resp` a numeric vector containing the response values

`dose` a numeric vector containing the dose values

Source

Bretz, F., Pinheiro, J. C., and Branson, M. (2005), Combining multiple comparisons and modeling techniques in dose-response studies, *Biometrics*, **61**, 738–748

---

bMCTtest *Performs Bayesian multiple contrast test*

Description

This function performs a Bayesian multiple contrast test using normal mixture priors for the response on each dose, as proposed in Fleischer et al. (2022). For a general description of the multiple contrast test see `MCTtest()`.

Usage

```
bMCTtest(
  dose,
  resp,
  data = NULL,
  models,
  S = NULL,
  type = c("normal", "general"),
  prior,
  alpha = 0.025,
  na.action = na.fail,
  mvtcontrol = mvtnorm.control(),
  contMat = NULL,
  critV = NULL
)
```

Arguments

`dose`, `resp`	Either vectors of equal length specifying dose and response values, or names of variables in the data frame specified in '`data`'.
`data`	Data frame containing the variables referenced in dose and resp if '`data`' is not specified it is assumed that '`dose`' and '`resp`' are variables referenced from data (and no vectors)
`models`	An object of class '`Mods`', see `Mods()` for details
`S`	The covariance matrix of '`resp`' when '`type = "general"`', see Description.

type	Determines whether inference is based on an ANCOVA model under a homoscedastic normality assumption (when 'type = "normal"'), or estimates at the doses and their covariance matrix and degrees of freedom are specified directly in 'resp', 'S' and 'df'. See also fitMod() and Pinheiro et al. (2014).
prior	List of length equal to the number of doses with the prior for each arm. Each element needs to be of class 'normMix' (See 'RBesT' package documentation). It is assumed that the i-th component of the prior list corresponds to the i-th largest dose. For example the first entry in the list is the prior for the placebo group, the second entry the prior for the second lowest dose and so on. Internally the priors across the different arms are combined (densities multiplied) assuming independence. The resulting multivariate normal mixture prior will have as many components as the product of the number of components of the individual mixture priors. The posterior mixture is part of the result object under "posterior".
alpha	Significance level for the frequentist multiple contrast test. If no critical values are supplied via 'critV' this is used to derive critical values for Bayesian decision rule.
na.action	A function which indicates what should happen when the data contain NAs.
mvtcontrol	A list specifying additional control parameters for the 'qmvt' and 'pmvt' calls in the code, see also mvtnorm.control() for details.
contMat	Contrast matrix to apply to the posterior dose-response estimates. The contrasts need to be in the columns of the matrix (i.e. the column sums need to be 0). If not specified optimal contrasts are calculated using optContr().
critV	Supply a critical value for the maximum posterior probability of the contrasts being greater than zero that needs to be surpassed to establish a non-flat dose-response. If this argument is NULL, this will be derived from critical values for frequentist MCP-Mod using the provided 'alpha'.

Details

If 'type = "normal"', an ANCOVA model based on a homoscedastic normality assumption is fitted and posteriors for dose-response and contrast vectors are obtained assuming a known variance.

For 'type = "general"' it is assumed multivariate normally distributed estimates are specified in 'resp' with covariance given by 'S', which define the likelihood. Posteriors for dose-response and contrast vectors are then obtained assuming a known covariance matrix S

The multiple contrast test decision is based on the maximum posterior probability of a contrast being greater than zero. Thresholds for the posterior probability can either be supplied or will be derived from frequentist critical values. In the latter case the Bayesian test will give approximately the same results as the frequentist multiple contrast test if uninformative priors are used.

For the default calculation of optimal contrasts the prior information is ignored (i.e. contrasts are calculated in the same way as in MCTtest()). Fleischer et al. (2022) discuss using contrasts that take the prior effective sample sizes into account, which can be slightly more favourable for the Bayesian MCT test. Such alternative contrasts can be directly handed over via the 'contMat' argument.

For analysis with covariate adjustment, covariate-adjusted 'resp' and 'S' can be supplied together with using 'type = "general"'. See vignette("binary_data") vignette "Design and analysis template MCP-Mod for binary data" for an example on how to obtain covariate adjusted estimates.

Value

An object of class bMCTtest, a list containing the output.

Author(s)

Marius Thomas

References

Fleischer, F., Bossert, S., Deng, Q., Loley, C. and Gierse, J. (2022). Bayesian MCP-Mod, *Pharmaceutical Statistics*, **21**, 654–670

See Also

MCTtest(), optContr()

Examples

```
if (require("RBesT")) {

################################
## Normal outcome
################################

data(biom)
## define shapes for which to calculate optimal contrasts
doses <- c(0, 0.05, 0.2, 0.6, 1)
modlist <- Mods(emax = 0.05, linear = NULL, logistic = c(0.5, 0.1),
                linInt = c(0, 1, 1, 1), doses = doses)
## specify an informative prior for placebo, weakly informative for other arms
plc_prior <- mixnorm(inf = c(0.8, 0.4, 0.1), rob = c(0.2, 0.4, 10))
vague_prior <- mixnorm(c(1, 0, 10))
## i-th component of the prior list corresponds to the i-th largest dose
## (e.g. 1st component -> placebo prior; last component prior for top dose)
prior <- list(plc_prior, vague_prior, vague_prior, vague_prior, vague_prior)

m1 <- bMCTtest(dose, resp, biom, models=modlist, prior = prior)
## now supply a critical value (= threshold for maxmimum posterior probability)
m2 <- bMCTtest(dose, resp, biom, models=modlist, prior = prior, critV = 0.99)

###################################
## Binary outcome with covariates
###################################
## Not run:
## generate data
logit <- function(p) log(p / (1 - p))
inv_logit <- function(y) 1 / (1 + exp(-y))
doses <- c(0, 0.5, 1.5, 2.5, 4)

## set seed and ensure reproducibility across R versions
set.seed(1, kind = "Mersenne-Twister", sample.kind = "Rejection", normal.kind = "Inversion")
```

```
group_size <- 100
dose_vector <- rep(doses, each = group_size)
N <- length(dose_vector)
## generate covariates
x1 <- rnorm(N, 0, 1)
x2 <- factor(sample(c("A", "B"), N, replace = TRUE, prob = c(0.6, 0.4)))
## assume approximately logit(10%) placebo and logit(35%) asymptotic response with ED50=0.5
prob <- inv_logit(emax(dose_vector, -2.2, 1.6, 0.5) + 0.3 * x1 + 0.3 * (x2 == "B"))
dat <- data.frame(y = rbinom(N, 1, prob),
                  dose = dose_vector, x1 = x1, x2 = x2)

## specify an informative prior for placebo (on logit scale), weakly informative for other arms
plc_prior <- mixnorm(inf = c(0.8, -2, 0.5), rob = c(0.2, -2, 10))
vague_prior <- mixnorm(c(1, 0, 10))
prior <- list(plc_prior, vague_prior, vague_prior, vague_prior, vague_prior)

## candidate models
mods <- Mods(emax = c(0.25, 1), sigEmax = rbind(c(1, 3), c(2.5, 4)), betaMod = c(1.1, 1.1),
             placEff = logit(0.1), maxEff = logit(0.35)-logit(0.1),
             doses = doses)

fit_cov <- glm(y~factor(dose) + 0 + x1 + x2, data = dat, family = binomial)

covariate_adjusted_estimates <- function(mu_hat, S_hat, formula_rhs,
                                         doses, other_covariates, n_sim) {
  ## predict every patient under *every* dose
 oc_rep <- as.data.frame(lapply(other_covariates, function(col) rep(col, times = length(doses))))
  d_rep <- rep(doses, each = nrow(other_covariates))
  pdat <- cbind(oc_rep, dose = d_rep)
  X <- model.matrix(formula_rhs, pdat)
  ## average on probability scale then backtransform to logit scale
  mu_star <- logit(tapply(inv_logit(X %*% mu_hat), pdat$dose, mean))
  ## estimate covariance matrix of mu_star
 pred <- replicate(n_sim, logit(tapply(inv_logit(X %*% drop(mvtnorm::rmvnorm(1, mu_hat, S_hat))),
                                       pdat$dose, mean)))
  return(list(mu_star = as.numeric(mu_star), S_star = cov(t(pred))))
}

ca <- covariate_adjusted_estimates(coef(fit_cov), vcov(fit_cov), ~factor(dose)+0+x1+x2,
                                   doses, dat[, c("x1", "x2")], 1000)
bMCTtest(doses, ca$mu_star, S = ca$S_star, type = "general", models = mods, prior = prior)

## End(Not run)
################################################
## example with contrasts handed over
################################################

data(biom)
## define shapes for which to calculate optimal contrasts
doses <- c(0, 0.05, 0.2, 0.6, 1)
modlist <- Mods(emax = 0.05, linear = NULL, sigEmax = c(0.5, 5),
                linInt = c(0, 1, 1, 1), doses = doses)
```

```
## specify an informative prior for placebo, weakly informative for other arms
plc_prior <- mixnorm(inf = c(0.8, 0.4, 0.1), rob = c(0.2, 0.4, 10), sigma = 0.7)
vague_prior <- mixnorm(c(1, 0, 10), sigma = 0.7)
prior <- list(plc_prior, vague_prior, vague_prior, vague_prior, vague_prior)

## use prior effective sample sizes to calculate optimal contrasts
prior_ess <- unlist(lapply(prior, ess))
n_grp <- as.numeric(table(biom$dose))
weights <- n_grp + prior_ess
cmat <- optContr(modlist, w = weights)

bMCTtest(dose, resp, biom, models=modlist, prior = prior, contMat = cmat)
}
```

`critVal` *Calculate critical value for multiple contrast test*

Description

Calculation of the critical value for a maximum contrast test. This is based on the equicoordinate quantile function of the multivariate normal or t distribution as implemented in the `qmvt` function from the mvtnorm package.

Usage

```
critVal(
  corMat,
  alpha = 0.025,
  df = NULL,
  alternative = c("one.sided", "two.sided"),
  control = mvtnorm.control()
)
```

Arguments

`corMat`	Correlation matrix of contrasts
`alpha`	Significance level for the multiple contrast test
`df`	Specify the degrees of freedom to use, if this argument is missing '`df = Inf`' is used (which corresponds to the multivariate normal distribution).
`alternative`	Character determining the alternative for the multiple contrast trend test.
`control`	A list specifying additional control parameters for the '`qmvt`' and '`pmvt`' calls in the code, see also `mvtnorm.control()` for details.

Author(s)

Bjoern Bornkamp

See Also

powMCT(), optContr(), MCTtest()

Examples

```
R <- matrix(c(1,0.5,0.5,1), nrow=2)
critVal(R, alpha = 0.05, df = 1)
critVal(R, alpha = 0.05, df = 20)
critVal(R, alpha = 0.05, df = Inf)
```

---

defBnds *Calculates default bounds for non-linear parameters in dose-response models*

Description

Calculates reasonable bounds for non-linear parameters for the built-in non-linear regression model based on the dose range under investigation.

For the logistic model the first row corresponds to the ED50 parameter and the second row to the delta parameter. For the sigmoid Emax model the first row corresponds to the ED50 parameter and the second row to the h parameter, while for the beta model first and second row correspond to the delta1 and delta2 parameters. See logistic, sigEmax and betaMod for details.

Usage

```
defBnds(mD, emax = c(0.001, 1.5)*mD,
        exponential = c(0.1, 2)*mD,
        logistic = matrix(c(0.001, 0.01, 1.5, 1/2)*mD, 2),
        sigEmax = matrix(c(0.001*mD, 0.5, 1.5*mD, 10), 2),
        betaMod = matrix(c(0.05,0.05,4,4), 2))
```

Arguments

mD	Maximum dose in the study.
emax, exponential, logistic, sigEmax, betaMod	values for the nonlinear parameters for these model-functions

Value

List containing bounds for the model parameters.

Author(s)

Bjoern Bornkamp

See Also

fitMod

Examples

```
defBnds(mD = 1)
defBnds(mD = 200)
```

DesignMCPModApp *Start externally hosted DesignMCPMod Shiny App*

Description

This function starts the externally hosted DesignMCPMod Shiny App in a browser window. The app was developed by Sophie Sun [aut, cre], Danyi Xiong [aut], Bjoern Bornkamp [ctb], Frank Bretz [ctb], Ardalan Mirshani [ctb]. This app performs power and sample size calculations for a multiple contrast test for normal, binary and negative binomial outcomes. The app uses the DoseFinding package as calculation backend and the R code underlying the calculations in the app can be extracted from the app.

Usage

```
DesignMCPModApp()
```

---

drmodels *Built-in dose-response models in DoseFinding*

Description

Dose-response model functions and gradients.

Below are the definitions of the model functions:

Emax model

$$f(d, \theta) = E_0 + E_{max} \frac{d}{ED_{50} + d}$$

Sigmoid Emax Model

$$f(d, \theta) = E_0 + E_{max} \frac{d^h}{ED_{50}^h + d^h}$$

Exponential Model

$$f(d, \theta) = E_0 + E_1(\exp(d/\delta) - 1)$$

Beta model

$$f(d,\theta) = E_0 + E_{max} B(\delta_1, \delta_2)(d/scal)^{\delta_1}(1 - d/scal)^{\delta_2}$$

here

$$B(\delta_1, \delta_2) = (\delta_1 + \delta_2)^{\delta_1+\delta_2}/(\delta_1^{\delta_1} \delta_2^{\delta_2})$$

and $scal$ is a fixed dose scaling parameter.

Linear Model

$$f(d,\theta) = E_0 + \delta d$$

Linear in log Model

$$f(d,\theta) = E_0 + \delta \log(d + off)$$

here off is a fixed offset parameter.

Logistic Model

$$f(d,\theta) = E_0 + E_{\max} / \{1 + \exp[(ED_{50} - d)/\delta]\}$$

Quadratic Model

$$f(d,\theta) = E_0 + \beta_1 d + \beta_2 d^2$$

The standardized model equation for the quadratic model is $d + \delta d^2$, with $\delta = \beta_2/\beta_1$.

Linear Interpolation model

The linInt model provides linear interpolation at the values defined by the nodes vector. In virtually all situations the nodes vector is equal to the doses used in the analysis. For example the `Mods()` and the `fitMod()` function automatically use the doses that are used in the context of the function call as nodes. The guesstimates specified in the `Mods()` function need to be the treatment effects at the active doses standardized to the interval [0,1] (see the examples in the `Mods()` function).

Usage

```
emax(dose, e0, eMax, ed50)
emaxGrad(dose, eMax, ed50, ...)

sigEmax(dose, e0, eMax, ed50, h)
sigEmaxGrad(dose, eMax, ed50, h, ...)

exponential(dose, e0, e1, delta)
exponentialGrad(dose, e1, delta, ...)

quadratic(dose, e0, b1, b2)
quadraticGrad(dose, ...)
```

```
betaMod(dose, e0, eMax, delta1, delta2, scal)
betaModGrad(dose, eMax, delta1, delta2, scal, ...)

linear(dose, e0, delta)
linearGrad(dose, ...)

linlog(dose, e0, delta, off = 1)
linlogGrad(dose, off, ...)

logistic(dose, e0, eMax, ed50, delta)
logisticGrad(dose, eMax, ed50, delta, ...)

linInt(dose, resp, nodes)
linIntGrad(dose, resp, nodes, ...)
```

Arguments

dose	Dose variable
e0	For most models placebo effect. For logistic model left-asymptote parameter, corresponding to a basal effect level (not the placebo effect)
eMax	Beta Model: Maximum effect within dose-range Emax, sigmoid Emax, logistic Model: Asymptotic maximum effect
ed50	Dose giving half of the asymptotic maximum effect
...	Just included for convenience in the gradient functions, so that for example `quadratic(dose, e0=0, b1=1, b2=3)` will not throw an error (although the gradient of the quadratic model is independent of e0, b1 and b2).
h	Hill parameter, determining the steepness of the model at the ED50
e1	Slope parameter for exponential model
delta	Exponential model: Parameter, controlling the convexity of the model. Linear and linlog model: Slope parameter Logistic model: Parameter controlling determining the steepness of the curve
b1	first parameter of quadratic model
b2	second parameter of quadratic model (controls, whether model is convex or concave)
delta1	delta1 parameter for beta model
delta2	delta2 parameter for beta model
scal	Scale parameter (treated as a fixed value, not estimated)
off	Offset value to avoid problems with dose=0 (treated as a fixed value, not estimated)
resp	Response values at the nodes for the linInt model
nodes	Interpolation nodes for the linear interpolation for the linInt model (treated as a fixed value, not estimated)

Details

The **Emax model** is used to represent monotone, concave dose-response shapes. To distinguish it from the more general sigmoid emax model it is sometimes also called hyperbolic emax model.

The **sigmoid Emax** model is an extension of the (hyperbolic) Emax model by introducing an additional parameter h, that determines the steepness of the curve at the ed50 value. The sigmoid Emax model describes monotonic, sigmoid dose-response relationships. In the toxicology literature this model is also called four-parameter log-logistic (4pLL) model.

The **quadratic** model is intended to capture a possible non-monotonic dose-response relationship.

The **exponential model** is intended to capture a possible sub-linear or a convex dose-response relationship.

The **beta model** is intended to capture non-monotone dose-response relationships and is more flexible than the quadratic model. The kernel of the beta model function consists of the kernel of the density function of a beta distribution on the interval [0,scal]. The parameter scal is not estimated but needs to be set to a value larger than the maximum dose. It can be set in most functions ('`fitMod`', '`Mods`') via the '`addArgs`' argument, when omitted a value of '`1.2*(maximum dose)`' is used as default, where the maximum dose is inferred from other input to the respective function.

The **linear in log-dose** model is intended to capture concave shapes. The parameter `off` is not estimated in the code but set to a pre-specified value. It can be set in most functions ('`fitMod`', '`Mods`') via the '`addArgs`' argument, when omitted a value of '`0.01*(maximum dose)`' is used as default, where the maximum dose is inferred from other input to the respective function.

The **logistic model** is intended to capture general monotone, sigmoid dose-response relationships. The logistic model and the sigmoid Emax model are closely related: The sigmoid Emax model is a logistic model in log(dose).

The **linInt model** provids linear interpolation of the means at the doses. This can be used as a "nonparametric" estimate of the dose-response curve, but is probably most interesting for specifying a "nonparametric" truth during planning and assess how well parametric models work under a nonparametric truth. For the function '`Mods`' and '`fitMod`' the interpolation '`nodes`' are selected equal to the dose-levels specified.

Value

Response value for model functions or matrix containing the gradient evaluations.

References

MacDougall, J. (2006). Analysis of dose-response studies - Emax model,*in* N. Ting (ed.), *Dose Finding in Drug Development*, Springer, New York, pp. 127–145

Pinheiro, J. C., Bretz, F. and Branson, M. (2006). Analysis of dose-response studies - modeling approaches, *in* N. Ting (ed.). *Dose Finding in Drug Development*, Springer, New York, pp. 146–171

See Also

`fitMod()`

Examples

```
## some quadratic example shapes
quadModList <- Mods(quadratic = c(-0.5, -0.75, -0.85, -1), doses = c(0,1))
plotMods(quadModList)

## some emax example shapes
emaxModList <- Mods(emax = c(0.02,0.1,0.5,1), doses = c(0,1))
plotMods(emaxModList)
## example for gradient
emaxGrad(dose = (0:4)/4, eMax = 1, ed50 = 0.5)

## some sigmoid emax example shapes
sigEmaxModList <- Mods(sigEmax = rbind(c(0.05,1), c(0.15,3), c(0.4,8),
                       c(0.7,8)), doses = c(0,1))
plotMods(sigEmaxModList)
sigEmaxGrad(dose = (0:4)/4, eMax = 1, ed50 = 0.5, h = 8)

## some exponential example shapes
expoModList <- Mods(exponential = c(0.1,0.25,0.5,2), doses=c(0,1))
plotMods(expoModList)
exponentialGrad(dose = (0:4)/4, e1 = 1, delta = 2)

## some beta model example shapes
betaModList <- Mods(betaMod = rbind(c(1,1), c(1.5,0.75), c(0.8,2.5),
                    c(0.4,0.9)), doses=c(0,1), addArgs=list(scal = 1.2))
plotMods(betaModList)
betaModGrad(dose = (0:4)/4, eMax = 1, delta1 = 1, delta2 = 1, scal = 5)

## some logistic model example shapes
logistModList <- Mods(logistic = rbind(c(0.5,0.05), c(0.5,0.15),
                      c(0.2,0.05), c(0.2,0.15)), doses=c(0,1))
plotMods(logistModList)
logisticGrad(dose = (0:4)/4, eMax = 1, ed50 = 0.5, delta = 0.05)

## some linInt shapes
genModList <- Mods(linInt = rbind(c(0.5,1,1),
                      c(0,1,1), c(0,0,1)), doses=c(0,0.5,1,1.5))
plotMods(genModList)
linIntGrad(dose = (0:4)/4, resp=c(0,0.5,1,1,1), nodes=(0:4)/4)
```

`fitMod` *Fit non-linear dose-response model*

Description

Fits a dose-response model. Built-in dose-response models are "linlog", "linear", "quadratic", "emax", "exponential", "sigEmax", "betaMod" and "logistic" (see `drmodels()`).

Usage

```
fitMod(
  dose,
  resp,
  data = NULL,
  model = NULL,
  S = NULL,
  type = c("normal", "general"),
  addCovars = ~1,
  placAdj = FALSE,
  bnds,
  df = NULL,
  start = NULL,
  na.action = na.fail,
  control = NULL,
  addArgs = NULL
)

## S3 method for class 'DRMod'
coef(object, sep = FALSE, ...)

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

## S3 method for class 'DRMod'
predict(
  object,
  predType = c("full-model", "ls-means", "effect-curve"),
  newdata = NULL,
  doseSeq = NULL,
  se.fit = FALSE,
  ...
)

## S3 method for class 'DRMod'
plot(
  x,
  CI = FALSE,
  level = 0.95,
  plotData = c("means", "meansCI", "raw", "none"),
  plotGrid = TRUE,
  colMn = 1,
  colFit = 1,
  ...
)

## S3 method for class 'DRMod'
logLik(object, ...)
```

```
## S3 method for class 'DRMod'
AIC(object, ..., k = 2)

## S3 method for class 'DRMod'
gAIC(object, ..., k = 2)
```

Arguments

dose, resp	Either vectors of equal length specifying dose and response values, or names of variables in the data frame specified in 'data'.
data	Data frame containing the variables referenced in dose and resp if 'data' is not specified it is assumed that 'dose' and 'resp' are variables referenced from data (and no vectors)
model	The dose-response model to be used for fitting the data. Built-in models are "linlog", "linear", "quadratic", "emax", "exponential", "sigEmax", "betaMod" and "logistic" (see drmodels).
S	The inverse weighting matrix used in case, when 'type = "general"', see Description. For later inference statements (vcov or predict methods) it is assumed this is the estimated covariance of the estimates in the first stage fit.
type	Determines whether inference is based on an ANCOVA model under a homoscedastic normality assumption (when 'type = "normal"'), or estimates at the doses and their covariance matrix and degrees of freedom are specified directly in 'resp', 'S' and 'df'. See also the Description above and Pinheiro et al. (2014).
addCovars	Formula specifying additional additive linear covariates (only for 'type = "normal"')
placAdj	Logical, if true, it is assumed that placebo-adjusted estimates are specified in 'resp' (only possible for 'type = "general"').
bnds	Bounds for non-linear parameters. If missing the the default bounds from defBnds() is used. When the dose-response model has only one non-linear parameter (for example Emax or exponential model), 'bnds' needs to be a vector containing upper and lower bound. For models with two non-linear parameters 'bnds' needs to be a matrix containing the bounds in the rows, see the Description section of defBnds() for details on the formatting of the bounds for the individual models.
df	Degrees of freedom to use in case of 'type = "general"'. If this argument is missing 'df = Inf' is used. For 'type = "normal"' this argument is ignored as the exact degrees of freedom can be deduced from the model.
start	Vector of starting values for the nonlinear parameters (ignored for linear models). When equal to NULL, a grid optimization is performed and the best value is used as starting value for the local optimizer.
na.action	A function which indicates what should happen when the data contain NAs.
control	A list with entries: "nlminbcontrol", "optimizetol" and "gridSize".

The entry nlminbcontrol needs to be a list and it is passed directly to control argument in the nlminb function, that is used internally for models with 2 nonlinear parameters.

The entry optimizetol is passed directly to the tol argument of the optimize function, which is used for models with 1 nonlinear parameters.

The entry gridSize needs to be a list with entries dim1 and dim2 giving the size of the grid for the gridsearch in 1d or 2d models.

`addArgs` List containing two entries named "scal" and "off" for the "betaMod" and "linlog" model. When addArgs is NULL the following defaults is used '`list(scal = 1.2*max(doses), off = 0.01*max(doses))`'.

`object`, `x` DRMod object

`sep` Logical determining whether all coefficients should be returned in one numeric or separated in a list.

`...` Additional arguments for plotting for the plot method. For all other cases additional arguments are ignored.

`predType`, `newdata`, `doseSeq`, `se.fit`

predType determines whether predictions are returned for the full model (including potential covariates), the ls-means (SAS type) or the effect curve (difference to placebo).

newdata gives the covariates to use in producing the predictions (for predType = "full-model"), if missing the covariates used for fitting are used.

doseSeq dose-sequence on where to produce predictions (for predType = "effect-curve" and predType = "ls-means"). If missing the doses used for fitting are used.

se.fit: logical determining, whether the standard error should be calculated.

`CI`, `level`, `plotData`, `plotGrid`, `colMn`, `colFit`

Arguments for plot method: '`CI`' determines whether confidence intervals should be plotted. '`level`' determines the level of the confidence intervals. '`plotData`' determines how the data are plotted: Either as means or as means with CI, raw data or none. In case of '`type = "normal"`' and covariates the ls-means are displayed, when '`type = "general"`' the option "raw" is not available. '`colMn`' and '`colFit`' determine the colors of fitted model and the raw means.

`k` Penalty to use for model-selection criterion (AIC uses 2, BIC uses log(n)).

Details

When '`type = "normal"`' ordinary least squares is used and additional additive covariates can be specified in '`addCovars`'. The underlying assumption is hence normally distributed data and homoscedastic variance.

For '`type = "general"`' a generalized least squares criterion is used

$$(f(dose, \theta) - resp)' S^{-1} (f(dose, \theta) - resp)$$

and an inverse weighting matrix is specified in '`S`', '`type = "general"`' is primarily of interest, when fitting a model to AN(C)OVA type estimates obtained in a first stage fit, then '`resp`' contains

the estimates and 'S' is the estimated covariance matrix for the estimates in 'resp'. Statistical inference (e.g. confidence intervals) rely on asymptotic normality of the first stage estimates, which makes this method of interest only for sufficiently large sample size for the first stage fit. A modified model-selection criterion can be applied to these model fits (see also Pinheiro et al. 2014 for details).

For details on the implemented numerical optimizer see the Details section below.

Details on numerical optimizer for model-fitting:
For linear models fitting is done using numerical linear algebra based on the QR decomposition. For nonlinear models numerical optimization is performed only in the nonlinear parameters in the model and optimizing over the linear parameters in each iteration (similar as the Golub-Pereyra implemented in `nls()`). For models with 1 nonlinear parameter the `optimize()` function is used for 2 nonlinear parameters the `nlminb()` function is used. The starting value is generated using a grid-search (with the grid size specified via '`control$gridSize`'), or can directly be handed over via '`start`'.

For details on the asymptotic approximation used for '`type = "normal"`', see Seber and Wild (2003, chapter 5). For details on the asymptotic approximation used for '`type = "general"`', and the gAIC, see Pinheiro et al. (2014).

Value

An object of class DRMod. Essentially a list containing information about the fitted model coefficients, the residual sum of squares (or generalized residual sum of squares),

Author(s)

Bjoern Bornkamp

References

Pinheiro, J. C., Bornkamp, B., Glimm, E. and Bretz, F. (2014) Model-based dose finding under model uncertainty using general parametric models, *Statistics in Medicine*, **33**, 1646–1661

Seber, G.A.F. and Wild, C.J. (2003). Nonlinear Regression, Wiley.

See Also

`defBnds()`, `drmodels()`

Examples

```
## Fit the emax model to the IBScovars data set
data(IBScovars)
fitemax <- fitMod(dose, resp, data=IBScovars, model="emax",
                  bnds = c(0.01, 4))

## methods for DRMod objects
summary(fitemax)
## extracting coefficients
coef(fitemax)
## (asymptotic) covariance matrix of estimates
vcov(fitemax)
```

```
## predicting
newdat <- data.frame(dose = c(0,0.5,1), gender=factor(1))
predict(fitemax, newdata=newdat, predType = "full-model", se.fit = TRUE)
## plotting
plot(fitemax, plotData = "meansCI", CI=TRUE)

## now include (additive) covariate gender
fitemax2 <- fitMod(dose, resp, data=IBScovars, model="emax",
                   addCovars = ~gender, bnds = c(0.01, 4))
vcov(fitemax2)
plot(fitemax2)
## fitted log-likelihood
logLik(fitemax2)
## extracting AIC (or BIC)
AIC(fitemax2)

## Illustrating the "general" approach for a binary regression
## produce first stage fit (using dose as factor)
data(migraine)
PFrate <- migraine$painfree/migraine$ntrt
doseVec <- migraine$dose
doseVecFac <- as.factor(migraine$dose)
## fit logistic regression with dose as factor
fitBin <- glm(PFrate~doseVecFac-1, family = binomial,
              weights = migraine$ntrt)
drEst <- coef(fitBin)
vCov <- vcov(fitBin)
## now fit an Emax model (on logit scale)
gfit <- fitMod(doseVec, drEst, S=vCov, model = "emax", bnds = c(0,100),
                type = "general")
## model fit on logit scale
plot(gfit, plotData = "meansCI", CI = TRUE)
## model on probability scale
logitPred <- predict(gfit, predType ="ls-means", doseSeq = 0:200,
                     se.fit=TRUE)
plot(0:200, 1/(1+exp(-logitPred$fit)), type = "l", ylim = c(0, 0.5),
     ylab = "Probability of being painfree", xlab = "Dose")
LB <- logitPred$fit-qnorm(0.975)*logitPred$se.fit
UB <- logitPred$fit+qnorm(0.975)*logitPred$se.fit
lines(0:200, 1/(1+exp(-LB)))
lines(0:200, 1/(1+exp(-UB)))


## now illustrate "general" approach for placebo-adjusted data (on
## IBScovars) note that the estimates are identical to fitemax2 above)
anovaMod <- lm(resp~factor(dose)+gender, data=IBScovars)
drFit <- coef(anovaMod)[2:5] # placebo adjusted estimates at doses
vCov <- vcov(anovaMod)[2:5,2:5]
dose <- sort(unique(IBScovars$dose))[-1]
## now fit an emax model to these estimates
gfit2 <- fitMod(dose, drFit, S=vCov, model = "emax", type = "general",
                placAdj = TRUE, bnds = c(0.01, 2))
## some outputs
```

```
summary(gfit2)
coef(gfit2)
vcov(gfit2)
predict(gfit2, se.fit = TRUE, doseSeq = c(1,2,3,4), predType = "effect-curve")
plot(gfit2, CI=TRUE, plotData = "meansCI")
gAIC(gfit2)
```

`glycobrom` *Glycopyrronium Bromide dose-response data*

Description

Data from a clinical study evaluating Efficacy and Safety of Four Doses of Glycopyrronium Bromide in Patients With Stable Chronic Obstructive Pulmonary Disease (COPD). This data set was obtained from clinicaltrials.gov (NCT00501852). The study design was a 4 period incomplete cross-over design. The primary endpoint is the trough forced expiratory volume in 1 second (FEV1) following 7 days of Treatment.

Usage

```
data(glycobrom)
```

Format

A data frame with 5 summary estimates (one per dose). Variables: A data frame with 5 summary estimates (one per dose). Variables:

`dose` a numeric vector containing the dose values

`fev1` a numeric vector containing the least square mean per dose

`sdev` a numeric vector containing the standard errors of the least square means per dose

`n` Number of participants analyzed per treatment group

Details

The data given here are summary estimates (least-square means) for each dose.

Source

http://clinicaltrials.gov/ct2/show/results/NCT00501852

Examples

```
## simulate a full data set with given means and sdv (here we ignore
 ## the original study was a cross-over design, and simulate a parallel
 ## group design)
 simData <- function(mn, sd, n, doses, fixed = TRUE){
   ## simulate data with means (mns) and standard deviations (sd), for
   ## fixed = TRUE, the data set will have observed means and standard
   ## deviations as given in mns and sd
   resp <- numeric(sum(n))
   uppind <- cumsum(n)
   lowind <- c(0,uppind)+1
   for(i in 1:length(n)){
     rv <- rnorm(n[i])
     if(fixed)
       rv <- scale(rv)
     resp[lowind[i]:uppind[i]] <- mn[i] + sd[i]*rv
   }
   data.frame(doses=rep(doses, n), resp=resp)
 }
 data(glycobrom)
 fullDat <- simData(glycobrom$fev1, glycobrom$sdev, glycobrom$n,
                    glycobrom$dose)
```

---

guesst *Calculate guesstimates based on prior knowledge*

Description

Calculates guesstimates for standardized model parameter(s) using the general approach described in Pinheiro et al. (2006).

Usage

```
guesst(
  d,
  p,
 model = c("emax", "exponential", "logistic", "quadratic", "betaMod", "sigEmax"),
  less = TRUE,
  local = FALSE,
  dMax,
  Maxd,
  scal
)
```

Arguments

d	Vector containing dose value(s).

p	Vector of expected percentages of the maximum effect achieved at d.
model	Character string. Should be one of "emax", "exponential", "quadratic", "betaMod", "sigEmax", "logistic".
less	Logical, only needed in case of quadratic model. Determines if d is smaller ('less=TRUE') or larger ('less=FALSE') than dopt (see Pinheiro et al. (2006) for details).
local	Logical indicating whether local or asymptotic version of guesstimate should be derived (defaults to 'FALSE'). Only needed for emax, logistic and sigEmax model. When 'local=TRUE' the maximum dose must be provided via 'Maxd'.
dMax	Dose at which maximum effect occurs, only needed for the beta model
Maxd	Maximum dose to be administered in the trial
scal	Scale parameter, only needed for the beta model

Details

Calculates guesstimates for the parameters θ_2 of the standardized model function based on the prior expected percentage of the maximum effect at certain dose levels. Note that this function should be used together with the `plot.Mods()` function to ensure that the guesstimates are reflecting the prior beliefs.

For the logistic and sigmoid emax models at least two pairs (d,p) need to be specified.

For the beta model the dose at which the maximum effect occurs (dMax) has to be specified in addition to the (d,p) pair.

For the exponential model the maximum dose administered (Maxd) needs to be specified in addition to the (d,p) pair.

For the quadratic model one (d,p) pair is needed. It is advisable to specify the location of the maximum within the dose range with this pair.

For the emax, sigmoid Emax and logistic model one can choose between a local and an asymptotic version. In the local version one explicitly forces the standardized model function to pass through the specified points (d,p). For the asymptotic version it assumed that the standardized model function is equal to 1 at the largest dose (this is the approach described in Pinheiro et al. (2006)). If the local version is used, convergence problems with the underlying nonlinear optimization can occur.

Value

Returns a numeric vector containing the guesstimates.

References

Bornkamp B., Pinheiro J. C., and Bretz, F. (2009). MCPMod: An R Package for the Design and Analysis of Dose-Finding Studies, *Journal of Statistical Software*, **29**(7), 1–23

Pinheiro, J. C., Bretz, F., and Branson, M. (2006). Analysis of dose-response studies - modeling approaches, *in* N. Ting (ed.), *Dose Finding in Drug Development*, Springer, New York, pp. 146–171

See Also

`emax()`, `logistic()`, `betaMod()`, `sigEmax()`, `quadratic()`, `exponential()`, `plot.Mods()`

Examples

```
## Emax model
## Expected percentage of maximum effect: 0.8 is associated with
## dose 0.3 (d,p)=(0.3, 0.8), dose range [0,1]
emx1 <- guesst(d=0.3, p=0.8, model="emax")
emax(0.3,0,1,emx1)
## local approach
emx2 <- guesst(d=0.3, p=0.8, model="emax", local = TRUE, Maxd = 1)
emax(0.3,0,1,emx2)/emax(1,0,1,emx2)
## plot models
models <- Mods(emax=c(emx1, emx2), doses=c(0,1))
plot(models)

## Logistic model
## Select two (d,p) pairs (0.2, 0.6) and (0.2, 0.95)
lgc1 <- guesst(d = c(0.2, 0.6), p = c(0.2, 0.95), "logistic")
logistic(c(0.2,0.6), 0, 1, lgc1[1], lgc1[2])
## local approach
lgc2 <- guesst(d = c(0.2, 0.6), p = c(0.2, 0.95), "logistic",
               local = TRUE, Maxd = 1)
r0 <- logistic(0, 0, 1, lgc2[1], lgc2[2])
r1 <- logistic(1, 0, 1, lgc2[1], lgc2[2])
(logistic(c(0.2,0.6), 0, 1, lgc2[1], lgc2[2])-r0)/(r1-r0)
## plot models
models <- Mods(logistic = rbind(lgc1, lgc2), doses=c(0,1))
plot(models)

## Beta Model
## Select one pair (d,p): (0.4,0.8)
## dose, where maximum occurs: 0.8
bta <- guesst(d=0.4, p=0.8, model="betaMod", dMax=0.8, scal=1.2, Maxd=1)
## plot
models <- Mods(betaMod = bta, doses=c(0,1), addArgs = list(scal = 1.2))
plot(models)

## Sigmoid Emax model
## Select two (d,p) pairs (0.2, 0.6) and (0.2, 0.95)
sgE1 <- guesst(d = c(0.2, 0.6), p = c(0.2, 0.95), "sigEmax")
sigEmax(c(0.2,0.6), 0, 1, sgE1[1], sgE1[2])
## local approach
sgE2 <- guesst(d = c(0.2, 0.6), p = c(0.2, 0.95), "sigEmax",
               local = TRUE, Maxd = 1)
sigEmax(c(0.2,0.6), 0, 1, sgE2[1], sgE2[2])/sigEmax(1, 0, 1, sgE2[1], sgE2[2])
models <- Mods(sigEmax = rbind(sgE1, sgE2), doses=c(0,1))
plot(models)

## Quadratic model
## For the quadratic model it is assumed that the maximum effect occurs at
## dose 0.7
quad <- guesst(d = 0.7, p = 1, "quadratic")
models <- Mods(quadratic = quad, doses=c(0,1))
plot(models)
```

```
## exponential model
## (d,p) = (0.8,0.5)
expo <- guesst(d = 0.8, p = 0.5, "exponential", Maxd=1)
models <- Mods(exponential = expo, doses=c(0,1))
plot(models)
```

IBScovars *Irritable Bowel Syndrome Dose Response data with covariates*

Description

A subset of the data used by (Biesheuvel and Hothorn, 2002). The data are part of a dose ranging trial on a compound for the treatment of the irritable bowel syndrome with four active treatment arms, corresponding to doses 1,2,3,4 and placebo. Note that the original dose levels have been blinded in this data set for confidentiality. The primary endpoint was a baseline adjusted abdominal pain score with larger values corresponding to a better treatment effect. In total 369 patients completed the study, with nearly balanced allocation across the doses.

Usage

```
data(IBScovars)
```

Format

A data frame with 369 observations on the following 2 variables.

`gender` a factor specifying the gender

`dose` a numeric vector

`resp` a numeric vector

Source

Biesheuvel, E. and Hothorn, L. A. (2002). Many-to-one comparisons in stratified designs, *Biometrical Journal*, **44**, 101–116

maFitMod *Fit dose-response models via bootstrap model averaging (bagging)*

Description

This function fits dose-response models in a bootstrap model averaging approach motivated by the bagging procedure (Breiman 1996). Given summary estimates for the outcome at each dose, the function samples summary data from the multivariate normal distribution. For each sample dose-response models are fit to these summary estimates and the best model according to the gAIC is selected.

Usage

```
maFitMod(dose, resp, S, models, nSim = 1000, control, bnds, addArgs = NULL)

## S3 method for class 'maFit'
predict(
  object,
  summaryFct = function(x) quantile(x, probs = c(0.025, 0.25, 0.5, 0.75, 0.975)),
  doseSeq = NULL,
  ...
)

## S3 method for class 'maFit'
plot(
  x,
  plotData = c("means", "meansCI", "none"),
  xlab = "Dose",
  ylab = "Response",
  title = NULL,
  level = 0.95,
  trafo = function(x) x,
  lenDose = 201,
  ...
)
```

Arguments

dose	Numeric specifying the dose variable.
resp	Numeric specifying the response estimate corresponding to the doses in dose
S	Covariance matrix associated with the dose-response estimate specified via resp
models	dose-response models to fit
nSim	Number of bootstrap simulations
control	Same as the control argument in fitMod().

bnds	Bounds for non-linear parameters. This needs to be a list with list entries corresponding to the selected bounds. The names of the list entries need to correspond to the model names. The `defBnds()` function provides the default selection.
addArgs	List containing two entries named "scal" and "off" for the "betaMod" and "linlog" model. When addArgs is NULL the following defaults are used '`list(scal = 1.2*max(doses), off = 0.01*max(doses))`'
object	Object of class maFit
summaryFct	If equal to NULL predictions are calculated for each sampled parameter value. Otherwise a summary function is applied to the dose-response predictions for each parameter value. The default is to calculate 0.025, 0.25, 0.5, 0.75, 0.975 quantiles of the predictions for each dose.
doseSeq	Where to calculate predictions.
...	Additional parametes (unused)
x	object of class maFit
plotData	Determines how the original data are plotted: Either as means or as means with CI or not at all. The level of the CI is determined by the argument '`level`'.
xlab	x-axis label
ylab	y-axis label
title	plot title
level	Level for CI, when plotData is equal to '`meansCI`'.
trafo	Plot the fitted models on a transformed scale (e.g. probability scale if models have been fitted on log-odds scale). The default for '`trafo`' is the identity function.
lenDose	Number of grid values to use for display.

Value

An object of class '`maFit`', which contains the fitted dose-response models '`DRMod`' objects, information on which model was selected in each bootstrap and basic input parameters.

Author(s)

Bjoern Bornkamp

References

Breiman, L. (1996). Bagging predictors. Machine learning, 24, 123-140.

See Also

`fitMod()`, `bFitMod()`, `drmodels()`

Examples

```
data(biom)
## produce first stage fit (using dose as factor)
anMod <- lm(resp~factor(dose)-1, data=biom)
drFit <- coef(anMod)
S <- vcov(anMod)
dose <- sort(unique(biom$dose))
## fit an emax and sigEmax model (increase nSim for real use)
mFit <- maFitMod(dose, drFit, S, model = c("emax", "sigEmax"), nSim = 10)
mFit
plot(mFit, plotData = "meansCI")
ED(mFit, direction = "increasing", p = 0.9)
```

MCPMod *MCPMod - Multiple Comparisons and Modeling*

Description

Tests for a dose-response effect using a model-based multiple contrast test (see MCTtest), selects one (or several) model(s) from the significant shapes, fits them using fitMod. For details on the method see Bretz et al. (2005).

Usage

```
MCPMod(dose, resp, data, models, S = NULL, type = c("normal", "general"),
       addCovars = ~1, placAdj = FALSE, selModel = c("AIC", "maxT", "aveAIC"),
       alpha = 0.025, df = NULL, critV = NULL, doseType = c("TD", "ED"),
       Delta, p, pVal = TRUE, alternative = c("one.sided", "two.sided"),
       na.action = na.fail, mvtcontrol = mvtnorm.control(),
       bnds, control = NULL)

## S3 method for class 'MCPMod'
predict(object,
        predType = c("full-model", "ls-means", "effect-curve"),
        newdata = NULL, doseSeq = NULL, se.fit = FALSE, ...)



## S3 method for class 'MCPMod'
plot(x, CI = FALSE, level = 0.95,
       plotData = c("means", "meansCI", "raw", "none"),
       plotGrid = TRUE, colMn = 1, colFit = 1, ...)
```

Arguments

dose, resp	Either vectors of equal length specifying dose and response values, or names of variables in the data frame specified in 'data'.

`data` Data frame containing the variables referenced in dose and resp if 'data' is not specified it is assumed that 'dose' and 'resp' are variables referenced from data (and no vectors)

`models` An object of class '"Mods"', see Mods for details

`S` The covariance matrix of 'resp' when 'type = "general"', see Description.

`type` Determines whether inference is based on an ANCOVA model under a homoscedastic normality assumption (when 'type = "normal"'), or estimates at the doses and their covariance matrix and degrees of freedom are specified directly in 'resp', 'S' and 'df'. See also fitMod and Pinheiro et al. (2014).

`addCovars` Formula specifying additive linear covariates (for 'type = "normal"')

`placAdj` Logical, if true, it is assumed that placebo-adjusted estimates are specified in 'resp' (only possible for 'type = "general"').

`selModel` Optional character vector specifying the model selection criterion for dose estimation. Possible values are

- AIC: Selects model with smallest AIC (this is the default)
- maxT: Selects the model corresponding to the largest t-statistic.
- aveAIC: Uses a weighted average of the models corresponding to the significant contrasts. The model weights are chosen by the formula: $w_i = \exp(-0.5AIC_i)/\sum_i(\exp(-0.5AIC_i))$ See Buckland et al. (1997) for details.

For 'type = "general"' the "gAIC" is used.

`alpha` Significance level for the multiple contrast test

`df` Specify the degrees of freedom to use in case 'type = "general"', for the call to MCTtest and fitMod. Infinite degrees of ('df=Inf') correspond to the multivariate normal distribution. For type = "normal" the degrees of freedom deduced from the AN(C)OVA fit are used and this argument is ignored.

`critV` Supply a pre-calculated critical value. If this argument is NULL, no critical value will be calculated and the test decision is based on the p-values. If 'critV = TRUE' the critical value will be calculated.

`doseType`, `Delta`, `p`
'doseType' determines the dose to estimate, ED or TD (see also Mods), and 'Delta' and 'p' need to be specified depending on whether TD or ED is to be estimated. See TD and ED for details.

`pVal` Logical determining, whether p-values should be calculated.

`alternative` Character determining the alternative for the multiple contrast trend test.

`na.action` A function which indicates what should happen when the data contain NAs.

`mvtcontrol` A list specifying additional control parameters for the 'qmvt' and 'pmvt' calls in the code, see also mvtnorm.control for details.

`bnds` Bounds for non-linear parameters. This needs to be a list with list entries corresponding to the selected bounds. The names of the list entries need to correspond to the model names. The defBnds function provides the default selection.

`control` Control list for the optimization.
A list with entries: "nlminbcontrol", "optimizetol" and "gridSize".
The entry nlminbcontrol needs to be a list and is passed directly to control argument in the nlminb function, that is used internally for models with 2 nonlinear parameters (e.g. sigmoid Emax or beta model).
The entry optimizetol is passed directly to the tol argument of the optimize function, which is used for models with 1 nonlinear parameters (e.g. Emax or exponential model).
The entry gridSize needs to be a list with entries dim1 and dim2 giving the size of the grid for the gridsearch in 1d or 2d models.

`object`, `x` MCPMod object

`predType`, `newdata`, `doseSeq`, `se.fit`, `...`
predType determines whether predictions are returned for the full model (including potential covariates), the ls-means (SAS type) or the effect curve (difference to placebo).
newdata gives the covariates to use in producing the predictions (for '`predType = "full-model"`'), if missing the covariates used for fitting are used.
doseSeq dose-sequence on where to produce predictions (for '`predType = "effect-curve"`' and '`predType = "ls-means"`'). If missing the doses used for fitting are used.
se.fit: logical determining, whether the standard error should be calculated.
...: Additional arguments, for plot.MCPMod these are passed to plot.DRMod.

`CI`, `level`, `plotData`, `plotGrid`, `colMn`, `colFit`
Arguments for plot method: '`CI`' determines whether confidence intervals should be plotted. '`level`' determines the level of the confidence intervals. '`plotData`' determines how the data are plotted: Either as means or as means with CI, raw data or none. In case of '`type = "normal"`' and covariates the ls-means are displayed, when '`type = "general"`' the option "raw" is not available. '`colMn`' and '`colFit`' determine the colors of fitted model and the raw means.

Value

An object of class '`MCPMod`', which contains the fitted '`MCTtest`' object as well as the '`DRMod`' objects and additional information (model selection criteria, dose estimates, selected models).

Author(s)

Bjoern Bornkamp

References

Bretz, F., Pinheiro, J. C., and Branson, M. (2005), Combining multiple comparisons and modeling techniques in dose-response studies, *Biometrics*, **61**, 738–748

Pinheiro, J. C., Bornkamp, B., and Bretz, F. (2006). Design and analysis of dose finding studies combining multiple comparisons and modeling procedures, *Journal of Biopharmaceutical Statistics*, **16**, 639–656

Pinheiro, J. C., Bretz, F., and Branson, M. (2006). Analysis of dose-response studies - modeling approaches, *in* N. Ting (ed.). *Dose Finding in Drug Development*, Springer, New York, pp. 146–171

Pinheiro, J. C., Bornkamp, B., Glimm, E. and Bretz, F. (2014) Model-based dose finding under model uncertainty using general parametric models, *Statistics in Medicine*, **33**, 1646–1661

Schorning, K., Bornkamp, B., Bretz, F., & Dette, H. (2016). Model selection versus model averaging in dose finding studies. *Statistics in Medicine*, **35**, 4021–4040

Xun, X. and Bretz, F. (2017) The MCP-Mod methodology: Practical Considerations and The DoseFinding R package, in O'Quigley, J., Iasonos, A. and Bornkamp, B. (eds) Handbook of methods for designing, monitoring, and analyzing dose-finding trials, CRC press

Buckland, S. T., Burnham, K. P. and Augustin, N. H. (1997). Model selection an integral part of inference, *Biometrics*, **53**, 603–618

Seber, G.A.F. and Wild, C.J. (2003). Nonlinear Regression, Wiley.

See Also

MCTtest, fitMod, drmodels

Examples

```
data(biom)
## first define candidate model set (only need "standardized" models)
models <- Mods(linear = NULL, emax=c(0.05,0.2), linInt=c(1, 1, 1, 1),
               doses=c(0,0.05,0.2,0.6,1))
plot(models)
## perform MCPMod procedure
MM <- MCPMod(dose, resp, biom, models, Delta=0.5)
## a number of things can be done with an MCPMod object
MM # print method provides basic information
summary(MM) # more information
## predict all significant dose-response models
predict(MM, se.fit=TRUE, doseSeq=c(0,0.2,0.4, 0.9, 1),
        predType="ls-means")
## display all model functions
plot(MM, plotData="meansCI", CI=TRUE)

## now perform model-averaging
MM2 <- MCPMod(dose, resp, biom, models, Delta=0.5, selModel = "aveAIC")
sq <- seq(0,1,length=11)
pred <- predict(MM, doseSeq=sq, predType="ls-means")
modWeights <- MM2$selMod
## model averaged predictions
pred <- do.call("cbind", pred)%*%modWeights
## model averaged dose-estimate
TDEst <- MM2$doseEst%*%modWeights

## now an example using a general fit and fitting based on placebo
## adjusted first-stage estimates
data(IBScovars)
## ANCOVA fit model including covariates
anovaMod <- lm(resp~factor(dose)+gender, data=IBScovars)
drFit <- coef(anovaMod)[2:5] # placebo adjusted estimates at doses
vCov <- vcov(anovaMod)[2:5,2:5]
```

```
dose <- sort(unique(IBScovars$dose))[-1] # no estimate for placebo
## candidate models
models <- Mods(emax = c(0.5, 1), betaMod=c(1,1), doses=c(0,4))
plot(models)
## hand over placebo-adjusted estimates drFit to MCPMod
MM3 <- MCPMod(dose, drFit, S=vCov, models = models, type = "general",
              placAdj = TRUE, Delta=0.2)
plot(MM3, plotData="meansCI")

## The first example, but with critical value handed over
## this is useful, e.g. in simulation studies
MM4 <- MCPMod(dose, resp, biom, models, Delta=0.5, critV = 2.31)
```

`MCTpval` *Calculate multiplicity adjusted p-values for multiple contrast test*

Description

Calculate multiplicity adjusted p-values for a maximum contrast test corresponding to a set of contrasts and given a set of observed test statistics. This function is exported as it may be a useful building block and used in more complex testing situations that are not covered by `MCTtest()`. Most users probably don't need to use this function.

Usage

```
MCTpval(
  contMat,
  corMat,
  df,
  tStat,
  alternative = c("one.sided", "two.sided"),
  control = mvtnorm.control()
)
```

Arguments

`contMat`	Contrast matrix to use. The individual contrasts should be saved in the columns of the matrix
`corMat`	Correlation matrix of contrasts
`df`	Degrees of freedom to use for calculation.
`tStat`	Vector of contrast test statistics
`alternative`	Character determining the alternative for the multiple contrast trend test.
`control`	A list specifying additional control parameters for the 'qmvt' and 'pmvt' calls in the code, see also `mvtnorm.control()` for details.

Value

Numeric containing the calculated p-values.

Author(s)

Bjoern Bornkamp

References

Pinheiro, J. C., Bornkamp, B., and Bretz, F. (2006). Design and analysis of dose finding studies combining multiple comparisons and modeling procedures, *Journal of Biopharmaceutical Statistics*, **16**, 639–656

See Also

MCTtest(), optContr()

Examples

```
data(biom)
## define shapes for which to calculate optimal contrasts
modlist <- Mods(emax = 0.05, linear = NULL, logistic = c(0.5, 0.1),
                linInt = c(0, 1, 1, 1), doses = c(0, 0.05, 0.2, 0.6, 1))
contMat <- optContr(modlist, w=1)$contMat
## calculate inputs needed for MCTpval
fit <- lm(resp~factor(dose)-1, data=biom)
est <- coef(fit)
vc <- vcov(fit)
ct <- as.vector(est %*% contMat)
covMat <- t(contMat) %*% vc %*% contMat
den <- sqrt(diag(covMat))
tStat <- ct/den
corMat <- cov2cor(t(contMat) %*% vc %*% contMat)
MCTpval(contMat, corMat, df=100-5, tStat)
## compare to
test <- MCTtest(dose, resp, biom, models=modlist)
attr(test$tStat, "pVal")
```

MCTtest *Performs multiple contrast test*

Description

This function performs a multiple contrast test. The contrasts are either directly specified in 'contMat' or optimal contrasts derived from the 'models' argument. The directionality of the data (i.e. whether an increase or decrease in the response variable is beneficial is inferred from the 'models' object, see Mods).

For 'type = "normal"' an ANCOVA model based on a homoscedastic normality assumption (with additive covariates specified in 'addCovars') is fitted.

For 'type = "general"' it is assumed multivariate normally distributed estimates are specified in 'resp' with covariance given by 'S', and the contrast test statistic is calculated based on this assumption. Degrees of freedom specified in 'df'.

Usage

```
MCTtest(dose, resp, data = NULL, models, S = NULL, type = c("normal", "general"),
        addCovars = ~1, placAdj = FALSE, alpha = 0.025, df = NULL,
        critV = NULL, pVal = TRUE,
        alternative = c("one.sided", "two.sided"), na.action = na.fail,
        mvtcontrol = mvtnorm.control(), contMat = NULL)
```

Arguments

dose, resp	Either vectors of equal length specifying dose and response values, or names of variables in the data frame specified in 'data'.
data	Data frame containing the variables referenced in dose and resp if 'data' is not specified it is assumed that 'dose' and 'resp' are variables referenced from data (and no vectors)
models	An object of class 'Mods', see Mods for details
S	The covariance matrix of 'resp' when 'type = "general"', see Description.
type	Determines whether inference is based on an ANCOVA model under a homoscedastic normality assumption (when 'type = "normal"'), or estimates at the doses and their covariance matrix and degrees of freedom are specified directly in 'resp', 'S' and 'df'. See also fitMod and Pinheiro et al. (2014).
addCovars	Formula specifying additive linear covariates (for 'type = "normal"')
placAdj	Logical, if true, it is assumed that placebo-adjusted estimates are specified in 'resp' (only possible for 'type = "general"').
alpha	Significance level for the multiple contrast test
df	Specify the degrees of freedom to use in case 'type = "general"'. If this argument is missing 'df = Inf' is used (which corresponds to the multivariate normal distribution). For type = "normal" the degrees of freedom deduced from the AN(C)OVA fit are used and this argument is ignored.
critV	Supply a pre-calculated critical value. If this argument is NULL, no critical value will be calculated and the test decision is based on the p-values. If 'critV = TRUE' the critical value will be calculated.
pVal	Logical determining, whether p-values should be calculated.
alternative	Character determining the alternative for the multiple contrast trend test.
na.action	A function which indicates what should happen when the data contain NAs.
mvtcontrol	A list specifying additional control parameters for the 'qmvt' and 'pmvt' calls in the code, see also mvtnorm.control for details.
contMat	Contrast matrix to apply to the ANCOVA dose-response estimates. The contrasts need to be in the columns of the matrix (i.e. the column sums need to be 0).

Details

Integrals over the multivariate t and multivariate normal distribution are calculated using the 'mvtnorm' package.

Value

An object of class MCTtest, a list containing the output.

Author(s)

Bjoern Bornkamp

References

Hothorn, T., Bretz, F., and Westfall, P. (2008). Simultaneous Inference in General Parametric Models, *Biometrical Journal*, **50**, 346–363

Pinheiro, J. C., Bornkamp, B., Glimm, E. and Bretz, F. (2014) Model-based dose finding under model uncertainty using general parametric models, *Statistics in Medicine*, **33**, 1646–1661

See Also

powMCT, optContr

Examples

```
## example without covariates
data(biom)
## define shapes for which to calculate optimal contrasts
modlist <- Mods(emax = 0.05, linear = NULL, logistic = c(0.5, 0.1),
                linInt = c(0, 1, 1, 1), doses = c(0, 0.05, 0.2, 0.6, 1))
m1 <- MCTtest(dose, resp, biom, models=modlist)
## now calculate critical value (but not p-values)
m2 <- MCTtest(dose, resp, biom, models=modlist, critV = TRUE, pVal = FALSE)
## now hand over critical value
m3 <- MCTtest(dose, resp, biom, models=modlist, critV = 2.24)

## example with covariates
data(IBScovars)
modlist <- Mods(emax = 0.05, linear = NULL, logistic = c(0.5, 0.1),
                linInt = c(0, 1, 1, 1), doses = c(0, 1, 2, 3, 4))
MCTtest(dose, resp, IBScovars, models = modlist, addCovars = ~gender)

## example using general approach (fitted on placebo-adjusted scale)
ancMod <- lm(resp~factor(dose)+gender, data=IBScovars)
## extract estimates and information to feed into MCTtest
drEst <- coef(ancMod)[2:5]
vc <- vcov(ancMod)[2:5, 2:5]
doses <- 1:4
MCTtest(doses, drEst, S = vc, models = modlist, placAdj = TRUE,
        type = "general", df = Inf)

## example with general alternatives handed over
data(biom)
## calculate contrast matrix for the step-contrasts
## represent them as linInt models
models <- Mods(linInt=rbind(c(1,1,1,1),
```

```
                            c(0,1,1,1),
                            c(0,0,1,1),
                            c(0,0,0,1)),
                 doses=c(0,0.05,0.2,0.6,1))
plot(models)
## now calculate optimal contrasts for these means
## use weights from actual sample sizes
weights <- as.numeric(table(biom$dose))
contMat <- optContr(models, w = weights)
## plot contrasts
plot(contMat)
## perform multiple contrast test
MCTtest(dose, resp, data=biom, contMat = contMat)

## example for using the Dunnett contrasts
## Dunnett contrasts
doses <- sort(unique(biom$dose))
contMat <- rbind(-1, diag(4))
rownames(contMat) <- doses
colnames(contMat) <- paste("D", doses[-1], sep="")
MCTtest(dose, resp, data=biom, contMat = contMat)
```

`migraine` *Migraine Dose Response data*

Description

Data set obtained from clinicaltrials.gov (NCT00712725). This was randomized placebo controlled dose-response trial for treatment of acute migraine. The primary endpoint was "pain freedom at 2 hours postdose" (a binary measurement).

Usage

```
data(migraine)
```

Format

A data frame with 517 columns corresponding to the patients that completed the trial

`dose` a numeric vector containing the dose values

`painfree` number of treatment responders

`ntrt` number of subject per treatment group

Source

http://clinicaltrials.gov/ct2/show/results/NCT00712725

Mods *Define dose-response models*

Description

The Mods functions allows to define a set of dose-response models. The function is used as input object for a number of other different functions.

Usage

```
Mods(
  ...,
  doses,
  placEff = 0,
  maxEff,
  direction = c("increasing", "decreasing"),
  addArgs = NULL,
  fullMod = FALSE
)

getResp(fmodels, doses)

plotMods(
  ModsObj,
  nPoints = 200,
  superpose = FALSE,
  xlab = "Dose",
  ylab = "Model means",
  modNams = NULL,
  trafo = function(x) x
)

## S3 method for class 'Mods'
plot(
  x,
  nPoints = 200,
  superpose = FALSE,
  xlab = "Dose",
  ylab = "Model means",
  modNams = NULL,
  plotTD = FALSE,
  Delta,
  ...
)
```

Arguments

...	In function Mods: Dose-response model names with parameter values specifying the guesstimates for the θ_2 parameters. See drmodels() for a complete list of dose-response models implemented. See below for an example specification. In function plot.Mods: Additional arguments to the 'xyplot' call.
doses	Dose levels to be used, this needs to include placebo.
placEff, maxEff	Specify used placebo effect and the maximum effect over placebo. Either a numeric vector of the same size as the number of candidate models or of length one. When these parameters are not specified 'placEff = 0' is assumed, for 'maxEff = 1' is assumed, if 'direction = "increasing"' and 'maxEff = -1' is assumed, for 'direction = "decreasing"'.
direction	Character determining whether the beneficial direction is 'increasing' or 'decreasing' with increasing dose levels. This argument is ignored if 'maxEff' is specified.
addArgs	List containing two entries named "scal" and "off" for the "betaMod" and "linlog". When addArgs is NULL the following defaults are used 'list(scal = 1.2*max(doses), off = 0.01*max(doses), nodes = doses)'.
fullMod	Logical determining, whether the model parameters specified in the Mods function (via the ... argument) should be interpreted as standardized or the full model parameters.
fmodels	An object of class Mods
ModsObj	For function 'plotMods' the 'ModsObj' should contain an object of class 'Mods'.
nPoints	Number of points for plotting
superpose	Logical determining, whether model plots should be superposed
xlab, ylab	Label for y-axis and x-axis.
modNams	When 'modNams == NULL', the names for the panels are determined by the underlying model functions, otherwise the contents of 'modNams' are used.
trafo	For function 'plotMods' there is the option to plot the candidate model set on a transformed scale (e.g. probability scale if the candidate models are formulated on log-odds scale). The default for 'trafo' is the identity function.
x	Object of class Mods with type Mods
plotTD	'plotTD' is a logical determining, whether the TD should be plotted. 'Delta' is the target effect to estimate for the TD.
Delta	Delta: The target effect size use for the target dose (TD) (Delta should be > 0).

Details

The dose-response models used in this package (see drmodels() for details) are of form

$$f(d) = \theta_0 + \theta_1 f^0(d, \theta_2)$$

where the parameter θ_2 is the only non-linear parameter and can be one- or two-dimensional, depending on the used model.

One needs to hand over the effect at placebo and the maximum effect in the dose range, from which θ_0, θ_1 are then back-calculated, the output object is of class '"Mods"'. This object can form the input for other functions to extract the mean response ('getResp') or target doses (TD() and ED()) corresponding to the models. It is also needed as input to the functions powMCT(), optDesign()

Some models, for example the beta model ('scal') and the linlog model ('off') have parameters that are not estimated from the data, they need to be specified via the 'addArgs' argument.

The default plot method for 'Mods' objects is based on a plot using the 'lattice' package for backward compatibility. The function 'plotMods' function implements a plot using the 'ggplot2' package.

NOTE: If a decreasing effect is beneficial for the considered response variable it needs to specified here, either by using 'direction = "decreasing"' or by specifying a negative "maxEff" argument.

Value

Returns an object of class '"Mods"'. The object contains the specified model parameter values and the derived linear parameters (based on '"placEff"' and '"maxEff"') in a list.

Author(s)

Bjoern Bornkamp

References

Pinheiro, J. C., Bornkamp, B., and Bretz, F. (2006). Design and analysis of dose finding studies combining multiple comparisons and modeling procedures, *Journal of Biopharmaceutical Statistics*, **16**, 639–656

See Also

Mods(), drmodels(), optDesign(), powMCT()

Examples

```
## Example on how to specify candidate models

## Suppose one would like to use the following models with the specified
## guesstimates for theta2, in a situation where the doses to be used are
## 0, 0.05, 0.2, 0.6, 1

## Model            guesstimate(s) for theta2 parameter(s) (name)
## linear           -
## linear in log    -
## Emax             0.05 (ED50)
## Emax             0.3 (ED50)
## exponential      0.7 (delta)
## quadratic       -0.85 (delta)
## logistic         0.4  0.09 (ED50, delta)
## logistic         0.3  0.1 (ED50, delta)
```

```
## betaMod          0.3  1.3 (delta1, delta2)
## sigmoid Emax     0.5  2 (ED50, h)
## linInt           0.5 0.75 1 1 (perc of max-effect at doses)
## linInt           0.5 1 0.7 0.5 (perc of max-effect at doses)

## for the linInt model one specifies the effect over placebo for
## each active dose.
## The fixed "scal" parameter of the betaMod is set to 1.2
## The fixed "off"  parameter of the linlog is set to 0.1
## These (standardized) candidate models can be specified as follows

models <- Mods(linear = NULL, linlog = NULL, emax = c(0.05, 0.3),
               exponential = 0.7, quadratic = -0.85,
               logistic = rbind(c(0.4, 0.09), c(0.3, 0.1)),
               betaMod = c(0.3, 1.3), sigEmax = c(0.5, 2),
               linInt = rbind(c(0.5, 0.75, 1, 1), c(0.5, 1, 0.7, 0.5)),
               doses = c(0, 0.05, 0.2, 0.6, 1),
               addArgs = list(scal=1.2, off=0.1))
## "models" now contains the candidate model set, as placEff, maxEff and
## direction were not specified a placebo effect of 0 and an effect of 1
## is assumed

## display of specified candidate set using default plot (based on lattice)
plot(models)
## display using ggplot2
plotMods(models)

## example for creating a candidate set with decreasing response
doses <- c(0, 10, 25, 50, 100, 150)
fmodels <- Mods(linear = NULL, emax = 25,
                  logistic = c(50, 10.88111), exponential = 85,
                  betaMod = rbind(c(0.33, 2.31), c(1.39, 1.39)),
                  linInt = rbind(c(0, 1, 1, 1, 1),
                                 c(0, 0, 1, 1, 0.8)),
                  doses=doses, placEff = 0.5, maxEff = -0.4,
                  addArgs=list(scal=200))
plot(fmodels)
plotMods(fmodels)
## some customizations (different model names, symbols, line-width)
plot(fmodels, lwd = 3, pch = 3, cex=1.2, col="red",
     modNams = paste("mod", 1:8, sep="-"))

## for a full-model object one can calculate the responses
## in a matrix
getResp(fmodels, doses=c(0, 20, 100, 150))

## calculate doses giving an improvement of 0.3 over placebo
TD(fmodels, Delta=0.3, direction = "decreasing")
## discrete version
TD(fmodels, Delta=0.3, TDtype = "discrete", doses=doses, direction = "decreasing")
## doses giving 50% of the maximum effect
ED(fmodels, p=0.5)
ED(fmodels, p=0.5, EDtype = "discrete", doses=doses)
```

```
plot(fmodels, plotTD = TRUE, Delta = 0.3)

## example for specifying all model parameters (fullMod=TRUE)
fmods <- Mods(emax = c(0, 1, 0.1), linear = cbind(c(-0.4,0), c(0.2,0.1)),
              sigEmax = c(0, 1.1, 0.5, 3),
              doses = 0:4, fullMod = TRUE)
getResp(fmods, doses=seq(0,4,length=11))
## calculate doses giving an improvement of 0.3 over placebo
TD(fmods, Delta=0.3)
## discrete version
TD(fmods, Delta=0.3, TDtype = "discrete", doses=0:4)
## doses giving 50% of the maximum effect
ED(fmods, p=0.5)
ED(fmods, p=0.5, EDtype = "discrete", doses=0:4)
plot(fmods)
```

`mvpostmix` *Prior to posterior updating for a multivariate normal mixture*

Description

Calculate conjugate posterior mixture of multivariate normals with known covariance matrix

Usage

```
mvpostmix(priormix, mu_hat, S_hat)
```

Arguments

`priormix`	Prior multivariate normal mixture given as a list of length 3. The first list entry contains the mixture weights, the second component the mean vectors and the third component of the list the covariance matrices.
`mu_hat`	estimated mean response for each dose
`S_hat`	estimated covariance matrix

Value

Returns a posterior multivariate normal mixture as a list of length 3, containing mixture weights, mean vectors and covariance matrices.

Author(s)

Marius Thomas

References

Bernardo, J. M., and Smith, A. F. (1994). Bayesian theory. John Wiley & Sons.

mvtnorm-control *Control options for pmvt and qmvt functions*

Description

Returns a list (an object of class "GenzBretz") with control parameters for the 'pmvt' and 'qmvt' functions from the 'mvtnorm' package. Note that the DoseFinding package always uses "GenzBretz" algorithm. See the mvtnorm documentation for more information.

Usage

```
mvtnorm.control(maxpts = 30000, abseps = 0.001, releps = 0, interval = NULL)
```

Arguments

maxpts	Maximum number of function values as integer.
abseps	Absolute error tolerance as double.
releps	Relative error tolerance as double.
interval	Interval to be searched, when the quantile is calculated.

neurodeg *Neurodegenerative disease simulated longitudinal dose-finding data set*

Description

This simulated data set is motivated by a real Phase 2 clinical study of a new drug for a neurodegenerative disease. The state of the disease is measured through a functional scale, with smaller values corresponding to more severe neurodeterioration. The goal of the drug is to reduce the rate of disease progression, which is measured by the linear slope of the functional scale over time.

Usage

```
data(neurodeg)
```

Format

A data frame with 100 observations on the following 2 variables.

`resp` a numeric vector containing the response values

`dose` a numeric vector containing the dose values

`id` Patient ID

`time` time of measurement

Details

The trial design includes placebo and four doses: 1, 3, 10, and 30 mg, with balanced allocation of 50 patients per arm. Patients are followed up for one year, with measurements of the functional scale being taken at baseline and then every three months.

The functional scale response is assumed to be normally distributed and, based on historical data, it is believed that the longitudinal progression of the functional scale over the one year of follow up can be modeled a simple linear trend. See the example below on how to analyse this type of data.

This data set was used in Pinheiro et al. (2014) to illustrate the generalized MCPMod methodology.

Source

Pinheiro, J. C., Bornkamp, B., Glimm, E. and Bretz, F. (2014) Model-based dose finding under model uncertainty using general parametric models, *Statistics in Medicine*, **33**, 1646–1661

Examples

```
## Not run:
## reproduce analysis from Pinheiro et al. (2014)
data(neurodeg)
## first fit the linear mixed effect model
library(nlme)
fm <- lme(resp ~ as.factor(dose):time, neurodeg, ~time|id, method = "ML")
muH <- fixef(fm)[-1] # extract estimates
covH <- vcov(fm)[-1,-1]

## derive optimal contrasts for candidate shapes
doses <- c(0, 1, 3, 10, 30)
mod <- Mods(emax = 1.11, quadratic= -0.022, exponential = 8.867,
            linear = NULL, doses = doses) #
contMat <- optContr(mod, S=covH) # calculate optimal contrasts
## multiple contrast test
MCTtest(doses, muH, S=covH, type = "general", critV = TRUE,
        contMat=contMat)
## fit the emax model
fitMod(doses, muH, S=covH, model="emax", type = "general",
       bnds=c(0.1, 10))


## alternatively one can also fit the model using nlme
nlme(resp ~ b0 + (e0 + eM * dose/(ed50 + dose))*time, neurodeg,
     fixed = b0 + e0 + eM + ed50 ~ 1, random = b0 + e0 ~ 1 | id,
     start = c(200, -4.6, 1.6, 3.2))
## both approaches lead to rather similar results

## End(Not run)
```

optContr *Calculate optimal contrasts*

Description

This function calculates a contrast vectors that are optimal for detecting certain alternatives. The contrast is optimal in the sense of maximizing the non-centrality parameter of the underlying contrast test statistic:

$$\frac{c'\mu}{\sqrt{c'Sc}}$$

Here μ is the mean vector under the alternative and S the covariance matrix associated with the estimate of μ. The optimal contrast is given by

$$c^{opt} \propto S^{-1}\left(\mu - \frac{\mu' S^{-1}1}{1' S^{-1}1}\right),$$

see Pinheiro et al. (2014).

Usage

```
optContr(
  models,
  doses,
  w,
  S,
  placAdj = FALSE,
  type = c("unconstrained", "constrained")
)

## S3 method for class 'optContr'
plot(
  x,
  superpose = TRUE,
  xlab = "Dose",
  ylab = NULL,
  plotType = c("contrasts", "means"),
  ...
)

plotContr(optContrObj, xlab = "Dose", ylab = "Contrast coefficients")
```

Arguments

models	An object of class 'Mods' defining the dose-response shapes for which to calculate optimal contrasts.
doses	Optional argument. If this argument is missing the doses attribute in the 'Mods' object specified in 'models' is used.

`w`, `S` Arguments determining the matrix S used in the formula for the optimal contrasts. Exactly one of '`w`' and '`S`' has to be specified. Note that '`w`' and '`S`' only have to be specified up to proportionality

w Vector specifying weights for the different doses, in the formula for calculation of the optimal contrasts. Specifying a weights vector is equivalent to specifying S=diag(1/w) (e.g. in a homoscedastic case with unequal sample sizes, '`w`' should be proportional to the group sample sizes).

S Directly specify a matrix proportional to the covariance matrix to use.

`placAdj` Logical determining, whether the contrasts should be applied to placebo-adjusted estimates. If yes the returned coefficients are no longer contrasts (i.e. do not sum to 0). However, the result of multiplying of this "contrast" matrix with the placebo adjusted estimates, will give the same results as multiplying the original contrast matrix to the unadjusted estimates.

`type` For '`type = "constrained"`' the contrast coefficients of the zero dose group are constrained to be different from the coefficients of the active treatment groups. So that a weighted sum of the active treatments is compared against the zero dose group. For an increasing trend the coefficient of the zero dose group is negative and all other coefficients have to be positive (for a decreasing trend the other way round).

`x`, `superpose`, `xlab`, `ylab`, `plotType`
Arguments for the plot method for optContr objects. plotType determines, whether the contrasts or the underlying (standardized) mean matrix should be plotted.

`...` Additional arguments for plot method

`optContrObj` For function '`plotContr`' the '`optContrObj`' should contain an object of class '`optContr`'.

Details

Note that the directionality (i.e. whether in "increase" in the response variable is beneficial or a "decrease", is inferred from the specified '`models`' object, see `Mods()` for details).

Constrained contrasts (type = "constrained") add the additional constraint in the optimization that the sign of the contrast coefficient for control and active treatments need to be different. The quadratic programming algorithm from the quadprog package is used to calculate the contrasts.

Value

Object of class '`optContr`'. A list containing entries contMat and muMat (i.e. contrast, mean and correlation matrix).

Author(s)

Bjoern Bornkamp

References

Bretz, F., Pinheiro, J. C., and Branson, M. (2005), Combining multiple comparisons and modeling techniques in dose-response studies, *Biometrics*, **61**, 738–748

Pinheiro, J. C., Bornkamp, B., Glimm, E. and Bretz, F. (2014) Model-based dose finding under model uncertainty using general parametric models, *Statistics in Medicine*, **33**, 1646–1661

See Also

`MCTtest()`

Examples

```
doses <- c(0,10,25,50,100,150)
models <- Mods(linear = NULL, emax = 25,
               logistic = c(50, 10.88111), exponential= 85,
               betaMod=rbind(c(0.33,2.31), c(1.39,1.39)),
               doses = doses, addArgs = list(scal = 200))
contMat <- optContr(models, w = rep(50,6))
plot(contMat)
plotContr(contMat) # display contrasts using ggplot2

## now we would like the "contrasts" for placebo adjusted estimates
dosPlac <- doses[-1]
## matrix proportional to cov-matrix of plac. adj. estimates for balanced data
S <- diag(5)+matrix(1, 5,5)
## note that we explicitly hand over the doses here
contMat0 <- optContr(models, doses=dosPlac, S = S, placAdj = TRUE)
## -> contMat0 is no longer a contrast matrix (columns do not sum to 0)
colSums(contMat0$contMat)
## calculate contrast matrix for unadjusted estimates from this matrix
## (should be same as above)
aux <- rbind(-colSums(contMat0$contMat), contMat0$contMat)
t(t(aux)/sqrt(colSums(aux^2))) ## compare to contMat$contMat

## now calculate constrained contrasts
if(requireNamespace("quadprog", quietly = TRUE)){
optContr(models, w = rep(50,6), type = "constrained")
optContr(models, doses=dosPlac, S = S, placAdj = TRUE,
         type = "constrained")
}
```

`optDesign` *Function to calculate optimal designs*

Description

Given a set of models (with full parameter values and model probabilities) the '`optDesign`' function calculates the optimal design for estimating the dose-response model parameters (D-optimal) or the design for estimating the target dose (TD-optimal design) (see Dette, Bretz, Pepelyshev and

Pinheiro (2008)), or a mixture of these two criteria. The design can be plotted (together with the candidate models) using 'plot.design'. 'calcCrit' calculates the design criterion for a discrete set of design(s). 'rndDesign' provides efficient rounding for the calculated continous design to a finite sample size.

Usage

```
optDesign(
  models,
  probs,
  doses,
  designCrit = c("Dopt", "TD", "Dopt&TD", "userCrit"),
  Delta,
  standDopt = TRUE,
  weights,
  nold = rep(0, length(doses)),
  n,
  control = list(),
  optimizer = c("solnp", "Nelder-Mead", "nlminb", "exact"),
  lowbnd = rep(0, length(doses)),
  uppbnd = rep(1, length(doses)),
  userCrit,
  ...
)

calcCrit(
  design,
  models,
  probs,
  doses,
  designCrit = c("Dopt", "TD", "Dopt&TD"),
  Delta,
  standDopt = TRUE,
  weights,
  nold = rep(0, length(doses)),
  n
)

rndDesign(design, n, eps = 1e-04)

## S3 method for class 'DRdesign'
plot(x, models, lwdDes = 10, colDes = rgb(0, 0, 0, 0.3), ...)
```

Arguments

models	An object of class 'c(Mods, fullMod)', see the Mods() function for details. When an TD optimal design should be calculated, the TD needs to exist for all models. If a D-optimal design should be calculated, you need at least as many doses as there are parameters in the specified models.

probs — Vector of model probabilities for the models specified in 'models', assumed in the same order as specified in models

doses — Optional argument. If this argument is missing the doses attribute in the 'c(Mods, fullMod)' object specified in 'models' is used.

designCrit — Determines which type of design to calculate. "TD&Dopt" uses both optimality criteria with equal weight.

Delta — Target effect needed for calculating "TD" and "TD&Dopt" type designs.

standDopt — Logical determining, whether the D-optimality criterion (specifically the log-determinant) should be standardized by the number of parameters in the model or not (only of interest if type = "Dopt" or type = "TD&Dopt"). This is of interest, when there is more than one model class in the candidate model set (traditionally standardization this is done in the optimal design literature).

weights — Vector of weights associated with the response at the doses. Needs to be of the same length as the 'doses'. This can be used to calculate designs for heteroscedastic or for generalized linear model situations.

nold, n — When calculating an optimal design at an interim analysis, 'nold' specifies the vector of sample sizes already allocated to the different doses, and 'n' gives sample size for the next cohort.

For 'optimizer = "exact"' one always needs to specify the total sample size via 'n'.

control — List containing control parameters passed down to numerical optimization algorithms (optim(), nlminb() or solnp function).

For 'type = "exact"' this should be a list with possible entries 'maxvls1' and 'maxvls2', determining the maximum number of designs allowed for passing to the criterion function (default 'maxvls2=1e5') and for creating the initial unrestricted matrix of designs (default 'maxvls1=1e6'). In addition there can be an entry 'groupSize' in case the patients are allocated a minimum group size is required.

optimizer — Algorithm used for calculating the optimal design. Options "Nelder-Mead" and "nlminb" use the optim() and nlminb() function and use a trigonometric transformation to turn the constrained optimization problem into an unconstrained one (see Atkinson, Donev and Tobias, 2007, pages 130,131).

Option "solnp" uses the solnp function from the Rsolnp package, which implements an optimizer for non-linear optimization under general constraints.

Option "exact" tries all given combinations of 'n' patients to the given dose groups (subject to the bounds specified via 'lowbnd' and 'uppbnd') and reports the best design. When patients are only allowed to be allocated in groups of a certain 'groupSize', this can be adjusted via the control argument. 'n/groupSize' and 'length(doses)' should be rather small for this approach to be feasible.

When the number of doses is small (<8) usually '"Nelder-Mead"' and '"nlminb"' are best suited ('"nlminb"' is usually a bit faster but less stable than '"Nelder-Mead"'). For a larger number of doses '"solnp"' is the most reliable option (but also slowest) ('"Nelder-Mead"' and '"nlminb"' often fail). When the sample size is small '"exact"' provides the optimal solution rather quickly.

lowbnd, uppbnd	Vectors of the same length as dose vector specifying upper and lower limits for the allocation weights. This option is only available when using the "solnp" and "exact" optimizers.
userCrit	User defined design criterion, should be a function that given a vector of allocation weights and the doses returns the criterion function. When specified 'models' does not need to be handed over. The first argument of 'userCrit' should be the vector of design weights, while the second argument should be the 'doses' argument (see example below). Additional arguments to 'userCrit' can be passed via ...
...	For function 'optDesign' these are additional arguments passed to 'userCrit'. For function 'plot.design' these are additional parameters passed to plot.Mods().
design	Argument for 'rndDesign' and 'calcCrit' functions: Numeric vector (or matrix) of allocation weights for the different doses. The rows of the matrices need to sum to 1. Alternatively also an object of class "DRdesign" can be used for 'rndDesign'. Note that there should be at least as many design points available as there are parameters in the dose-response models selected in models (otherwise the code returns an NA).
eps	Argument for 'rndDesign' function: Value under which elements of w will be regarded as 0.
x	Object of class 'DRdesign' (for 'plot.design')
lwdDes, colDes	Line width and color of the lines plotted for the design (in 'plot.design')

Details

Let M_m denote the Fisher information matrix under model m (up to proportionality). M_m is given by $\sum a_i w_i g_i^T g_i$, where a_i is the allocation weight to dose i, w_i the weight for dose i specified via 'weights' and g_i the gradient vector of model m evaluated at dose i.

For 'designCrit = "Dopt"' the code minimizes the design criterion

$$-\sum_m p_m/k_m \log(\det(M_m))$$

where p_m is the probability for model m and k_m is the number of parameters for model m. When 'standDopt = FALSE' the k_m are all assumed to be equal to one.

For 'designCrit = "TD"' the code minimizes the design criterion

$$\sum_m p_m \log(v_m)$$

where p_m is the probability for model m and v_m is proportional to the asymptotic variance of the TD estimate and given by $b'_m M_m^- b_m$ (see Dette et al. (2008), p. 1227 for details).

For 'designCrit = "Dopt&TD"' the code minimizes the design criterion

$$\sum_m p_m(-0.5 \log(\det(M_m))/k_m + 0.5 \log(v_m))$$

Again, for 'standDopt = FALSE' the k_m are all assumed to be equal to one.

For details on the 'rndDesign' function, see Pukelsheim (1993), Chapter 12.

Note

In some cases (particularly when the number of doses is large, e.g. 7 or larger) it might be necessary to allow a larger number of iterations in the algorithm (via the argument 'control'), particularly for the Nelder-Mead algorithm. Alternatively one can use the solnp optimizer that is usually the most reliable, but not fastest option.

Author(s)

Bjoern Bornkamp

References

Atkinson, A.C., Donev, A.N. and Tobias, R.D. (2007). Optimum Experimental Designs, with SAS, Oxford University Press

Dette, H., Bretz, F., Pepelyshev, A. and Pinheiro, J. C. (2008). Optimal Designs for Dose Finding Studies, *Journal of the American Statisical Association*, **103**, 1225–1237

Pinheiro, J.C., Bornkamp, B. (2017) Designing Phase II Dose-Finding Studies: Sample Size, Doses and Dose Allocation Weights, in O'Quigley, J., Iasonos, A. and Bornkamp, B. (eds) Handbook of methods for designing, monitoring, and analyzing dose-finding trials, CRC press

Pukelsheim, F. (1993). Optimal Design of Experiments, Wiley

See Also

Mods(), drmodels()

Examples

```
## calculate designs for Emax model
doses <- c(0, 10, 100)
emodel <- Mods(emax = 15, doses=doses, placEff = 0, maxEff = 1)
optDesign(emodel, probs = 1)
## TD-optimal design
optDesign(emodel, probs = 1, designCrit = "TD", Delta=0.5)
## 50-50 mixture of Dopt and TD
optDesign(emodel, probs = 1, designCrit = "Dopt&TD", Delta=0.5)
## use dose levels different from the ones specified in emodel object
des <- optDesign(emodel, probs = 1, doses = c(0, 5, 20, 100))
## plot models overlaid by design
plot(des, emodel)
## round des to a sample size of exactly 90 patients
rndDesign(des, n=90) ## using the round function would lead to 91 patients

## illustrating different optimizers (see Note above for more comparison)
optDesign(emodel, probs=1, optimizer="Nelder-Mead")
optDesign(emodel, probs=1, optimizer="nlminb")
## optimizer solnp (the default) can deal with lower and upper bounds:
```

```
optDesign(emodel, probs=1, designCrit = "TD", Delta=0.5,
          optimizer="solnp", lowbnd = rep(0.2,3))
## exact design using enumeration of all possibilites
optDesign(emodel, probs=1, optimizer="exact", n = 30)
## also allows to fix minimum groupSize
optDesign(emodel, probs=1, designCrit = "TD", Delta=0.5,
          optimizer="exact", n = 30,  control = list(groupSize=5))


## optimal design at interim analysis
## assume there are already 10 patients on each dose and there are 30
## left to randomize, this calculates the optimal increment design
optDesign(emodel, 1, designCrit = "TD", Delta=0.5,
          nold = c(10, 10, 10), n=30)

## use a larger candidate model set
doses <- c(0, 10, 25, 50, 100, 150)
fmods <- Mods(linear = NULL, emax = 25, exponential = 85,
              linlog = NULL, logistic = c(50, 10.8811),
              doses = doses, addArgs=list(off=1),
              placEff=0, maxEff=0.4)
probs <- rep(1/5, 5) # assume uniform prior
desDopt <- optDesign(fmods, probs, optimizer = "nlminb")
desTD <- optDesign(fmods, probs, designCrit = "TD", Delta = 0.2,
                   optimizer = "nlminb")
desMix <- optDesign(fmods, probs, designCrit = "Dopt&TD", Delta = 0.2)
## plot design and truth
plot(desMix, fmods)

## illustrate calcCrit function
## calculate optimal design for beta model
doses <- c(0, 0.49, 25.2, 108.07, 150)
models <- Mods(betaMod = c(0.33, 2.31), doses=doses,
                addArgs=list(scal=200),
                placEff=0, maxEff=0.4)
probs <- 1
deswgts <- optDesign(models, probs, designCrit = "Dopt",
                     control=list(maxit=1000))
## now compare this design to equal allocations on
## 0, 10, 25, 50, 100, 150
doses2 <- c(0, 10, 25, 50, 100, 150)
design2 <- c(1/6, 1/6, 1/6, 1/6, 1/6, 1/6)
crit2 <- calcCrit(design2, models, probs, doses2, designCrit = "Dopt")
## ratio of determinants (returned criterion value is on log scale)
exp(deswgts$crit-crit2)

## example for calculating an optimal design for logistic regression
doses <- c(0, 0.35, 0.5, 0.65, 1)
fMod <- Mods(linear = NULL, doses=doses, placEff=-5, maxEff = 10)
## now calculate weights to use in the covariance matrix
mu <- as.numeric(getResp(fMod, doses=doses))
mu <- 1/(1+exp(-mu))
weights <- mu*(1-mu)
```

```
des <- optDesign(fMod, 1, doses, weights = weights)

## one can also specify a user defined criterion function
## here D-optimality for cubic polynomial
CubeCrit <- function(w, doses){
  X <- cbind(1, doses, doses^2, doses^3)
  CVinv <- crossprod(X*w)
  -log(det(CVinv))
}
optDesign(doses = c(0,0.05,0.2,0.6,1),
          designCrit = "userCrit", userCrit = CubeCrit,
          optimizer = "nlminb")
```

---

planMod *Evaluate performance metrics for fitting dose-response models*

Description

This function evaluates, the performance metrics for fitting dose-response models (using asymptotic approximations or simulations). Note that some metrics are available via the print method and others only via the summary method applied to planMod objects. The implemented metrics are

- Root of the mean-squared error to estimate the placebo-adjusted dose-response averaged over the used dose-levels, i.e. a rather discrete set (`dRMSE`). Available via the print method of planMod objects.
- Root of the mean-squared error to estimate the placebo-adjusted dose-response (`cRMSE`) averaged over fine (almost continuous) grid at 101 equally spaced values between placebo and the maximum dose. NOTE: Available via the summary method applied to planMod objects.
- Ratio of the placebo-adjusted mean-squared error (at the observed doses) of model-based vs ANOVA approach (`Eff-vs-ANOVA`). This can be interpreted on the sample size scale. NOTE: Available via the summary method applied to planMod objects.
- Power that the (unadjusted) one-sided '`1-alpha`' confidence interval comparing the dose with maximum effect vs placebo is larger than '`tau`'. By default '`alpha = 0.025`' and '`tau = 0`' (`Pow(maxDose)`). Available via the print method of planMod objects.
- Probability that the EDp estimate is within the true [EDpLB, EDpUB] (by default '`p=0.5`', '`pLB=0.25`' and '`pUB=0.75`'). This metric gives an idea on the ability to characterize the increasing part of the dose-response curve (`P(EDp)`). Available via the print method of planMod objects.
- Length of the quantile range for a target dose (TD or EDp). This is calculated by taking the difference of the dUB and dLB quantile of the empirical distribution of the dose estimates. (`lengthTDCI` and `lengthEDpCI`). It is NOT calculated by calculating confidence interval lengths in each simulated data-set and taking the mean. NOTE: Available via the summary method of planMod objects.

Usage

```
planMod(
  model,
  altModels,
  n,
  sigma,
  S,
  doses,
  asyApprox = TRUE,
  simulation = FALSE,
  alpha = 0.025,
  tau = 0,
  p = 0.5,
  pLB = 0.25,
  pUB = 0.75,
  nSim = 100,
  cores = 1,
  showSimProgress = TRUE,
  bnds,
  addArgs = NULL
)

## S3 method for class 'planMod'
summary(
  object,
  digits = 3,
  len = 101,
  Delta = NULL,
  p = NULL,
  dLB = 0.05,
  dUB = 0.95,
  ...
)

## S3 method for class 'planMod'
plot(
  x,
  type = c("dose-response", "ED", "TD"),
  p,
  Delta,
  placAdj = FALSE,
  xlab = "Dose",
  ylab = "",
  ...
)
```

Arguments

model	Character vector determining the dose-response model(s) to be used for fitting the data. When more than one dose-response model is provided the best fitting model is chosen using the AIC. Built-in models are "linlog", "linear", "quadratic", "emax", "exponential", "sigEmax", "betaMod" and "logistic" (see drmodels).
altModels	An object of class 'Mods', defining the true mean vectors under which operating characteristics should be calculated.
n, sigma, S	Either a vector 'n' and 'sigma' or 'S' need to be specified. When 'n' and 'sigma' are specified it is assumed computations are made for a normal homoscedastic ANOVA model with group sample sizes given by 'n' and residual standard deviation 'sigma', i.e. the covariance matrix used for the estimates is thus sigma^2*diag(1/n) and the degrees of freedom are calculated as sum(n)-nrow(contMat). When a single number is specified for 'n' it is assumed this is the sample size per group and balanced allocations are used. When 'S' is specified this will be used as covariance matrix for the estimates.
doses	Doses to use
asyApprox, simulation	Logicals determining, whether asymptotic approximations or simulations should be calculated. If multiple models are specified in 'model' asymptotic approximations are not available.
alpha, tau	Significance level for the one-sided confidence interval for model-based contrast of best dose vs placebo. Tau is the threshold to compare the confidence interval limit to. CI(MaxDCont) gives the percentage that the bound of the confidence interval was larger than tau.
p, pLB, pUB	p determines the type of EDp to estimate. pLB and pUB define the bounds for the EDp estimate. The performance metric Pr(Id-ED) gives the percentage that the estimated EDp was within the true EDpLB and EDpUB.
nSim	Number of simulations
cores	Number of cores to use for simulations. By default 1 cores is used, note that cores > 1 will have no effect Windows, as the mclapply function is used internally.
showSimProgress	In case of simulations show the progress using a progress-bar.
bnds	Bounds for non-linear parameters. This needs to be a list with list entries corresponding to the selected bounds. The names of the list entries need to correspond to the model names. The defBnds() function provides the default selection.
addArgs	See the corresponding argument in function fitMod(). This argument is directly passed to fitMod.
object, digits	object: A planMod object. digits: Digits in summary output
len	Number of equally spaced points to determine the mean-squared error on a grid (cRMSE).
Delta	Additional arguments determining what dose estimate to plot, when 'type = "ED"' or 'type = "TD"'

dLB, dUB	Which quantiles to use for calculation of `lengthTDCI` and `lengthEDpCI`. By default dLB = 0.05 and dUB = 0.95, so that this corresponds to a 90% interval.
...	Additional arguments (currently ignored)
x	An object of class planMod
type	Type of plot to produce
placAdj	When '`type = "dose-response"`', this determines whether dose-response estimates are shown on placebo-adjusted or original scale
xlab, ylab	Labels for the plot (ylab only applies for '`type = "dose-response"`')

Details

A plot method exists to summarize dose-response and dose estimations graphically.

Author(s)

Bjoern Bornkamp

References

TBD

See Also

`fitMod()`

Examples

```
## Not run:
doses <- c(0,10,25,50,100,150)
fmodels <- Mods(linear = NULL, emax = 25,
                logistic = c(50, 10.88111), exponential= 85,
                betaMod=rbind(c(0.33,2.31),c(1.39,1.39)),
                doses = doses, addArgs=list(scal = 200),
                placEff = 0, maxEff = 0.4)
sigma <- 1
n <- rep(62, 6)*2

model <- "quadratic"
pObj <- planMod(model, fmodels, n, sigma, doses=doses,
               simulation = TRUE,
               alpha = 0.025, nSim = 200,
               p = 0.5, pLB = 0.25, pUB = 0.75)
print(pObj)
## to get additional metrics (e.g. Eff-vs-ANOVA, cRMSE, lengthTDCI, ...)
summary(pObj, p = 0.5, Delta = 0.3)
plot(pObj)
plot(pObj, type = "TD", Delta=0.3)
plot(pObj, type = "ED", p = 0.5)

## End(Not run)
```

powMCT *Calculate power for multiple contrast test*

Description

Calculate power for a multiple contrast test for a set of specified alternatives.

Usage

```
powMCT(
  contMat,
  alpha = 0.025,
  altModels,
  n,
  sigma,
  S,
  placAdj = FALSE,
  alternative = c("one.sided", "two.sided"),
  df,
  critV = TRUE,
  control = mvtnorm.control()
)
```

Arguments

contMat	Contrast matrix to use. The individual contrasts should be saved in the columns of the matrix
alpha	Significance level to use
altModels	An object of class 'Mods', defining the mean vectors under which the power should be calculated
n, sigma, S	Either a vector 'n' and 'sigma' or 'S' need to be specified. When 'n' and 'sigma' are specified it is assumed computations are made for a normal homoscedastic ANOVA model with group sample sizes given by 'n' and residual standard deviation 'sigma', i.e. the covariance matrix used for the estimates is thus sigma^2*diag(1/n) and the degrees of freedom are calculated as sum(n)-nrow(contMat). When a single number is specified for 'n' it is assumed this is the sample size per group and balanced allocations are used. When 'S' is specified this will be used as covariance matrix for the estimates.
placAdj	Logical, if true, it is assumed that the standard deviation or variance matrix of the placebo-adjusted estimates are specified in 'sigma' or 'S', respectively. The contrast matrix has to be produced on placebo-adjusted scale, see optContr(), so that the coefficients are no longer contrasts (i.e. do not sum to 0).
alternative	Character determining the alternative for the multiple contrast trend test.

df	Degrees of freedom to assume in case 'S' (a general covariance matrix) is specified. When 'n' and 'sigma' are specified the ones from the corresponding ANOVA model are calculated.
critV	Critical value, if equal to 'TRUE' the critical value will be calculated. Otherwise one can directly specify the critical value here.
control	A list specifying additional control parameters for the 'qmvt' and 'pmvt' calls in the code, see also 'mvtnorm.control' for details.

Value

Numeric containing the calculated power values

Author(s)

Bjoern Bornkamp

References

Pinheiro, J. C., Bornkamp, B., and Bretz, F. (2006). Design and analysis of dose finding studies combining multiple comparisons and modeling procedures, *Journal of Biopharmaceutical Statistics*, **16**, 639–656

See Also

powN(), sampSizeMCT(), MCTtest(), optContr(), Mods()

Examples

```
## look at power under some dose-response alternatives
## first the candidate models used for the contrasts
doses <- c(0,10,25,50,100,150)
## define models to use as alternative
fmodels <- Mods(linear = NULL, emax = 25,
                logistic = c(50, 10.88111), exponential= 85,
                betaMod=rbind(c(0.33,2.31),c(1.39,1.39)),
                doses = doses, addArgs=list(scal = 200),
                placEff = 0, maxEff = 0.4)
## plot alternatives
plot(fmodels)
## power for to detect a trend
contMat <- optContr(fmodels, w = 1)
powMCT(contMat, altModels = fmodels, n = 50, alpha = 0.05, sigma = 1)

## Not run:
## power under the Dunnett test
## contrast matrix for Dunnett test with informative names
contMatD <- rbind(-1, diag(5))
rownames(contMatD) <- doses
colnames(contMatD) <- paste("D", doses[-1], sep="")
powMCT(contMatD, altModels = fmodels, n = 50, alpha = 0.05, sigma = 1)
```

```
## now investigate power of the contrasts in contMat under "general" alternatives
altFmods <- Mods(linInt = rbind(c(0, 1, 1, 1, 1),
                                c(0.5, 1, 1, 1, 0.5)),
                 doses=doses, placEff=0, maxEff=0.5)
plot(altFmods)
powMCT(contMat, altModels = altFmods, n = 50, alpha = 0.05, sigma = 1)

## now the first example but assume information only on the
## placebo-adjusted scale
## for balanced allocations and 50 patients with sigma = 1 one obtains
## the following covariance matrix
S <- 1^2/50*diag(6)
## now calculate variance of placebo adjusted estimates
CC <- cbind(-1,diag(5))
V <- (CC)%*%S%*%t(CC)
linMat <- optContr(fmodels, doses = c(10,25,50,100,150),
                   S = V, placAdj = TRUE)
powMCT(linMat, altModels = fmodels, placAdj=TRUE,
       alpha = 0.05, S = V, df=6*50-6) # match df with the df above

## End(Not run)
```

`powMCTInterim` *Calculate Conditional or Predictive Power for Multiple Contrast Test*

Description

Calculates the predictive or conditional power for a multiple contrast test based on interim data, e.g. for a futility interim analysis. This function can also be applied to longitudinal endpoints, where at the time of interim analysis incomplete data is available. For more details see the vignette on longitudinal data analysis with MCP-Mod: `vignette("Longitudinal Data MCP-Mod", package = "DoseFinding")`.

Usage

```
powMCTInterim(
  contMat,
  mu_0t,
  S_0t,
  S_01,
  alpha = 0.025,
  type = c("predictive", "conditional"),
  mu_assumed = NULL,
  alternative = c("one.sided", "two.sided"),
  control = mvtnorm.control()
)
```

Arguments

contMat	Contrast matrix to use. The individual contrasts should be saved in the columns of the matrix
mu_0t	The first stage estimates
S_0t	The covariance matrix for the first stage estimates
S_01	The covariance matrix anticipated for the estimates at study end
alpha	Significance level to use
type	Whether predictive power (for a flat prior) or conditional power should be calculated. For conditional power mu_assumed needs to be specified.
mu_assumed	Mean vector to assume for the second stage (only used when type is 'conditional'). If NULL (default), the first stage estimates mu_0t are used.
alternative	Character determining the alternative for the multiple contrast trend test.
control	A list specifying additional control parameters for the 'pmvnorm' calls in the code, see also 'mvtnorm.control' for details.

Value

Numeric containing the calculated power values

References

Bornkamp, B., Zhou, J., Xi, D. and Cao W. (2025). Futility analyses for the MCP-Mod methodology based on longitudinal models, *arXiv:2406.19965*

See Also

powMCT() MCTtest(), optContr()

Examples

```
# Setup the scenario.
doses <- c(0, 0.5, 1, 2, 4, 8)
mods <- Mods(
  emax = c(0.5, 1, 2, 4),
  sigEmax = rbind(c(0.5, 3), c(1, 3), c(2, 3), c(4, 3)),
  quadratic = -0.1,
  doses = doses
)
w <- c(1, 0.5, 0.5, 0.5, 1, 1)
contMat <- optContr(models = mods, w = w)$contMat
sigma <- 0.3
n_final <- round(531 * w / sum(w))
n <- floor(n_final / 2)
S_0t <- diag(sigma^2 / n)
S_01 <- diag(sigma^2 / n_final)
## assumed interim estimate
mu_0t <- 0.05 * doses / (doses + 1) + rnorm(6, 0, 0.382 / sqrt(n))
## assumed mu (needed for conditional power)
```

```
mu_assumed <- 0.135 * doses / (doses + 1)

# Calculate predictive and conditional power.
powMCTInterim(
  contMat = contMat, S_0t = S_0t, S_01 = S_01, mu_0t = mu_0t,
  type = "predictive"
)
powMCTInterim(
  contMat = contMat, S_0t = S_0t, S_01 = S_01, mu_0t = mu_0t,
  type = "conditional", mu_assumed = mu_assumed
)
powMCTInterim(
  contMat = contMat, S_0t = S_0t, S_01 = S_01, mu_0t = mu_0t,
  type = "predictive", alternative = "two.sided"
)
powMCTInterim(
  contMat = contMat, S_0t = S_0t, S_01 = S_01, mu_0t = mu_0t,
  type = "predictive", control = mvtnorm.control(maxpts = 1e5)
)
```

`sampSize` *Sample size calculations*

Description

The 'sampSize' function implements a bisection search algorithm for sample size calculation. The user can hand over a general target function (via 'targFunc') that is then iterated so that a certain 'target' is achieved. The 'sampSizeMCT' is a convenience wrapper of 'sampSize' for multiple contrast tests using the power as target function.

Usage

```
sampSize(
  upperN,
  lowerN = floor(upperN/2),
  targFunc,
  target,
  tol = 0.001,
  alRatio,
  Ntype = c("arm", "total"),
  verbose = FALSE
)

sampSizeMCT(
  upperN,
  lowerN = floor(upperN/2),
  ...,
  power,
```

```
  sumFct = mean,
  tol = 0.001,
  alRatio,
  Ntype = c("arm", "total"),
  verbose = FALSE
)

targN(
  upperN,
  lowerN,
  step,
  targFunc,
  alRatio,
  Ntype = c("arm", "total"),
  sumFct = c("min", "mean", "max")
)

powN(
  upperN,
  lowerN,
  step,
  ...,
  alRatio,
  Ntype = c("arm", "total"),
  sumFct = c("min", "mean", "max")
)

## S3 method for class 'targN'
plot(x, superpose = TRUE, line.at = NULL, xlab = NULL, ylab = NULL, ...)
```

Arguments

upperN, lowerN	Upper and lower bound for the target sample size. lowerN defaults to floor(upperN/2).
targFunc, target	The target function needs to take as an input the vector of sample sizes in the different dose groups. For 'sampSize' it needs to return a univariate number. For function 'targN' it should return a numerical vector. Example: 'targFunc' could be a function that calculates the power of a test, and 'target' the desired target power value. For function 'sampSize' the bisection search iterates the sample size so that a specific target value is achieved (the implicit assumption is that targFunc is monotonically increasing in the sample size). Function 'targN' simply calculates 'targFunc' for a given set of sample sizes.
tol	A positive numeric value specifying the tolerance level for the bisection search algorithm. Bisection is stopped if the 'targFunc' value is within 'tol' of 'target'.

alRatio	Vector describing the relative patient allocations to the dose groups up to proportionality, e.g. 'rep(1, length(doses))' corresponds to balanced allocations.
Ntype	One of "arm" or "total". Determines, whether the sample size in the smallest arm or the total sample size is iterated in bisection search algorithm.
verbose	Logical value indicating if a trace of the iteration progress of the bisection search algorithm should be displayed.
...	Arguments directly passed to the powMCT() function in the 'sampSizeMCT' and 'powN' function.
power, sumFct	power is a numeric defining the desired summary power to achieve (in 'sampSizeMCT').
step	Only needed for functions 'targN' and 'powN'. Stepsize for the sample size at which the target function is calculated. The steps are calculated via seq(lowerN,upperN,by=step).
x, superpose, line.at, xlab, ylab	arguments for the plot method of 'targN' and 'powN', additional arguments are passed down to the low-level lattice plotting routines.

Details

The 'targN' functions calculates a general target function for different given sample sizes. The 'powN' function is a convenience wrapper of 'targN' for multiple contrast tests using the power as target function.

Author(s)

Jose Pinheiro, Bjoern Bornkamp

References

Pinheiro, J. C., Bornkamp, B., and Bretz, F. (2006). Design and analysis of dose finding studies combining multiple comparisons and modeling procedures, *Journal of Biopharmaceutical Statistics*, **16**, 639–656

Pinheiro, J.C., Bornkamp, B. (2017) Designing Phase II Dose-Finding Studies: Sample Size, Doses and Dose Allocation Weights, in O'Quigley, J., Iasonos, A. and Bornkamp, B. (eds) Handbook of methods for designing, monitoring, and analyzing dose-finding trials, CRC press

See Also

powMCT()

Examples

```
## sampSize examples

## first define the target function
## first calculate the power to detect all of the models in the candidate set
fmodels <- Mods(linear = NULL, emax = c(25),
                logistic = c(50, 10.88111), exponential=c(85),
                betaMod=matrix(c(0.33,2.31,1.39,1.39), byrow=TRUE, nrow=2),
                doses = c(0,10,25,50,100,150), placEff=0, maxEff=0.4,
```

```
                addArgs = list(scal=200))
## contrast matrix to use
contMat <- optContr(fmodels, w=1)
## this function calculates the power under each model and then returns
## the average power under all models
tFunc <- function(n){
  powVals <- powMCT(contMat, altModels=fmodels, n=n, sigma = 1,
                    alpha=0.05)
  mean(powVals)
}

## assume we want to achieve 80% average power over the selected shapes
## and want to use a balanced allocations
## Not run:
sSize <- sampSize(upperN = 80, targFunc = tFunc, target=0.8,
                  alRatio = rep(1,6), verbose = TRUE)
sSize


## Now the same using the convenience sampSizeMCT function
sampSizeMCT(upperN=80, contMat = contMat, sigma = 1, altModels=fmodels,
            power = 0.8, alRatio = rep(1, 6), alpha = 0.05)
## Alternatively one can also specify an S matrix
## covariance matrix in one observation (6 total observation result in a
## variance of 1 in each group)
S <- 6*diag(6)
## this uses df = Inf, hence a slightly smaller sample size results
sampSizeMCT(upperN=500, contMat = contMat, S=S, altModels=fmodels,
            power = 0.8, alRatio = rep(1, 6), alpha = 0.05, Ntype = "total")


## targN examples
## first calculate the power to detect all of the models in the candidate set
fmodels <- Mods(linear = NULL, emax = c(25),
                logistic = c(50, 10.88111), exponential=c(85),
                betaMod=matrix(c(0.33,2.31,1.39,1.39), byrow=TRUE, nrow=2),
                doses = c(0,10,25,50,100,150), placEff=0, maxEff=0.4,
                addArgs = list(scal=200))
## corresponding contrast matrix
contMat <- optContr(fmodels, w=1)
## define target function
tFunc <- function(n){
  powMCT(contMat, altModels=fmodels, n=n, sigma = 1, alpha=0.05)
}
powVsN <- targN(upperN = 100, lowerN = 10, step = 10, tFunc,
                alRatio = rep(1, 6))
plot(powVsN)

## the same can be achieved using the convenience powN function
## without the need to specify a target function
powN(upperN = 100, lowerN=10, step = 10, contMat = contMat,
     sigma = 1, altModels = fmodels, alpha = 0.05, alRatio = rep(1, 6))
```

```
## End(Not run)
```

`Target doses` *Calculate dose estimates for a fitted dose-response model (via* `fitMod()`, `bFitMod()`*) or* `maFitMod()`*) or a* `Mods()` *object*

Description

The TD (target dose) is defined as the dose that achieves a target effect of Delta over placebo (if there are multiple such doses, the smallest is chosen):

$$TD_{\Delta} = \min\{x | f(x) > f(0) + \Delta\}$$

If a decreasing trend is beneficial the definition of the TD is

$$TD_{\Delta} = \min\{x | f(x) < f(0) - \Delta\}$$

When Δ is the clinical relevance threshold, then the TD is similar to the usual definition of the minimum effective dose (MED).

The ED (effective dose) is defined as the dose that achieves a certain percentage p of the full effect size (within the observed dose-range!) over placebo (if there are multiple such doses, the smallest is chosen).

$$ED_p = \min\{x | f(x) > f(0) + p(f(dmax) - f(0))$$

Note that this definition of the EDp is different from traditional definition based on the Emax model, where the EDp is defined relative to the *asymptotic* maximum effect (rather than the maximum effect in the observed dose-range).

ED or TD calculation for bootstrap model averaging (maFit) objects is based on first calculating the pointwise median dose-response curve estimate. Then calculating the dose estimate based on this curve.

Usage

```
TD(
  object,
  Delta,
  TDtype = c("continuous", "discrete"),
  direction = c("increasing", "decreasing"),
  doses = NULL
)

ED(
  object,
  p,
  EDtype = c("continuous", "discrete"),
  direction = c("increasing", "decreasing"),
  doses = NULL
)
```

Arguments

`object`	An object of class c(Mods, fullMod), DRMod, bFitMod or maFit
`Delta`, `p`	Delta: The target effect size use for the target dose (TD) (Delta should be > 0). p: The percentage of the dose to use for the effective dose.
`TDtype`, `EDtype`	character that determines, whether the dose should be treated as a continuous variable when calculating the TD/ED or whether the TD/ED should be calculated based on a grid of doses specified in '`doses`'
`direction`	Direction to be used in defining the TD. This depends on whether an increasing or decreasing of the response variable is beneficial. In case of ED calculation only needed for maFit objects.
`doses`	Dose levels to be used if '`TDtype`' or '`EDtype`' are equal to '`"discrete"`'. Needs to include placebo, and may not exceed the dose range of the model(s) provided in '`object`'.

Value

Returns the dose estimate

Author(s)

Bjoern Bornkamp

See Also

`Mods()`, `drmodels()`, `fitMod()`, `bFitMod()`

Examples

```
## example for creating a "full-model" candidate set placebo response
## and maxEff already fixed in Mods call
doses <- c(0, 10, 25, 50, 100, 150)
fmodels <- Mods(linear = NULL, emax = 25,
                logistic = c(50, 10.88111), exponential = 85,
                betaMod = rbind(c(0.33, 2.31), c(1.39, 1.39)),
                linInt = rbind(c(0, 1, 1, 1, 1),
                               c(0, 0, 1, 1, 0.8)),
                doses=doses, placEff = 0, maxEff = 0.4,
                addArgs=list(scal=200))
## calculate doses giving an improvement of 0.3 over placebo
TD(fmodels, Delta=0.3)
## discrete version
TD(fmodels, Delta=0.3, TDtype = "discrete", doses=doses)
## doses giving 50% of the maximum effect
ED(fmodels, p=0.5)
ED(fmodels, p=0.5, EDtype = "discrete", doses=doses)
plot(fmodels, plotTD = TRUE, Delta = 0.3)
```

Index