

Package ‘DoseFinding’

April 4, 2012

Type Package

Title Planning and Analyzing Dose Finding experiments

Version 0.6-2

Date 2012-04-04

Author Bjoern Bornkamp, Jose Pinheiro, Frank Bretz

Depends lattice, mvtnorm, R (>= 2.4.1)

Suggests numDeriv, Rsolnp, partitions, multcomp

Maintainer Bjoern Bornkamp <bornkamp@statistik.tu-dortmund.de>

Description The DoseFinding package provides functions for the design and analysis of dose-finding experiments (for example pharmaceutical Phase II clinical trials). It provides functions for: multiple contrast tests, fitting non-linear dose-response models, calculating optimal designs and an implementation of the MCPMod methodology.

License GPL-3

LazyLoad yes

Repository CRAN

Date/Publication 2012-04-04 18:55:24

R topics documented:

DoseFinding-package	2
AIC.DRMod	4
biom	5
bootMCPMod	6
calcBayesEst	7
calcCrit	10
calcOptDesign	12

critVal	16
DR-Models	18
DRMod and gDRMod methods	21
ED.DRMod	24
fit.control	25
fitDRModel	26
fullMod	29
genDFdata	31
getBnds	32
getGrad	33
getInit	34
getPars	35
getUpdDesign	36
gFitDRModel	38
gMCPtest	40
guesst	43
IBScovars	45
LP	46
MCPMod	48
MCPtest	55
MED.DRMod	58
migraine	60
modelMeans	60
mvtnorm.control	61
planMM	62
plot.fullMod	64
plot.LP	64
plot.MCPMod	65
plot.planMM	66
plot.powerMM	67
plotModels	69
powCalc	70
powerMM	72
powerScenario	74
predict.MCPMod	75
rndDesign	77
sampSize	77

Index**81**

Description

The DoseFinding package provides functions for the design and analysis of dose-finding experiments (for example pharmaceutical Phase II clinical trials). It provides functions for: multiple contrast tests (`MCPtest`), fitting non-linear dose-response models (`fitDRModel`), calculating optimal designs (`calcOptDesign`) and an implementation of the MCPMod methodology (`MCPMod`). For non-normal endpoints see `gMCPtest` and `gFitDRModel`.

Details

Package:	DoseFinding
Type:	Package
Version:	0.6-2
Date:	2012-04-04
License:	GPL-3
LazyLoad:	yes

The main functions are: `MCPtest`, implementing multiple contrast tests, `fitDRModel` that fits non-linear dose-response models, `calcOptDesign` that calculate optimal designs and `MCPMod`, which implements the MCPMod methodology. `gMCPtest` and `gFitDRModel` are generalizations of `MCPtest` and `fitDRModel` for non-normal data.

Author(s)

Bjoern Bornkamp, Jose Pinheiro, Frank Bretz

Maintainer: Bjoern Bornkamp <bornkamp@statistik.tu-dortmund.de>

References

- Bornkamp, B., Bretz, F., Dette, H. and Pinheiro, J. C. (2011). Response-Adaptive Dose-Finding under model uncertainty, *Annals of Applied Statistics*, **5**, 1611–1631
- 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
- Bretz, F., Pinheiro, J. C., and Branson, M. (2005), Combining multiple comparisons and modeling techniques in dose-response studies, *Biometrics*, **61**, 738–748
- Dette, H., Bretz, F., Pepelyshev, A. and Pinheiro, J. C. (2008). Optimal Designs for Dose Finding Studies, *Journal of the American Statistical Association*, **103**, 1225–1237
- 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
- Seber, G.A.F. and Wild, C.J. (2003). Nonlinear Regression, Wiley

Examples

```
data(IBScovars)
```

```

# perform (model based) multiple contrast test
modlist <- list(linear = NULL, emax = 0.2, quadratic = -0.17)
# plot model shapes (need to specify base and maxEff additionally)
plotModels(modlist, c(0,4), base = 0, maxEff = 1)
fittest <- MCPtest(resp ~ dose, IBScovars, modlist, addCovars = ~ gender)

# fit non-linear dose-response model
fitemax <- fitDRModel(resp ~ dose, IBScovars, "emax")
# display fitted dose-effect curve
plot(fitemax)
# estimate minimum effective dose
MED(fitemax, clinRel = 0.25)

# MCP-Mod is a combination of (model based) multiple contrast tests
# and nonlinear regression
models <- list(linear = NULL, emax = 0.2, quadratic = -0.17)
dfe <- MCPMod(resp ~ dose, IBScovars, models, addCovars = ~gender,
              pVal = TRUE, selModel = "maxT",
              doseEst = "MED2", clinRel = 0.25)
# detailed information is available via summary
summary(dfe)
# plots data with selected model function
plot(dfe)

# Calculate optimal designs for MED estimation
doses <- c(0, 10, 25, 50, 100, 150)
mods <- list(linear = NULL, emax = 25, exponential = 85,
            linlog = NULL, logistic = c(50, 10.8811))
fMod <- fullMod(mods, doses, base=0, maxEff=0.4, off=1)
weights <- rep(1/5, 5)
desMED <- calcOptDesign(fMod, weights, doses, clinRel=0.2, scal=200,
                       off=1, method = "nlminb")

```

AIC.DRMod

Calculate AIC, BIC or log-likelihood for a DRMod object

Description

Calculate AIC, BIC or log-likelihood for a DRMod object

Usage

```

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

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

```

Arguments

object	A DRMod object
...	Additional parameters
k	Penalty used for the parameters, $k=2$ corresponds to the AIC, while $k=\log(\text{number of observations})$ corresponds to BIC.

See Also[fitDRModel](#)

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

bootMCPMod

*Evaluate precision of dose estimate by nonparametric bootstrapping***Description**

Applies nonparametric bootstrapping to evaluate the precision of the dose-estimate, the dose-response estimate and the model selection of a MCPMod object. Bootstrapping is done stratified, so that we resample within each dose group.

Usage

```
bootMCPMod(x, nSim = 1000, fitControl = list(), start = NULL,
           seed = 100, calcDR = FALSE, ...)
```

Arguments

x	A MCPMod object, with a calculated critical value (see <code>critV</code> argument in MCPMod function)
nSim	Number of bootstrap replications
fitControl	List of control parameters for nonlinear fitting (see <code>fit.control</code> for details), passed down to MCPMod function
start	List of starting values for nls, passed down to MCPMod function during bootstrapping
seed	Random seed (to be able to reproduce results)
calcDR	Logical values indicating, whether bootstrap dose-response estimates should be calculated and saved in the bootMCPMod object.
...	Additional arguments passed to the <code>predict.MCPMod</code> function, in case <code>calcDR = TRUE</code> , see <code>predict.MCPMod</code> for details.

Value

An object of class `bootMCPMod`, ie. a list with entries `doseEst`, `model` and (if `calcDR` is `TRUE`) `doseResp`, containing the bootstrap replications. In case of model averaging `doseEstMods` contains the bootstrap replications of the dose estimates of the different significant models. Additionally the original fitted MCPMod object is included.

Author(s)

Bjoern Bornkamp

See Also

[MCPMod](#)

Examples

```

## Not run:
data(biom)
models <- list(linear = NULL, linlog = NULL, emax = 0.2,
               exponential = c(0.279,0.15), quadratic = c(-0.854,-1))
dfe <- MCPMod(resp ~ dose, biom, models, alpha = 0.05, doseEstPar = 0.05,
              selModel = "maxT", doseEst = "MED2old", clinRel = 0.4, off = 1,
              critV = TRUE, optimizer = "bndnls")
btMED <- bootMCPMod(dfe, nSim = 1000)
btMED
## more information on quantiles of dose-estimate
quantile(btMED$doseEst, c(0.05,0.25,0.5,0.75,0.95), na.rm=TRUE)
## plot density estimate of dose-estimate
dens <- density(btMED$doseEst, from = 0, to = 1, na.rm=TRUE)
plot(dens, main = "Bootstrap MED Replications")
## summarize selected models
table(btMED$model)/1000

## more time consuming
## now use model averaging and estimate ED50
data(IBScovars)
models <- list(emax = 0.2, quadratic = -0.2, linlog = NULL)
dfe <- MCPMod(resp ~ dose, IBScovars, models, addCovars = ~gender,
              alpha = 0.05, critV = TRUE, selModel = "aveAIC", doseEst = "ED",
              clinRel = 0.25, off = 1, optimizer = "bndnls")
## now also investigate uncertainty in DR estimate
## predict at sq
sq <- seq(0, 4, length = 101)
## Note: type = "EffectCurve", doseSeq = sq are passed to predict.MCPMod
btED <- bootMCPMod(dfe, nSim = 1000, calcDR = TRUE,
                  type = "EffectCurve", doseSeq = sq)
btED
## display information on ED50 estimate
dens <- density(btED$doseEst, from = 0, to = 4, na.rm=TRUE)
plot(dens, main = "Bootstrap MED Replications")
## plot uncertainty quantiles for dose-response curve
res <- apply(btED$doseResp, 2, quantile,
             prob = c(0.1,0.25, 0.5,0.75, 0.9), na.rm=T)
plot(sq, res[3,], ylim = c(0, 0.6), type = "l",
     xlab = "Dose", ylab = "Effect Curve")
polygon(c(sq,rev(sq)), c(res[1,], rev(res[5,])), col = "lightgrey")
polygon(c(sq,rev(sq)), c(res[2,], rev(res[4,])), col = "grey")
lines(sq, res[3,], type = "l")

## End(Not run)

```

Description

Calculates posterior means (or a variant of the posterior mode, see Details below) and the posterior model probabilities for a set of candidate models. The data are assumed to be homoscedastic normally distributed and no additional covariates are allowed. The methodology is described in Bornkamp et al. (2011), see below for the reference.

Usage

```
calcBayesEst(data, models, prior, bnds = getBnds(mD = max(data$dose)),
             weights = NULL, numPar = c(100, 1597), meanInd = TRUE,
             clinRel, scal = NULL, off = NULL)
```

Arguments

- | | |
|--------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| data | Data frame containing the dose and the response data. The code assumes the columns to be named "dose" and "resp". |
| models | models list (as in <code>MCPMod</code> function). Note that for this function only the models "linear", "linlog", "emax", "exponential", "logistic" and "betaMod" are allowed. |
| prior | <p>List with the following entries</p> <p>a, d, m (vector of length 2), V (2x2 matrix), S</p> <p>components of prior list:</p> <p>a,d,m,V determine the normal inverse gamma prior for baseline, maximum effect and sigma2 see O'Hagan and Forster (p., 305, 2006) or Bornkamp et al. (2011) for the parameterization used here (the parameter d is called nu in the Bornkamp et al. (2011) paper).</p> <p>For the non-linear parameters (those in the standardized model function), beta distributions are used, the bounds of the beta distribution are determined from the bnds argument, while mode of the beta distribution is obtained from the guesstimates from the model list.</p> <p>The parameter S determines the variability in the prior.</p> <p>a, d - determine the prior mode of the prior for sigma2 the mode of the prior for sigma2 is $a/(d+2)$, the prior mean is $a/(d-2)$ ($d>2$) the prior variance for sigma2 is infinite for $d\leq 4$. Note that d is called ν in the Bornkamp et al. (2011) paper.</p> <p>m - vector of length 2: prior means for placebo and maximum effect of the dose-response models (note that this information is transformed to priors for the two linear model parameters (e.g. E0, E1 for the exponential model) when the parameters do not directly have the interpretation of placebo and maximum effect)</p> <p>V - Covariance matrix of prior (for placebo and maximum effect)</p> <p>S - Sum of beta distribution parameters (should be > 2), the mode of the beta distributions is automatically selected as the specified guesstimates in the models list.</p> |

bnds	List of parameter bounds for models (see getBnds for details)
weights	prior model probabilities: in same order as models in list. By default equal prior model probabilities are assumed.
numPar	vector with two entries (number of glp points for 1d and 2d integration)
meanInd	indicator, whether posterior means (meanInd = TRUE) or the posterior mode (meanInd = FALSE) should be calculates. See Details for more information
clinRel	Clinical relevance
scal	scale parameter for beta model
off	Offset parameter for linlog model

Details

When meanInd = FALSE, the code finds the mode of the marginal distribution of the non-linear parameters. Then it calculates the posterior mode of the linear parameters given the mode of the non-linear parameters.

Value

Returns a list with the Bayesian parameter estimates, each list entry corresponds to one model. The list has an attribute weights, which contains the calculated model probabilities. In addition there is an attribute existsMED containing a logical vector indicating, whether the MED estimate exists for the given dose-response parameter estimates.

Author(s)

Bjoern Bornkamp

References

Bornkamp, B., Bretz, F., Dette, H. and Pinheiro, J. C. (2011). Response-Adaptive Dose-Finding under model uncertainty, *Annals of Applied Statistics*, 5, 1611-1631

O'Hagan, A. and Forster, J. (2006) *Kendall's Advanced Theory of Statistics 2B: Bayesian Inference*, 2nd edition, Arnold, London

Examples

```
## example 1
doses <- c(0, 62.5, 125, 250, 500)
mods <- list(emax = 25, linear = NULL, logistic = c(50, 50), betaMod = c(1, 1))
clinRel <- 200
d <- 4
prior <- list(a=350^2*(d+2), d=d, m=c(60,280), V=matrix(c(100000,0,0,100000),2,2), S=10)
dats <- genDFdata("emax", c(e0 = 60, eMax = 294, ed50 = 25), doses, rep(46, 5), 350)
## calculate posterior means
calcBayesEst(dats, mods, prior, weights = rep(1/4, 4), clinRel = clinRel, scal=600)
## calculate posterior mode
```

```

calcBayesEst(dats, mods, prior, weights = rep(1/4, 4), clinRel =
              clinRel, scal=600, meanInd = FALSE)

## example 2 (investigate under different prior scenarios)
s2 <- 1
models <- list(emax = c(7.5), logistic = matrix(c(30,60,4,11), nrow=2),
              betaMod=c(1,1))
data <- genDFdata("emax", c(e0 = 0, eMax = 1, ed50 = 7.5), c(0, 10, 37.5, 75),
                 n=(280/4), sigma=sqrt(s2))
priorProp <- list(S=3, a = s2*6, d = 4, m = c(0, 1), V = c(10,0,0,10))
priorAlt1 <- list(S=3, a = s2*6, d = 4, m = c(-0.5, 1.2), V = c(10,0,0,10))
priorAlt2 <- list(S=3, a = s2*6, d = 4, m = c(0, 1), V = c(1000,0,0,1000))
priorAlt3 <- list(S=3, a = s2*60, d = 40, m = c(0, 1), V = c(40,0,0,40))
priorAlt4 <- list(S=20, a = s2*60, d = 40, m = c(0, 1), V = c(40,0,0,40))

calcBayesEst(data, models, prior = priorProp, clinRel = 0.5, scal = 100,
              meanInd = FALSE)
calcBayesEst(data, models, prior = priorAlt1, clinRel = 0.5, scal = 100,
              meanInd = FALSE)
calcBayesEst(data, models, prior = priorAlt2, clinRel = 0.5, scal = 100,
              meanInd = FALSE)
calcBayesEst(data, models, prior = priorAlt3, clinRel = 0.5, scal = 100,
              meanInd = FALSE)
calcBayesEst(data, models, prior = priorAlt4, clinRel = 0.5, scal = 100,
              meanInd = FALSE)

```

calcCrit

Calculate design criterion for a specified design.

Description

This function calculates the design criterion for a specified design under a specified candidate set of models.

For type = MED this is the weighted average of log-variances for the MED under the different models, note that these are only the variances up to a constant which is irrelevant in the optimization. For type = Dopt this is the weighted average of log-determinants of the inverse of the Fisher information. For type = MED&Dopt this an equally weighted mixture of the criterions type = Dopt and type = MED.

Usage

```

calcCrit(design, fullModels, weights, doses, clinRel,
         nold = rep(0, length(doses)), n2 = NULL,
         scal=1.2*max(doses), off=0.1*max(doses),
         type = c("MED", "Dopt", "MED&Dopt"), standDopt = FALSE)

```

Arguments

design	Numeric vector of allocation weights or matrix for the different doses. The rows of the matrices need to sum to 1. Alternatively also an object of class "design" can be used. Note that there should be at least as many design points available as there are parameters in the dose-response models selected in fullModels (otherwise the code returns an NA).
fullModels	List containing all model parameters for the models (can for example be a fullMod object, see the fullMod function for details). When an MED optimal design should be calculated the MED needs to exist for all models specified in fullModels. If a D-optimal design should be calculated, you need at least as many doses as there are parameters in the specified models.
weights	Vector of model probabilities for the models specified in fullModels.
doses	Doses available
clinRel	Clinical relevance
nold	Vector of sample sizes already allocated to the different doses.
n2	Sample size for next cohort.
scal	Scal parameter for beta model
off	Offset parameter for linlog model
type	Determines which type of design criterion to calculate. "MED&Dopt" uses both optimality criteria with equal weight.
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 = "MED&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).

Value

Value of design criterion

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 Statistical Association*, **103**, 1225–1237

See Also

[calcOptDesign](#)

Examples

```

## Example (i) Dette et al. (2008), p. 1228, Table 2, line 5
## calculate optimal design for Emax model
mods <- list(emax = 25)
doses <- c(0, 18.75, 150)
fMod <- fullMod(mods, doses, base=0, maxEff=0.4)
fMod$emax[2] <- 0.6666667
weights <- 1 # just one model
des <- calcOptDesign(fMod, weights, doses, clinRel=0.2)

## 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, fMod, weights, doses2, clinRel=0.2)
## ratio of asymptotic variances (constant cancels)
exp(des$crit-crit2)
## slightly modified design
design3 <- c(0.3, 0.3, 0.2, 0, 0, 0.2)
crit3 <- calcCrit(design3, fMod, weights, doses2, clinRel=0.2)
## ratio of asymptotic variances
exp(des$crit-crit3)

## Example (ii) Dette et al. (2008), p. 1230, Table 5, line 5
## calculate optimal design for beta model
fmods <- list(betaMod = c(0.33, 2.31))
doses <- c(0, 0.49, 25.2, 108.07, 150)
fMod <- fullMod(fmods, doses, base=0, maxEff=0.4, scal=200)
weights <- 1
deswgt <- calcOptDesign(fMod, weights, doses, clinRel=0.1,
  scal=200, 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, fMod, weights, doses2, clinRel=0.1, scal=200)
## ratio of asymptotic variances
exp(deswgt$crit-crit2)

## example with matrix
designs <- rbind(c(0.25,0.5,0.25), c(0.5,0.25,0.25), c(0.34,0.33,0.33))
mods <- list(emax = 25)
doses <- c(0, 18.75, 150)
fMod <- fullMod(mods, doses, base=0, maxEff=1)
weights <- 1
calcCrit(designs, fMod, weights, doses, clinRel = 0.2)

```

Description

Given a set of models (with full parameter values and model probabilities) this function calculates the MED optimal design (see Dette, Bretz, Pepelyshev and Pinheiro (2008) for details) or the D-optimal design, or a mixture of these two criteria.

Usage

```
calcOptDesign(fullModels, weights, doses, clinRel = NULL,
              nold = rep(0, length(doses)), n2 = NULL,
              control = list(), scal=1.2*max(doses),
              off=0.1*max(doses),
              type = c("MED", "Dopt", "MED&Dopt", "userCrit"),
              method = c("Nelder-Mead", "nlminb", "solnp", "exact"),
              lowbnd = rep(0, length(doses)),
              uppbnd = rep(1, length(doses)),
              standDopt = FALSE, userCrit = NULL, ...)
```

Arguments

fullModels	List containing all model parameters for the models (can for example be a fullMod object, see the fullMod function for details). When an MED optimal design should be calculated the MED needs to exist for all models specified in fullModels. If a D-optimal design should be calculated, you need at least as many doses as there are parameters in the specified models.
weights	Vector of model probabilities for the models specified in fullModels.
doses	Doses available
clinRel	Clinical relevance needed for calculating "MED" and "MED&Dopt" type designs.
nold	Vector of sample sizes already allocated to the different doses.
n2	Sample size for next cohort (mandatory for method = "exact" and when any entry of nold is larger than 0).
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 blockSize in case the patients are allocated a minimum group size is requiredy.
scal	Scal parameter for beta model
off	Offset parameter for linlog model
type	Determines which type of design to calculate. "MED&Dopt" uses both optimality criteria with equal weight.
method	Algorithm used for calculating the optimal design. Options "Nelder-Mead" and "nlminb" use the optim and nlminb function and use trigonometric functions 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_2 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 blockSize, this can be adjusted via the control argument. $n_2/\text{blockSize}$ and $\text{length}(\text{doses})$ 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 and more reliable than "Nelder-Mead"). For a larger number of doses option "solnp" is the best option ("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" optimizer.
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 = "MED&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).
userCrit	User defined design criterion, should be a function that given a vector of allocation weights and the doses returns the criterion function. 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 ...
...	Additional arguments for userCrit.

Details

The difference to the methodology proposed in Dette et al. (2008) is the fact that the doses are treated as fixed (and specified via doses): The design is only optimized with respect to the design weights (ie the allocation weights for the different doses).

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 Statistical Association*, **103**, 1225–1237

See Also

[calcCrit](#)

Examples

```
## first example calculate MED optimal design for one Emax model
## fullModel should be a list containing all model parameters
fMod <- list(emax = c(0, 2/3, 25))
doses <- c(0, 18.75, 150)
weights <- 1 # just one model
## by default calculates MED optimal design
des1 <- calcOptDesign(fMod, weights, doses, clinRel=0.2)
des2 <- calcOptDesign(fMod, weights, doses, type = "Dopt")
des3 <- calcOptDesign(fMod, weights, doses, clinRel=0.2, type = "MED&Dopt")

## illustrating the different optimizers
des1 <- calcOptDesign(fMod, weights, doses, clinRel=0.2, method="Nelder-Mead")
des2 <- calcOptDesign(fMod, weights, doses, clinRel=0.2, method="nlminb")
des3 <- calcOptDesign(fMod, weights, doses, clinRel=0.2, method="solnp")
## assume additional constraints (only available for method = solnp)
des4 <- calcOptDesign(fMod, weights, doses, clinRel=0.2, lowbnd = rep(0.2,3),
  uppbnd = rep(0.45, 3), method="solnp")
## assume only 20 patients are to be allocated (exact calculation possible)
des5 <- calcOptDesign(fMod, weights, doses, n2 = 20, clinRel=0.2,
  method="exact")
## assume the minimum block-size is 5
des6 <- calcOptDesign(fMod, weights, doses, n2 = 20, clinRel=0.2,
  method="exact", control = list(blockSize = 5))

## larger candidate model set
doses <- c(0, 10, 25, 50, 100, 150)
mods <- list(linear = NULL, emax = 25, exponential = 85,
  linlog = NULL, logistic = c(50, 10.8811))
## now use fullMod function to produce the fullModel list
## from the 'usual' models list
fMod <- fullMod(mods, doses, base=0, maxEff=0.4, off=1)
weights <- rep(1/5, 5)
desMED <- calcOptDesign(fMod, weights, doses, clinRel=0.2, scal=200,
  off=1, method = "nlminb")
desDopt <- calcOptDesign(fMod, weights, doses, scal=200, off=1,
  type = "Dopt")
desMix <- calcOptDesign(fMod, weights, doses, clinRel=0.2, scal=200,
  off=1, type = "MED&Dopt")
## allocated 100 persons according to desMix design
rndDesign(desMix, 100)

#####
#### using already allocated patients
```

```

mods <- list(betaMod = c(0.33, 2.31))
doses <- c(0, 0.49, 25.2, 108.07, 150)
fMod <- fullMod(mods, doses, base=0, maxEff=0.4, scal=200)
weights <- 1
## no previously allocated patients
des <- calcOptDesign(fMod, weights, doses, clinRel=0.1, scal=200,
                    control=list(maxit=1000))

## now use previously allocated patients
nold <- c(45, 50, 0, 0, 0)
des2 <- calcOptDesign(fMod, weights, doses, clinRel=0.1, n2=30, scal=200,
                    control=list(maxit=1000), nold=nold)
## the calculated overall design is the same as the 1-step design
(30*des2$design+nold)/(30+sum(nold))
des$design

## same with exact optimizer
des <- calcOptDesign(fMod, weights, doses, clinRel=0.1, scal=200,
                    control=list(blockSize = 10), n2=120, method = "exact")
nold <- c(10, 10, 0, 0, 0)
des2 <- calcOptDesign(fMod, weights, doses, clinRel=0.1, n2=100, scal=200,
                    control=list(blockSize = 10), nold=nold, method = "exact")
(des2$design*100+nold)/(100+sum(nold))

#####
#### user defined criterion function (D-optimality for cubic polynomial)
CubeCrit <- function(w, doses){
  X <- cbind(1, doses, doses^2, doses^3)
  CVinv <- crossprod(X*w)
  -log(det(CVinv))
}
calcOptDesign(doses = c(0,0.05,0.2,0.6,1),
              type = "userCrit", userCrit = CubeCrit,
              method = "nlnminb")

```

critVal

Calculate critical value for multiple contrast test

Description

This function calculates the critical value for a multiple contrast test via numerical integration (using the methods implemented in the `mvtnorm` package).

Usage

```

critVal(cMat, n, alpha = 0.025,
        alternative = c("one.sided", "two.sided"),
        control = mvtnorm.control(), corMat = NULL, nDF = NULL)

```

Arguments

cMat	A matrix with the contrasts in the columns
n	A vector giving the sample size per group. If only one number is specified it is assumed that the sample sizes are balanced.
alpha	Level of significance (defaults to 0.025)
alternative	Character determining the alternative for the multiple contrast trend test.
control	A list of options for the pmvt and qmvt functions as produced by mvtnorm.control
corMat	Optional: correlation matrix of contrasts can be specified
nDF	Degrees of freedom of the underlying t-statistics, if NULL the degrees of freedom are calculated from n under the assumption of a parallel group design.

Value

Critical value

References

- Bretz, F., Pinheiro, J. and Branson, M. (2005), Combining Multiple Comparisons and Modeling Techniques in Dose-Response Studies, *Biometrics*, **61**, 738–748
- Hothorn, T., Bretz, F., and Genz, A. (2001), On multivariate t and Gauss probabilities in R, *R News*, **1**, 27–29

See Also

[planMM](#), [mvtnorm.control](#), [MCPtest](#)

Examples

```
## Calculation of critical value for Dunnett test
## Set up contrast matrix (3 active doses)
CM <- rbind(-1,diag(3))
## 30 patients per group, one-sided alpha 0.05
critVal(CM, n=30, alpha = 0.05)
## Example from R News 1(2) p. 28, 29
CM <- c(1, 1, 1, 0, 0, -1, 0, 0, 1, 0, 0, -1, 0, 0,
        1, 0, 0, 0, -1, -1, 0, 0, -1, 0, 0)
CM <- t(matrix(CM, ncol = 5))
critVal(CM, n=c(26, 24, 20, 33, 32), alpha = 0.05,
        alternative = "two.sided")
```

Description

Dose-response model functions and gradients.

Below are the definitions of the model functions:

E_{max} model

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

Sigmoid E_{max} 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 scale parameter that is not estimated in the code.

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 an offset parameter not estimated in the code.

Logistic Model

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

Quadratic Model

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

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, ...)

```

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
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
delta1	delta1 parameter for beta model
delta2	delta2 parameter for beta model
b1	first parameter of quadratic model
b2	second parameter of quadratic model (controls, whether model is convex or concave)
off	Offset value to avoid problems with dose=0 (treated as a fixed value in the code)
scal	Scale parameter (treated as a fixed value in the code)
...	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).

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 ed_{50} value. The sigmoid Emax model describes monotonic, sigmoid dose-response relationships.

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 via the argument $scal$ (a reasonable value is often $1.2 * (\text{maximum dose})$).

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.

The **logistic model** is intended to capture general monotone, sigmoid dose-response relationships.

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

[fitDRModel](#), [gFitDRModel](#)

Examples

```
## some emax example shapes
emaxModList <- list(emax = c(0.02,0.1,0.5,1))
plotModels(emaxModList, c(0,1), base = 0, maxEff = 1)
## example for gradient
emaxGrad(dose = (0:4)/4, eMax = 1, ed50 = 0.5)

## some sigmoid emax example shapes
sigEmaxModList <- list(sigEmax = rbind(c(0.5,1), c(0.5,8), c(0.2,1),
                                     c(0.2,8)))
plotModels(sigEmaxModList, c(0,1), base = 0, maxEff = 1)
sigEmaxGrad(dose = (0:4)/4, eMax = 1, ed50 = 0.5, h = 8)

## some exponential example shapes
expoModList <- list(exponential = c(0.1,0.25,0.5,2))
plotModels(expoModList, c(0,1), base = 0, maxEff = 1)
exponentialGrad(dose = (0:4)/4, e1 = 1, delta = 2)
```

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

## some logistic model example shapes
logistModList <- list(logistic = rbind(c(0.5,0.05), c(0.5,0.15),
                                       c(0.2,0.05), c(0.2,0.15)))
plotModels(logistModList, c(0,1), base = 0, maxEff = 1)
logisticGrad(dose = (0:4)/4, eMax = 1, ed50 = 0.5, delta = 0.05)
```

DRMod and gDRMod methods

Methods for DRMod and gDRMod objects

Description

coef.(g)DRMod - Extract coefficients object, if wanted, the function can separate the coefficients into coefficients of the dose-response model and the additional linear coefficients.

predict.(g)DRMod - Predict a DRMod or gDRMod object

intervals.(g)DRMod, vcov.(g)DRMod - Calculate (asymptotic) covariance matrix, confidence intervals for parameters in a DRMod or gDRMod object, based on standard asymptotic theory for nonlinear models, see Seber and Wild (2003). **NOTE:** When bounds are used for the non-linear parameters (i.e., when the optimizer was "bndnls" or `gFitDRModel` was used) the provided approximation is only valid if the true parameter is in the interior of the provided bounds. When the parameter estimate is on the boundary the provided approximation is often not practically useful.

plot.(g)DRMod - Plot a DRMod or gDRMod object. `plot.(g)DRMod` displays either the dose-response effect curve or the full dose-response curve for a particular selection of the additional covariates of a DRMod object. The plot function (invisibly) returns all information necessary for producing the display (when it is desired to have a more customized display). See the Value section for more details.

Usage

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

## S3 method for class 'DRMod'
predict(object, type = c("fullModel", "EffectCurve"),
        newdata = NULL, doseSeq = NULL, se.fit = FALSE, lenSeq = 101,
        data = getData(object), uGrad = NULL,...)
## S3 method for class 'gDRMod'
predict(object, type = c("fullModel", "EffectCurve"),
```

```

doseSeq = NULL, se.fit = FALSE, lenSeq = 101, ...)

## S3 method for class 'DRMod'
vcov(object, data = getData(object), uGrad = NULL, ...)
## S3 method for class 'gDRMod'
vcov(object, ...)

## S3 method for class 'DRMod'
intervals(object, level = 0.95, data = getData(object),
           uGrad = NULL, ...)
## S3 method for class 'gDRMod'
intervals(object, level = 0.95, ...)

## S3 method for class 'DRMod'
plot(x, type = c("EffectCurve", "DRCurve"),
     addCovarVals = NULL, CI = FALSE, level = 0.95,
     plotData = c("means", "complData", "none"),
     display = TRUE, lenDose = 201,
     data = getData(x), uGrad, ...)
## S3 method for class 'gDRMod'
plot(x, type = c("DRCurve", "EffectCurve"), CI = FALSE,
     level = 0.95, plotData = c("means", "meansCI", "none"),
     display = TRUE, lenDose = 201, ...)

```

Arguments

object	A DRMod or gDRMod object
x	A DRMod or gDRMod object
sep	Logical determining whether all coefficients should be returned in one numeric or separated in a list.
data	Data frame containing the variables on which the DRMod object was fitted (this needs to be handed over if DRMod object does not contain the data on which it was fitted).
level	Confidence level to be used
newdata	Data frame containing values where to predict when using type="fullModel", if missing use the data, where the model was fitted on.
doseSeq	Numeric specifying doses where to predict in case of type="EffectCurve", if missing use lenSeq equally spaced values between smallest and largest dose in the data.
se.fit	Logical determining, whether standard deviations for the predicted mean should be calculated.
lenSeq	If doseSeq is not specified a equally spaced grid of "lenSeq" values between placebo and the maximal dose in the study is used.
type	Character specifying the type of plot/prediction. If type = "EffectCurve", the effect curve is plotted/predicted. If type = "DRCurve" the dose-response curve is plotted/predicted for a particular value of the additional covariates

addCovarVals	If type = "DRCurve" (and addCovars != 1) one needs to specify the values of the other covariates (in a data.frame with one row!)
CI	Logical indicating whether confidence curve for the mean should be plotted
plotData	If addCovars = ~1 and type = "DRCurve" this determines, whether means or complete data or no data should be plotted within the plot of the dose-response curve
display	Determines whether the plot should be displayed or only data necessary for the plot should be produced.
lenDose	If doseSeq is not specified a equally spaced grid of lenDose values between placebo and the maximal dose in the study is used.
uGrad	Function to return the gradient of a user defined model, see Examples of the fitDRModel function.
...	Additional arguments, for plot.DRMod these are passed to the call of plot

Value

For predict(g)DRMod: Either a numeric consisting out of the predictions (when se.fit = FALSE), or a list with elements, fit, se.fit, residual.scale and df, containing predicted mean, associated (asymptotic) standard deviation, the residual sum of squares and the degrees of freedom of the fit.

Author(s)

Bjoern Bornkamp

References

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

See Also

[fitDRModel](#), [gFitDRModel](#)

Examples

```
## examples for predict method
data(IBScovars)
fm <- fitDRModel(resp ~ dose, IBScovars, "emax", addCovars = ~ gender)

# predict only effect relative to placebo at 0, 1, 2, 3, 4
predict(fm, type = "EffectCurve", doseSeq = 0:4)
# add standard deviation for effect
predict(fm, type = "EffectCurve", doseSeq = 0:4, se.fit = TRUE)

# predict full model, specify where to predict via newdata
preddat <- data.frame(dose = 0:4, gender = as.factor(rep(1, 5)))
predict(fm, type = "fullModel", newdata = preddat)
# add standard deviations for predictions
predict(fm, type = "fullModel", newdata = preddat, se.fit = TRUE)
```

```
## examples for plot method
set.seed(123)
x <- rep(c(0,0.05,0.2,0.6,1), c(30,30,30,20,10))
y <- sigEmax(x, 0, 2, 0.2, 2)
age <- rnorm(120, 50, 10)
center <- as.factor(sample(c("C1","C2","C3"), 120, replace = TRUE))
y <- y + -0.03*age + 0.5*(center=="C1") + rnorm(120, 0, 1)
datfram <- data.frame(x, y, age, center)
fit <- fitDRModel(y ~ x, datfram, "sigEmax", addCovars = ~age+center)
# just plot effect curve
plot(fit, type = "EffectCurve")
# plot full dose-response curve, need values for covariates
op <- par()$mfrow
par(mfrow = c(1,2))
plot(fit, type = "DRCurve",
     addCovarVals = data.frame(age = 20, center = as.factor("C1")))
plot(fit, type = "DRCurve",
     addCovarVals = data.frame(age = 50, center = as.factor("C3")))
par(mfrow = op)
# plot full dose-response curve for covariates from datfram
plot(fit, type = "DRCurve",addCovarVals = datfram[1,3:4])
```

ED.DRMod

*Calculate ED_p estimator for a DRMod object***Description**

This function calculates the effective dose of a specified level for a DRMod object. Here the effective dose of level p is the smallest dose that achieves p percent of the maximum effect observed *within* the dose range (and not with respect to the asymptotic maximum effect).

Usage

```
## S3 method for class 'DRMod'
ED(object, p, doseSeq = NULL, lenDose = 101,
    direction = c("increasing", "decreasing"),
    data = getData(object),...)
```

Arguments

object	An object of class DRMod
p	Numeric specifying the level used for the ED _p estimate.
doseSeq	Sequence of values allowed for the dose estimator. Should include 0 (ie placebo) as a first value.
lenDose	If doseSeq is not specified an equally spaced grid of lenDose values between placebo and the maximal dose in the study is used.
direction	Character specifying the direction of the dose effect (ie whether the response should increase or decrease with increasing dose).

data Data frame containing the variables on which the DRMod object was fitted
 ... Additional arguments

Value

ED dose estimate

Author(s)

Bjoern Bornkamp

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

See Also

[MED.DRMod](#), [fitDRModel](#)

Examples

```
data(biom)
fit <- fitDRModel(resp ~ dose, biom, "logistic")
ED(fit, p = c(0.25,0.5,0.95))
```

<code>fit.control</code>	<i>Set control parameters for non-linear model fitting</i>
--------------------------	------------------------------------------------------------

Description

Function to set the control parameters for the nonlinear optimizers used to find the maximum likelihood estimates.

Usage

```
fit.control(nlscontrol = list(), nlminbcontrol = list(),
            optimizetol = .Machine$double.eps^0.5,
            gridSize = list(dim1 = 30, dim2 = 144))
```

Arguments

`nlscontrol` List of control parameters for the nls function, see [nls.control](#) for details.
`nlminbcontrol` List of control parameters for the nlminb function see [nlminb](#) and its control argument for details.
`optimizetol` tol parameter for the [optimize](#) function.
`gridSize` List with two components named `dim1` and `dim2`, determining the size of the grid for the brute force optimizer used for finding a starting value (for models with one nonlinear parameter (`dim1`) and two nonlinear parameters (`dim2`)).

Value

List containing the different control parameters

Author(s)

Bjoern Bornkamp

See Also

[nls.control](#), [nlminb](#), [optimize](#)

fitDRModel

Fit a non-linear regression model with linear covariates.

Description

Fits a non-linear regression model (possibly with linear covariates) by minimizing the least squares criterion. The implemented algorithms are based on the idea performing the non-linear optimization only for the non-linear parameters and solving linear least squares problems in each iteration (see Golub and Pereyra, 2003 for a review). The optimizer "nls" uses the [nls](#) function with `plinear` option. Option "bndnls" imposes bounds on the non-linear parameters (a sufficient condition for existence of the least squares estimate, see for example Seber and Wild (2003)), and is described in more details below, see the argument 'optimizer'.

Usage

```
fitDRModel(formula, data, model = NULL, addCovars = ~1, na.action = na.fail,
  optimizer = c("nls", "nls&bndnls", "bndnls"), bnds = NULL, start = NULL,
  nlscontrol = nls.control(), gridSize =
  list(dim1 = 30, dim2 = 144), nlminbcontrol = list(),
  optimizetol = .Machine$double.eps^0.5, off = NULL,
  scal = NULL, keep.data = TRUE, uModPars = NULL, addArgs = NULL)
```

Arguments

formula	A formula object specifying the response and the dose variable (in the form response ~ dose). Additional covariates need to be specified via the addCovars argument, see below for details.
data	Dose response data frame containing the variables needed for fitting the non-linear model.
model	The non-linear model to be used for fitting the data. Built-in models are "linlog", "linear", "quadratic", "emax", "exponential", "sigEmax", "betaMod" and "logistic" (see DR-Models). User defined models can also be specified. The name specified in the model argument needs to be equal to the name of the model function, see the examples for details.
addCovars	Formula specifying additional (linear) covariates

na.action	A function which indicates what should happen when the data contain NAs.
optimizer	A character specifying the optimizer to be used. Option "nls" uses the <code>nls</code> function with "plinear" option. Option "bndnls" uses a non-linear least squares algorithm with simple box constraints on the (non-linear) parameters (the algorithm works by evaluating the objective function first on a grid and then starting the <code>nlminb</code> function from the best function value found). The advantage of imposing bounds is that it guarantees the existence of an optimum of the objective function. Option "nls&bndnls" first uses <code>nls</code> and when non-convergence of "nls" occurs it uses the non-linear least squares algorithm with bounds "bndnls".
bnds	Bounds for non-linear parameters, needed when <code>optimizer="bndnls"</code> . When the dose-response model has only one non-linear parameter (for example <code>Emax</code> or exponential model), <code>bnds</code> needs to be a vector containing upper and lower bound. For models with two non-linear parameters <code>bnds</code> needs to be a matrix containing the bounds in the rows, see the Description section of getBnds for details on the formatting of the bounds for the individual models.
start	List containing starting values for the "nls" fitting algorithm (for "bndnls" no starting values are needed). The names of the list elements need to be equal to the names of the model functions. The names of the starting vector should equal the corresponding names for the model parameters. For built-in models starting values need to be provided only for the non-linear parameters. In case of a user model (not built in) starting values for the all parameters need to be supplied.
nlscontrol	List of parameters to be used in the calls to the <code>nls</code> function. See also <code>nls.control</code> function.
gridSize	List with two components (<code>dim1</code> , <code>dim2</code>) giving the size of the grid used for "bndnls" optimization for 1-dim and 2-dim case (in 2-dim case the smallest generalized lattice point set larger than or equal to the number given is used)
nlminbcontrol	List of parameters to be used in the calls to the <code>nlminb</code> function. See the control argument of nlminb for details.
optimizetol	tol parameter of the one dimensional optimize function. See optimize for details.
off	Fixed offset parameter needed when the linear in log model is used. See also documentation of the linear in log model: "linlog". A reasonable default is to use $(\text{maximum dose}) \cdot 0.1$ for <code>off</code> .
scal	Fixed scale parameter needed when the beta model is used. See also documentation of the beta model: betaMod . A reasonable default is to use $(\text{maximum dose}) \cdot 1.2$ for <code>scal</code> .
keep.data	Logical: should the data argument be saved as part of the model object?
uModPars	Optional character vector with names/expressions of user-defined model parameters (<code>names(start)</code> used by default).
addArgs	Optional character vector with names of additional arguments (variables) to user-defined model.

Value

An object of class `DRMod`. A list with entries

`coef` A vector containing the coefficients of the fitted model. See also [coef.DRMod](#)

RSS2 The residual sum of squares of the fitted model
df The degrees of freedom of the fit
addCovars The formula for the additional covariates used for fitting

Additionally the object contains a variety of additional information as attributes.

References

Golub, G. and Pereyra, V. (2003). Seperable nonlinear least squares: the variable projection method and its applications, *Inverse Problems*, **19**, R1–R26

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

See Also

[linear](#), [linlog](#), [sigEmax](#), [emax](#), [quadratic](#), [logistic](#), [betaMod](#), [exponential](#), [predict.DRMod](#), [plot.DRMod](#), [AIC.DRMod](#), [MED.DRMod](#), [ED.DRMod](#)

Examples

```
## Example (i): Overview of capabilities
data(biom)
fitemax <- fitDRModel(resp ~ dose, biom, "emax")

## a lot of things can be done with a DRMod object
## basic information
fitemax
## a bit more information
summary(fitemax)
## predicting
predict(fitemax, newdata=data.frame(dose = c(0,0.5,1)))
## predictions with standard error
predict(fitemax, newdata=data.frame(dose = c(0,0.5,1)), se.fit = TRUE)
## plotting
plot(fitemax, type = "DRCurve")
plot(fitemax, type = "EffectCurve") # difference from placebo
## extracting coefficients
coef(fitemax)
## (asymptotic) covariance matrix of estimates
vcov(fitemax)
## confidence intervals for estimates
intervals(fitemax, level = 0.95)
## fitted log-likelihood
logLik(fitemax)
## extracting AIC (or BIC)
AIC(fitemax)
## calculate the minimum effective dose
MED(fitemax, type = "MED2", clinRel = 0.2, gamma = c(0.05, 0.1))
## calculate effective dose
ED(fitemax, p = 0.5)

## quite a few models are built in
```

```

fitsigEmax <- fitDRModel(resp ~ dose, biom, "sigEmax")
plot(fitsigEmax, type = "DRCurve")

fitlog <- fitDRModel(resp ~ dose, biom, "logistic")
plot(fitlog, type = "DRCurve")

fitQuad <- fitDRModel(resp ~ dose, biom, "quadratic")
plot(fitQuad, type = "DRCurve")

## need additional scal parameter for beta model
fitBeta <- fitDRModel(resp ~ dose, biom, "betaMod", scal = 1.2)
plot(fitBeta, type = "DRCurve")

## additionally there is the exponential, the linear
## and linear in log model...

## Example (ii): Fitting with linear covariates
## specify those via addCovars
data(IBScovars)
fitemax <- fitDRModel(resp ~ dose, IBScovars, "emax", addCovars = ~gender)
plot(fitemax, type = "EffectCurve")
plot(fitemax, type = "DRCurve", addCovarVals=data.frame(gender = as.factor(1)))
plot(fitemax, type = "DRCurve", addCovarVals=data.frame(gender = as.factor(2)))

## for logistic model use bndnls (plain nls does not converge)
fitlog <- fitDRModel(resp ~ dose, IBScovars, "logistic", addCovars = ~gender,
                    optimizer = "bndnls", bnds = rbind(c(1,4), c(0,1)))
## optimum lies on the boundary of bound
plot(fitlog, type = "EffectCurve")

## Example (iii): fitting a user model
## is a cubic polynomial model
userModel <- function(dose, par0, par1, par2, par3){
  par0+par1*dose+par2*dose^2+par3*dose^3
}

userModelGrad <- function(dose, par0, par1, par2, par3){
  cbind(1, dose, dose^2, dose^3)
}

fit <- fitDRModel(resp ~ dose, biom, "userModel", addCovars = ~1,
                 start = c(par0=0.2, par1=3, par2=-5, par3=3),
                 uModPars = NULL, addArgs = NULL)
plot(fit, uGrad = userModelGrad)

```

Description

Calculates location and scale parameters for all models in the candidate set using the maximum approach from Pinheiro et al. (2006). This is done by repeatedly calling the `getPars` function.

Usage

```
fullMod(models, doses, base, maxEff, off = 0.1*max(doses), scal = 1.2 * max(doses))
```

Arguments

<code>models</code>	A list specifying the candidate models. The names of the list entries should be equal to the names of the model functions. The list entries should be equal to the guesstimates
<code>doses</code>	Dose levels to be administered
<code>base, maxEff</code>	Placebo effect and maximum change from placebo (within considered dose range) to be used for calculating the location and scale parameters of the model
<code>off</code>	Offset parameter for the linear in log model
<code>scal</code>	Scale parameter for the beta model

Value

Returns an object of class `fullMod`, containing all parameter values for the models in a list.

References

Bornkamp B., Pinheiro J. C., 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., 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

[getPars](#), [sampSize](#), [powerMM](#), [plotModels](#), [LP](#)

Examples

```
doses <- c(0, 10, 25, 50, 100, 150)
models <- list(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))
fMod <- fullMod(models, doses, base = 0, maxEff = 0.4, scal = 200)
plot(fMod) # automatically calls the plotModels function
```

genDFdata	<i>Simulate dose-response data</i>
-----------	------------------------------------

Description

The function simulates normally distributed dose-response data, according to a prespecified dose-response model (or mean vector) and a given standard deviation.

Usage

```
genDFdata(model, argsMod, doses, n, sigma, mu = NULL, offset = NULL)
```

Arguments

model	Character string giving the name of a model function. The first argument of the model function should be the dose variable. See DR-Models for details on the built in dose-response models
argsMod	A vector with the arguments for the model function, see DR-Models for details on the built in dose-response models.
doses	Dose levels to be used.
n	Group sample sizes.
sigma	Standard deviation.
mu	If model is not specified mu is used to determine the mean vector of the observations.
offset	Offset vector. If not equal to NULL, offset is added to the mean of the simulated normal distribution

Value

A data frame with two columns called dose and resp, corresponding to the dose and simulated response values.

See Also

[DR-Models](#)

Examples

```
## Not run:
# use emax model
genDFdata("emax", c(e0 = 0.2, eMax = 1, ed50 = 0.05), c(0,0.05,0.2,0.6,1), 20, 1)
# use fixed mean vector
genDFdata(mu = 1:5, doses = 0:4, n = c(20, 20, 10, 5, 1), sigma = 0.2)
# use covariates via offset
X <- rbind(runif(100), c(rep(1,20), rep(0,80)))
beta <- c(1,0.3)
```

```
offset <- crossprod(X,beta)
genDFdata("emax", c(e0 = 0.2, eMax = 1, ed50 = 0.05),
          c(0,0.05,0.2,0.6,1), 20, 1, offset = offset)

## End(Not run)
```

getBnds *Calculates default bounds for non-linear parameters*

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

```
getBnds(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, 30), 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 parameters for the models

Value

List containing bounds for the model parameters.

Author(s)

Bjoern Bornkamp

See Also

[fitDRModel](#)

Examples

```
getBnds(mD = 1)
getBnds(mD = 200)
```

getGrad	<i>Calculate the gradient for the non-linear part of a DRMod object</i>
---------	-------------------------------------------------------------------------

Description

Calculate the gradient for the non-linear part of a DRMod object. The gradient is used to calculate variance covariance matrix, confidence intervals and confidence intervals for predictions, based on a linearization of the non-linear model see Seber and Wild (2003) for details. This function is also used to calculate the design criterion for the [calcOptDesign](#) function (which depends on the asymptotic covariance matrix).

Usage

```
getGrad(object, dose, uGrad = NULL)
```

Arguments

object	A DRMod object
dose	Dose where to evaluate the gradient
uGrad	Function to return the gradient of a user defined model, see Examples of the fitDRModel function.

Value

Matrix containing the gradient

Author(s)

Bjoern Bornkamp

References

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

See Also

[vcov.DRMod](#), [predict.DRMod](#), [intervals.DRMod](#), [fitDRModel](#)

getInit	<i>Starting values for non-linear parameters.</i>
---------	---------------------------------------------------

Description

Calculates starting values for non-linear parameters in non-linear models. Note that starting values are only needed if optimizer = "nls" in the MCPMod function.

Usage

```
getInit(data,  
        model = c("emax", "exponential", "logistic", "betaMod", "sigEmax"),  
        scal, weights, addCovars)
```

Arguments

data	Data frame containing dose response data set. The dose column should be called "dose", the response column "resp", all other variables possibly referenced in addCovars should also be present.
model	Character specifying the model.
scal	Scal parameter for betaMod model.
weights	Numeric specifying weights for fitting.
addCovars	Formula specifying the additional covariates.

Value

Starting value

References

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

[fitDRModel](#)

`getPars`*Calculate location and scale parameters*

Description

Given the baseline, the maximum effect and the standardized model parameters this function calculates the location and scale parameters in the model function using the maximum approach, see Pinheiro et al. (2006) for the basic idea.

Usage

```
getPars(model, doses, initEstim, base, maxEff,  
        off = 0.1 * max(doses), scal = 1.2 * max(doses))
```

Arguments

<code>model</code>	A character string with the model name. Built-in models have their full parameterization derived internally. For user-defined models, it is assumed that a function named as "Par" appended to end of model name exists (e.g., for model = "cubic", it is assumed that there is function "cubicPar" that calculates the necessary parameters; this function is assumed to have arguments "doses", "initEstim", "base", and "maxEff", in that order (see below for an example).
<code>doses</code>	Doses to be used in design
<code>initEstim</code>	Vector of guesstimates
<code>base</code>	Expected placebo effect
<code>maxEff</code>	Expected maximum change from placebo (in considered dose range)
<code>off</code>	Offset parameter for the linear in log model (default 1).
<code>scal</code>	Scale parameter for the beta model (default: 20 perc. larger than maximum dose).

Value

Vector containing all model parameters.

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

[fullMod](#)

Examples

```
doses <- c(0, 10, 25, 50, 100, 150)
getPars("emax", doses, 25, 0, 0.4)
getPars("logistic", doses, c(50, 10.88111), 0, 0.4) # compare JBS 16, p.650
getPars("betaMod", doses, initEstim = c(0.33, 2.31), base = 0,
        maxEff = 0.4)
#example for user model
userMod <- function(dose, e0, eMax, ed50, h){
  e0 + eMax * ( dose^h / (ed50^h + dose^h) )
}
# function to return location and scale parameters
userModPar <- function(dose, initEstim, base, maxEff){
  # function to get linear parameters
  # ed50 parameter assumed to be first in initEstim
  ed50 <- initEstim[1]
  h <- initEstim[2]
  dmax <- max(dose)
  emax <- maxEff*(ed50^h+dmax^h)/dmax^h
  c(base, emax, initEstim)
}
getPars("userMod", doses, initEstim = c(50,2), base = 0, maxEff = 1)
```

getUpdDesign

Calculate Bayes estimates and optimal design for next cohort

Description

Wrapper function that calculates Bayesian dose-response model parameter estimates, model probabilities and optimal design for next cohort of patients. Internally two main function are called (i) [calcBayesEst](#) to calculate the Bayes estimates and (ii) [calcOptDesign](#) to calculate the optimal design.

Usage

```
getUpdDesign(data = NULL, doses, n2, clinRel = NULL, models, prior, scal = NULL,
             meanInd = TRUE, sWeights, sDoses, method = c("Nelder-Mead", "nlminb", "solnp", "exact"),
             type = c("MED", "Dopt", "MED&Dopt"), control = list())
```

Arguments

data	Data frame containing the dose and the response data. The code assumes the columns to be named "dose" and "resp". If equal to NULL, the design specified via sWeights is used.
doses	Numeric vector given the doses available for the adaption
n2	Numeric specifying the sample size of the next cohort
clinRel	Clinical relevance threshold

models	A list specifying the candidate models. The names of the list entries should be equal to the names of the model functions. See help page for MCPMod function.
prior	list specifying the parameters of the prior (see calcBayesEst for details)
scal	scale parameter for beta dose-response model
meanInd	indicator, whether posterior means (if meanInd = TRUE) or the posterior mode should be used for updating (see calcBayesEst for details).
sWeights	Allocation weights for the start design (used when data=NULL)
sDoses	Doses to be used in start design (these might be different from the doses available for adaption)
method	Algorithm used for calculating the optimal design (see also calcOptDesign for details).
type	Determines which type of design to calculate. "MED&Dopt" uses both optimality criteria with equal weight.
control	control list passed down to calcOptDesign

Details

The function uses a Bayesian updating rule to update parameter estimates and model probabilities. Then it calculates the optimal design for the next cohort of patients.

When there is a dose-response model, for which no dose in the dose-range achieves clinical relevance, this model is removed from the model set (for calculation of the optimal design), and the model probabilities are recalculated to sum to 1. When there is no model with a clinically relevant dose, a balanced allocation to all doses available is used.

Author(s)

Bjoern Bornkamp

References

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

Bornkamp, B., Bretz, F., Dette, H. and Pinheiro, J. C. (2011). Response-Adaptive Dose-Finding under model uncertainty, *Annals of Applied Statistics*, 5, 1611-1631

Examples

```
## examples
models <- list(emax = 0.5, linear = NULL, logistic = c(4.5, 0.8), betaMod = c(1.5,1.5))
prior <- list(S=3, a = 5*6, d = 4, m = c(0, 1.65), V = c(4.5,0,0,4.5))
dfdata <- NULL
doses <- c(0,2,4,6,8)
startDes <- getUpdDesign(dfdata, doses, n2 = 100, clinRel = 0.4, models, prior, scal=10,
                        meanInd = FALSE, sWeights = rep(1/4,4), sDoses = c(0,2,4,8))
dfdata <- genDFdata("emax", c(e0 = 0.2, eMax = 1, ed50 = 0.5),
                  startDes[,1], startDes[,2], sqrt(4.5))
getUpdDesign(dfdata, doses, n2 = 225, clinRel = 1.3, models, prior,
```

```

        scal=10, meanInd = FALSE, method = "solnp",
        sWeights = rep(1/4,4), sDoses = c(0,2,4,8), type="MED&Dopt")
## now with 9 doses (0:8)
getUpdDesign(dfdata, 0:8, n2 = 225, clinRel = 1.3, models, prior,
        scal=10, method = "solnp", meanInd = FALSE,
        sWeights = rep(1/4,4), sDoses = c(0,2,4,8), type="MED&Dopt")

```

gFitDRModel	<i>Generalized fitting of dose-response models to raw dose-response estimates</i>
-------------	-----------------------------------------------------------------------------------

Description

The function fits a dose-response model to raw dose-response estimates (obtained for example from an ANOVA type model), by optimizing the generalized least squares objective function:

$$(f(\text{dose}, \theta) - \text{drEst})' \text{vCov}^{-1} (f(\text{dose}, \theta) - \text{drEst})$$

where drEst are the response estimates at the dose-levels dose and vCov the associated covariance matrix. This approach is rather general: drEst and vCov can be the output from another model fitting procedure like generalized linear models, survival models or even mixed models. The underlying optimizer is similar to the bndnls option in the fitDRModel function: First a grid-search optimization is used for the non-linear parameters of the dose-response model, then the nlminb optimizer starts from the best value found on the grid. gFitDRModel will always return an estimate (the best of the two optimizations, usually the estimate from nlminb unless it failed). Inference capabilities (confidence intervals, etc) for gFitDRModel fits, will be added in later releases of the DoseFinding package.

Usage

```

gFitDRModel(dose, drEst, vCov, model = NULL, intercept = TRUE,
            bnds = NULL, start = NULL, gridSize = list(dim1 = 30,
            dim2 = 144), nlminbcontrol = list(), off = NULL,
            scal = NULL)

```

Arguments

dose	Numeric specifying the dose variable.
drEst	Numeric specifying the response estimate corresponding to the doses in dose
vCov	Covariance matrix associated with the dose-response estimate specified via drEst
model	Dose-response model to fit, possible models are "linlog", "linear", "quadratic", "emax", "exponential", "sigEmax", "betaMod" and "logistic", see DR-Models .
intercept	Whether or not to include an intercept in the model. Models without intercept are of interest for example for fitting placebo-corrected estimates (note that the linear in log and the logistic model cannot be fitted without intercept)
bnds	Bounds for the nonlinear parameters, need to be finite. Per default the bounds are determined with getBnds function is used (max(dose) as input parameter).

start	Optional starting values for the nonlinear parameters of the dose-response model. If equal to NULL a grid search is first performed to find a suitable starting value
gridSize	List with two components (dim1, dim2) giving the size of the grid used for "bndnls" optimization for 1-dim and 2-dim case (in 2-dim case the smallest generalized lattice point set larger than or equal to the number given is used)
nlminbcontrol	List of parameters to be used in the calls to the nlminb function. See the control argument of nlminb for details
off	Fixed offset parameter needed when the linear in log model is used. See also documentation of the linear in log model: "linlog". A reasonable default is to use (maximum dose)*0.1 for off
scal	Fixed scale parameter needed when the beta model is used. See also documentation of the beta model: betaMod A reasonable default is to use (maximum dose)*1.2 for scal.

Value

An object of class gDRMod. A list with entries

coef	A vector containing the coefficients of the fitted model.
gRSS2	The residual sum of squares of the fitted model under the generalized least squares criterion
data	The dose response data set to which the model has been fitted

Additionally the object contains a variety of additional information as attributes.

Author(s)

Bjoern Bornkamp

See Also

[fitDRModel](#), [linear](#), [linlog](#), [sigEmax](#), [emax](#), [quadratic](#), [logistic](#), [betaMod](#), [exponential](#), [coef.gDRMod](#), [predict.gDRMod](#), [vcov.gDRMod](#), [plot.gDRMod](#)

Examples

```
## apply to normal data (fitDRModel leads to the same results)
data(biom)
## produce first stage fit (using dose as factor)
anMod <- lm(resp~factor(dose)-1, data=biom)
drFit <- coef(anMod)
vCov <- vcov(anMod)
dose <- sort(unique(biom$dose))
## now fit an emax model to these estimates
gfit <- gFitDRModel(dose, drFit, vCov, model = "emax",
                   bnds = c(0.01, 2))
## a lot of things can be done with gDRMod objects
print(gfit)
summary(gfit)
```

```

coef(gfit)
## see ?vcov.gDRMod for the caveats of the asymptotic
## inference statements
vcov(gfit)
predict(gfit, se.fit = TRUE)
plot(gfit, CI=TRUE, plotData = "meansCI")

## example for binary data (migraine)
data(migraine)
PFrate <- migraine$painfree/migraine$ntrt
doseVec <- migraine$dose
doseVecFac <- as.factor(migraine$dose)
## fit logistic regression with dose as factor
logfit <- glm(PFrate~doseVecFac-1, family = binomial,
              weights = migraine$ntrt)
drEst <- coef(logfit)
vCov <- vcov(logfit)
## now fit an Emax model (on logit scale)
gfit2 <- gFitDRModel(doseVec, drEst, vCov, model = "emax", bnds = c(0,300))
## model fit on logit scale
plot(gfit2)

## now fit placebo adjusted data
data(biom)
## produce first stage fit (using dose as factor)
anMod <- lm(resp~factor(dose), data=biom)
## estimates (contrasts to placebo)
drFitC <- coef(anMod)[-1]
vCovC <- vcov(anMod)[-1,-1]
dose <- sort(unique(biom$dose))[-1]
## now fit an emax model to these estimates
gfit3 <- gFitDRModel(dose, drFitC, vCovC, model = "emax",
                    bnds = c(0.01, 2), intercept = FALSE)
plot(gfit3, plotData = "meansCI")

```

gMCPtest

Generalized multiple contrast tests

Description

The function performs a multiple contrast test based on dose-response estimates specified via `drEst` and `vCov`. It is assumed that the estimates in `drEst` approximately follow a multivariate normal distribution for calculation of p-values. The contrasts are chosen corresponding to models specified via the `models` argument.

Usage

```

gMCPtest(dose, drEst, vCov, models, alpha = 0.025, contMat = NULL,
         critV = NULL, pVal = TRUE, alternative = c("one.sided", "two.sided"),
         direction = c("increasing", "decreasing"),
         mvtcontrol = mvtnorm.control(), std = TRUE, off, scal)

```

Arguments

dose	Numeric specifying the dose variable.
drEst	Numeric specifying the response estimate corresponding to the doses in dose
vCov	Covariance matrix associated with the dose-response estimate specified via drEst
models	A candidate models list. A list specifying the model shapes to be included in the contrast matrix. The names of the list entries should be equal to the names of the model functions. The list entries should be equal to prior estimates for standardized model parameters. See the the MCPMod function for details on how to specify candidate model shapes.
alpha	Significance level for the multiple contrast test
contMat	Optional matrix containing the optimal contrasts in the columns. If specified the code does not calculate the optimal contrasts.
critV	Critical value, if NULL, no critical value will be calculated, and the test decision will be based on the p-values. If critV = TRUE the critical value will be calculated (the test decision will be based on the critical value). If critV is equal to a numerical value it will be assumed that the critical value is pre-specified and it will not be calculated by the code (the test decision will then also be based on the critical value).
pVal	Optional logical determining whether p-values should be calculated, defaults to TRUE. If the critical value is supplied, p-values will not be calculated.
alternative	Character determining the alternative for the multiple contrast trend test.
direction	Character determining the trend direction of the data, which one wants to investigate (e.g., if one wants to investigate whether the response gets larger with increasing dose direction should be equal to "increasing"). When the contrast matrix is handed over via the 'contMat' argument the direction argument is ignored (the direction is implicit in the contrast matrix).
mvtcontrol	A list specifying additional control parameters for the qmvt and pmvt calls in the code, see also mvtnorm.control for details.
std	Optional logical value determining, whether standardized versions should be assumed for calculation of the optimal contrasts. If FALSE all model parameters need to be specified in the models argument (also location and scale parameters).
off	Fixed offset parameter needed when the linear in log model is used. See also documentation of the linear in log model: "linlog". When off = NULL by default (maximum dose)*0.1 is used for off.
scal	Fixed scale parameter needed when the beta model is used. See also documentation of the beta model: "betaMod". When scal = NULL by default (maximum dose)*1.2 is used for scal.

Value

An object of class gMCPtest, it is a list containing entries

contMat	The used contrast matrix
corMat	The used correlation matrix

tStat	The calculated test-statistics
alpha	The used type one error rate
alternative	The alternative used for the test
critVal	The used critical value

Author(s)

Bjoern Bornkamp

References

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

See Also

[MCPtest](#)

Examples

```
## apply to normal data
data(biom)
## produce first stage ANOVA fit
anMod <- lm(resp~factor(dose)-1, data=biom)
drFit <- coef(anMod)
vCov <- vcov(anMod)
dose <- sort(unique(biom$dose))
## now fit an emax model to these estimates
models <- list(emax = 0.2, linear = NULL)
gtst <- gMCPtest(dose, drFit, vCov, models = models, critV=TRUE)
print(gtst)

## apply to binary migraine data
data(migraine)
PFrate <- migraine$painfree/migraine$nrtrt
doseVec <- migraine$dose
doseVecFac <- as.factor(migraine$dose)
## fit logistic regression with dose as factor
logfit <- glm(PFrate~doseVecFac-1, family = binomial, weights = migraine$nrtrt)
drEst <- coef(logfit)
vCov <- vcov(logfit)
models <- list(linear = NULL, emax = 20)
gtst2 <- gMCPtest(doseVec, drEst, vCov, models = models)
gtst2

## use user specified contrast matrix
data(biom)
## calculate a contrast matrix
mu1 <- c(1, 2, 2, 2, 2)
mu2 <- c(1, 1, 2, 2, 2)
```

```

mu3 <- c(1, 1, 1, 2, 2)
mMat <- cbind(mu1, mu2, mu3)
dimnames(mMat)[[1]] <- sort(unique(biom$dose))
contMat <- planMM(muMat = mMat, doses = doses, n = 20, cV = FALSE)$contMat
## fit first stage model
anMod <- lm(resp~factor(dose)-1, data=biom)
drFit <- coef(anMod)
vCov <- vcov(anMod)
## perform MCP analysis
gMCPtest(drEst = drFit, vCov = vCov, contMat = contMat)

```

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".
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 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 `plotModels` 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 is 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](#), [plotModels](#)

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")
## local approach
emx2 <- guesst(d=0.3, p=0.8, model="emax", local = TRUE, Maxd = 1)
## plot models
models <- list(emax=c(emx1, emx2))
plotModels(models, c(0,1), base= 0, maxEff = 1)

## Logistic model
## Select two (d,p) pairs (0.2, 0.5) and (0.6, 0.95)
lgc1 <- guesst(d = c(0.2, 0.5), p = c(0.6, 0.95), "logistic")
```

```

## local approach
lgc2 <- guesst(d = c(0.2, 0.5), p = c(0.6, 0.95), "logistic",
              local = TRUE, Maxd = 1)
## plot models
models <- list(logistic = matrix(c(lgc1, lgc2), byrow = TRUE, nrow = 2))
plotModels(models, c(0,1), base= 0, maxEff = 1)

## Beta Model
## Select one pair (d,p): (0.5,0.5)
## dose, where maximum occurs: 0.8
bta <- guesst(d=0.5, p=0.5, model="betaMod", dMax=0.8, scal=1.2, Maxd=1)
## plot
models <- list(betaMod = bta)
plotModels(models, c(0,1), base= 0, maxEff = 1)

## Sigmoid Emax model
## Select two (d,p) pairs (0.2, 0.5) and (0.6, 0.95)
sgE1 <- guesst(d = c(0.2, 0.5), p = c(0.6, 0.95), "sigEmax")
## local approach
sgE2 <- guesst(d = c(0.2, 0.5), p = c(0.6, 0.95), "sigEmax",
              local = TRUE, Maxd = 1)
models <- list(sigEmax = matrix(c(sgE1, sgE2), byrow = TRUE, nrow = 2))
plotModels(models, c(0,1), base= 0, maxEff = 1)

## 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 <- list(quadratic = quad)
plotModels(models, c(0,1), base= 0, maxEff = 1)

## exponential model
## (d,p) = (0.8,0.5)
expo <- guesst(d = 0.8, p = 0.5, "exponential", Maxd=1)
models <- list(exponential = expo)
plotModels(models, c(0,1), base= 0, maxEff = 1)

```

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.

The difference to the IBS data set (in the MCPMod package) is the inclusion of the gender column.

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

LP	<i>Sensitivity analysis for misspecification of standardized model parameters in MCPMod</i>
----	---------------------------------------------------------------------------------------------

Description

Calculates the loss in power associated with misspecification of the standardized model parameters for a specific model in the MCPMod procedure.

Usage

```
LP(models, model, type = c("both", "LP1", "LP2"), paramRange,
    doses, base, maxEff, sigma, n, len = c(10, 1), nr = 1,
    alpha = 0.025, alternative = c("one.sided", "two.sided"),
    off = 0.1 * max(doses), scal = 1.2 * max(doses),
    control = mvtnorm.control())
```

Arguments

models	A list specifying the candidate models. This can also be a fullMod object, then the arguments base, maxEff, off and scal are ignored
model	Character string giving the model for which the sensitivity should be investigated.
type	Character string: One of "LP1", "LP2" or "both".
paramRange	Numeric of length two, giving lower and upper limits for standardized model parameter values when the model has just one standardized model parameter. For models with two standardized model parameters a 2x2 matrix with the boundaries for each standardized model parameter in the rows. See examples for details.
doses	Dose levels to be administered

base	Placebo effect
maxEff	Maximum change from placebo (in considered dose range)
sigma	Standard deviation
n	Numeric vector of sample sizes per group. In case just one number is specified, it is assumed that all group sample sizes are equal to this number.
len	Number of points in the standardized model parameter range on which LP is calculated. Has to be of length 2 in case of models with 2 standardized model parameters.
nr	Numeric giving the number of the model (in the order given in the model argument) in case there is more than one model from one model class in the candidate set (e.g. two emax models).
alpha	Level of significance (default: 0.025)
alternative	Character determining the alternative for the multiple contrast trend test.
off	Offset parameter for the linear in log model (default 10 perc. of maximum dose).
scal	Scale parameter for the beta model (default 20 perc. larger than maximum dose).
control	A list of options for the pmvt and qmvt functions as produced by mvtnorm.control

Details

For a given set of candidate models the power-sensitivity of the multiple contrast test with respect to misspecification of the guesstimates is investigated. Two measures to measure loss in power ("LP1" or "LP2") can be used. Roughly LP1 can be interpreted as the difference between the power that "was intended" (nominal power), when designing the study and "what one actually gets" (actual power).

LP2 can be interpreted as the difference between "what could be achieved knowing the true value of the parameter in advance" (potential power) and "what one actually gets". For a detailed definition see the reference below. The power values are calculated on a number of points specified by the len argument. The calculation of LP2 is computationally more demanding as the optimal contrasts and the critical value need to be recalculated for each point in the standardized model parameter space.

Value

An object of class LP, i.e. a matrix containing the different alternative standardized model parameters, associated potential/actual power values and the loss in power values.

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., 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

[plot.LP](#), [guesst](#)

Examples

```
## Not run:
doses <- c(0,10,25,50,100,150)
models <- list(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))

## Examples from JBS paper, p.654
LPobj <- LP(models, model = "emax", type = "both", paramRange = c(10,70),
            doses = doses, base = 0, maxEff = 0.4, sigma = 1, n = 60,
            alpha = 0.05, len = 15, scal = 200)
print(LPobj)
plot(LPobj)

## for exponential model with fullMod and LP1:
fMod <- fullMod(models, doses, base = 0, maxEff = 0.4, scal=200)
LPobj <- LP(fMod, "exponential", "LP1", c(50, 120), sigma = 1,
            alpha = 0.05, len = 20, n = 60)
plot(LPobj)

## Examples for models with two standardized model parameters
LP(models, "betaMod", "LP1",
    paramRange = matrix(c(0.3,1.9,0.4,2.5),nrow=2),
    doses, 0, 0.4, 1, 60, alpha=0.05, len=c(10,4), scal=200)

## Time consuming example
LPobj <- LP(models, "logistic", "both",
            paramRange = matrix(c(40,5,60,15),nrow=2),
            doses, 0, 0.4, 1, 60, alpha=0.05, len=c(10,4), scal=200)
plot(LPobj)

## End(Not run)
```

MCPMod

Perform MCPMod analysis of a data set

Description

Tests for a dose-response effect using a model-based multiple contrast test (see [MCPtest](#)), selects one (or several) model(s) from the significant shapes, fits them using [fitDRModel](#) and estimates the MED or the ED (see [MED.DRMod](#), [ED.DRMod](#)). For details see Bretz et al. (2005).

Usage

```
MCPMod(formula, data, models = NULL, addCovars = ~1,
        contMat = NULL, critV = NULL, off = NULL, scal = NULL,
        alpha = 0.025, alternative = c("one.sided", "two.sided"),
        direction = c("increasing", "decreasing"),
        selModel = c("maxT", "AIC", "BIC", "aveAIC", "aveBIC"),
```

```
doseEst = c("MED2", "MED1", "MED3", "ED", "MED1old", "MED2old", "MED3old"),
doseEstPar = NULL, std = TRUE, start = NULL,
uModPars = NULL, addArgs = NULL, clinRel = NULL,
lenDose = 101, pWeights = NULL, fitControl = fit.control(),
optimizer = c("nls", "nls&bndnls", "bndnls"), pVal = TRUE,
testOnly = FALSE, mvtcontrol = mvtnorm.control(),
na.action = na.fail, bnds = NULL, uGrad = NULL)
```

Arguments

formula	A formula object specifying the names of the response and the dose variable contained in 'data' (in the form response ~ dose). Additional covariates need to be specified via the addCovars argument, see below for details.
data	Data frame containing the dose response data and possible additional covariates specified in the argument 'addCovars'. The code assumes that there are columns corresponding to the dose and the response variable are named "dose" and "resp". Otherwise the names of the dose and the response column can be handed over via the dose and resp argument see below.
models	A list specifying the candidate models. The names of the list entries should be equal to the names of the model functions. The list entries should be equal to the guesstimates. See the Examples (ii) for details on this topic. If the contMat argument is specified, this argument is ignored, see Examples (iv).
addCovars	Additional covariates to be adjusted for linearly in the model should be specified as a formula (as is standard in R). By default there is only an intercept included in the model (note that an intercept should always be included). See the example section for examples how to specify the formula for the additional covariates.
contMat	Optional matrix containing the optimal contrasts in the columns. The names of the columns should be equal to the names of the underlying model functions. If specified the code does not calculate the optimal contrasts.
critV	Critical value, if NULL, no critical value will be calculated, and the test decision will be based on the p-values. If critV = TRUE the critical value will be calculated (the test decision will be based on the critical value). If critV is equal to a numerical value it will be assumed that the critical value is pre-specified and it will not be calculated by the code (the test decision will then also be based on the critical value).
off	Fixed 'offset' parameter needed when the linear in log model is used. See the documentation of the linear in log model: linlog . When off = NULL by default (maximum dose)*0.1 is used for off.
scal	Fixed scale parameter needed when the beta model is used. See the documentation of the beta model: betaMod . When scal = NULL by default (maximum dose)*1.2 is used for scal.
alpha	Level of significance for the multiple contrast test (defaults to 0.025)
alternative	Character determining the alternative for the multiple contrast trend test. Note that two sided alternatives, in most situations, do not really make sense. When the contrast matrix is handed over via the 'contMat' argument the alternative argument only determines the sidedness of the test (the direction is implicit in the contrast matrix).

direction	Character determining the trend direction of the data, which one wants to investigate (e.g., if one wants to investigate whether the response gets larger with increasing dose, direction should be equal to "increasing").
selModel	Optional character vector specifying the model selection criterion for dose estimation. Possible values are "maxT": Selects the model corresponding to the largest t-statistic (this is the default). "AIC": Selects model with smallest AIC "BIC": Selects model with smallest BIC "aveAIC": Uses a weighted average of the models corresponding to the significant contrasts. The model weights are chosen by the formula: $w_i = \exp(-0.5AIC_i) / \sum(\exp(-0.5AIC_i))$. See Buckland et al. (1997) for details. "aveBIC": Same as "aveAIC", but the BIC is used to calculate the model weights.
doseEst	Determines which dose estimator to use, possible values are "MED2", "MED1", "MED3", "MED1old", "MED2old", "MED3old" and "ED". The MED1-MED3 estimators calculate the MED based on the dose effect curve. The dose effect curve takes the difference between the model evaluated at a dose and the model evaluated at placebo (and hence is 0 at placebo). The MED is now calculated as the dose that achieves <code>clinRel</code> in the dose effect curve and a second requirement based on the pointwise confidence interval for the mean. The types MED1-MED3 are obvious generalization of the estimators described in Bretz et al. (2005). For back-compatibility with old versions of the MCPMod function the estimators MED1old-MED3old are included, which are based on the full dose-response model. They can only be used, when there are no additional covariates (that is if <code>addCovars = ~1</code>). See Bretz et al. (2005) for a detailed description of MED1old-MED3old. If ED is specified, the code returns the dose that gives a pre-specified percentage of the maximum effect, observed within the dose-range (not the asymptotic maximum effect!).
doseEstPar	Numeric, defining parameter used for dose estimators. For the MED-type estimators <code>doseEstPar</code> determines the confidence level γ used in the estimator. The used confidence level is given by $1-2*\text{doseEstPar}$. The default for <code>doseEstPar</code> for MED-type estimators is 0.1. For ED-type estimators <code>doseEstPar</code> determines which effective dose is estimated. Specifying 0.95 for example results in an estimate of the ED95. If the ED estimator is used the default for <code>doseEstPar</code> is 0.5.
std	Optional logical value determining, whether standardized versions should be assumed for calculation of the optimal contrasts. If FALSE all model parameters need to be specified in the models argument (also location and scale parameters).
start	List containing starting values for the nls fitting algorithm. The names of the list elements need to be equal to the names of the model functions. The names of the starting vector should equal the corresponding names for the model parameters. For built-in models starting values need to be provided only for the non-linear parameters (see last example of (i) for an illustration). In case of a user model (not built in) starting values for the all parameters need to be supplied (see Example (iii)). When the optimizer "bndnls" is used, the start argument is ignored

	as this option does not require a starting value.
<code>uModPars</code>	Optional character vector with names/expressions of user-defined model parameters (names(start) used by default).
<code>addArgs</code>	Optional character vector with names of additional arguments (variables) to user-defined model.
<code>clinRel</code>	Numeric of length one, specifying the clinical relevance threshold (needed for the MED estimate).
<code>lenDose</code>	Numeric vector specifying the number of points in the dose-range to search for the dose estimate, defaults to 101.
<code>pWeights</code>	Optional vector specifying prior weights for the different models. Should be a named vector with names matching the names of the models list. Only relevant if <code>selModel = "aveAIC"</code> or <code>"aveBIC"</code>
<code>fitControl</code>	List of parameters to be used when fitting the dose response models, see the fit.control function for details.
<code>optimizer</code>	Type of optimizer to be used for calculating the parameter estimates of the dose-response model. Option <code>"nls"</code> uses the <code>nls</code> function with <code>"plinear"</code> option. Option <code>"bndnls"</code> uses a non-linear least squares algorithm with simple box constraints on the (non-linear) parameters (based on a grid search and <code>nlsminb</code> , see fitDRModel for details). The advantage of imposing bounds is that it guarantees the existence of an optimum of the objective function. Option <code>"nls&bndnls"</code> first uses <code>nls</code> and when non-convergence of <code>"nls"</code> occurs it uses the non-linear least squares algorithm with bounds <code>"bndnls"</code> .
<code>pVal</code>	Optional logical determining whether p-values should be calculated, defaults to TRUE. If the critical value is supplied, p-values will not be calculated.
<code>testOnly</code>	Logical value determining, whether only the multiple comparisons test should be performed. See Examples (v) below. If you are really only interested in the testing part consider using the MCPtest function, which is solely build for this purpose.
<code>mvtcontrol</code>	A list specifying additional control parameters for the <code>qmvt</code> and <code>pmvt</code> calls in the code, see also <code>mvtnorm.control</code> for details.
<code>na.action</code>	A function which indicates what should happen when the data contain NAs.
<code>bnds</code>	A list containing bounds for non-linear parameters, needed when <code>optimizer="bndnls"</code> . The list elements should correspond to the bounds for the individual models, the list elements should be named according to the underlying dose-response models. See getBnds for details on the formatting of the bounds. If not specified, reasonable bounds are extracted from the data (using the getBnds function based on its defaults).
<code>uGrad</code>	Function to return the gradient of a user defined model, see Examples (iii) below.

Details

This function performs the multiple comparisons and modelling (MCPMod) procedure presented in Bretz et al. (2005). The method consists of two steps:

(i) MCP step (apply [MCPtest](#) function): The function calculates the optimal contrasts (if not supplied) and the contrast test statistics. In the calculation of the critical value and p-values multiplicity

is taken into account.

(ii) Modelling step (apply `fitDRModel` function): If there is at least one significant contrast, one model or a combination of models is chosen (depending on the `selModel` argument) for dose estimation. In case of non-convergence of certain non-linear models the remaining significant models are used. Finally the target dose is estimated.

Built in models are the linear, linear in log, emax, sigmoid emax, logistic, exponential, quadratic and beta model (for their definitions see [DR-Models](#)). Users may hand over their own model functions for details have a look at the Example (iii).

Value

An object of class MCPMod, with the following entries:

<code>signf</code>	Logical indicating, whether multiple contrast test is significant
<code>model1</code>	Model with largest contrast test statistic
<code>model2</code>	Model(s) used for estimation of target doses
<code>input</code>	A list with information on input parameters for the MCPMod function: <code>formula</code> , <code>models</code> , <code>off</code> , <code>scal</code> , <code>alpha</code> , <code>alpha</code> , first entry of <code>alternative</code> , <code>direction</code> , first entry of <code>selModel</code> , <code>doseEst</code> , <code>std</code> , <code>doseEstPar</code> , <code>uModArgs</code> , <code>addArgs</code> , <code>start</code> , <code>uGrad</code> , <code>clinRel</code> , <code>lenDose</code> , <code>pVal</code> , <code>testOnly</code> , first entry of <code>optimizer</code> , <code>addCovars</code>
<code>data</code>	The data set.
<code>contMat</code>	The contrast matrix.
<code>corMat</code>	The correlation matrix.
<code>cVal</code>	The critical value for the multiple contrast test.
<code>tStat</code>	The contrast test-statistics. If <code>'pVal=TRUE'</code> the p-values are also attached.
<code>fm</code>	List containing the dose-response model(s) used for dose-estimation. These are all DRMod objects, which can be further used. See the <code>fitDRModel</code> function for details on DRMod objects and methods that can be applied to DRMod objects. Note that the data set used for fitting the DRMod object is not included in the objects, so that most methods (for example the <code>predict</code> method when <code>se.fit = T</code> or the <code>vcov</code> method) require to specify the data on which the model was fitted via the <code>data</code> argument of the methods.
<code>estDose</code>	Estimated dose(s), in case of model averaging the dose estimates under the individual models are attached.

Note: If `testOnly=TRUE`, or no model is significant, the object does not contain `fm` and `estDose` entries

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
- 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

Bretz, F., Pinheiro, J. C., and Branson, M. (2004), On a hybrid method in dose-finding studies, *Methods of Information in Medicine*, **43**, 457–460

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

[MCPtest](#), [fitDRModel](#), [plot.MCPMod](#), [predict.MCPMod](#), [logistic](#), [sigEmax](#), [linlog](#), [linear](#), [quadratic](#), [emax](#), [betaMod](#), [exponential](#), [mvtnorm.control](#), [getBnds](#)

Examples

```
## (i)
## example from Biometrics paper p. 743
data(biom)
models <- list(linear = NULL, linlog = NULL, emax = 0.2,
               exponential = c(0.279,0.15), quadratic = c(-0.854,-1))
dfe <- MCPMod(resp ~ dose, biom, models, alpha = 0.05, doseEstPar = 0.05,
              pVal = TRUE, selModel = "maxT", doseEst = "MED2old",
              clinRel = 0.4, off = 1)
## detailed information is available via summary
summary(dfe)
## plots data with selected model function
plot(dfe)

## example with IBS data
## now explicitly calculate critical value and perform
## model averaging based on the AIC
data(IBScovars)
models <- list(emax = 0.2, quadratic = -0.2, linlog = NULL)
dfe2 <- MCPMod(resp ~ dose, IBScovars, models, alpha = 0.05, critV = TRUE,
               pVal = TRUE, selModel = "aveAIC",
               clinRel = 0.25, off = 1)
dfe2
## illustrate some methods for MCPMod objects
summary(dfe2)
plot(dfe2, complData = TRUE)
plot(dfe2, CI = TRUE, clinRel = TRUE)

## use different optimizer (and include some models
## not converging using nls)
models <- list(quadratic = -0.2, linlog = NULL, betaMod = c(1,1),
               sigEmax = rbind(c(0.2,1), c(1, 3)))
dfe3 <- MCPMod(resp ~ dose, IBScovars, models, alpha = 0.05, pVal = TRUE,
```

```

selModel = "aveAIC", clinRel = 0.25, off = 1,
scal = 6, optimizer = "bndnls")
plot(dfe3)

## use additional linear covariates
data(IBScovars)
models <- list(emax = 0.2, quadratic = -0.2, linlog = NULL)
dfe4 <- MCPMod(resp ~ dose, IBScovars, models, addCovars = ~gender,
alpha = 0.05, pVal = TRUE,
selModel = "aveAIC", clinRel = 0.25, off = 1)
plot(dfe4, CI = TRUE) # plot method now only plots the effect curves

## simulate dose-response data
set.seed(1)
dfData <- genDFdata(model = "emax", argsMod = c(e0 = 0.2, eMax = 1,
ed50 = 0.05), doses = c(0,0.05,0.2,0.6,1), n=20, sigma=0.5)
models <- list(emax = 0.1, logistic = c(0.2, 0.08),
betaMod = c(1, 1))
## hand over starting values manually
start <- list(emax = c(ed50 = 0.1), logistic = c(ed50=0.05, delta =
0.1), betaMod = c(delta1 = 0.5, delta2 = 0.5))
dfe5 <- MCPMod(resp ~ dose, dfData, models, clinRel = 0.4, critV = 1.891,
scal = 1.5, start = start)

## (ii) Example for constructing a model list

## Contrasts to be included:
## Model guesstimate(s) for stand. model parameter(s) (name)
## linear -
## linear in log -
## Emax 0.2 (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)
##
## For each model class exactly one list entry needs to be used.
## The names for the list entries need to be written exactly
## as the model functions ("linear", "linlog", "quadratic", "emax",
## "exponential", "logistic", "betaMod", "sigEmax").
## For models with no parameter in the standardized model just NULL is
## specified as list entry.
## For models with one parameter a vector needs to be used with length
## equal to the number of contrasts to be used for this model class.
## For the models with two parameters in the standardized model a vector
## is used to hand over the contrast, if it is desired to use just one
## contrast. Otherwise a matrix with the guesstimates in the rows needs to
## be used. For the above example the models list has to look like this

list(linear = NULL, linlog = NULL, emax = c(0.2, 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))

## Additional parameters (which will not be estimated) are the "off"
## parameter for the linlog model and the "scal" parameter for the
## beta model, which are not handed over in the model list.

## (iii) example for incorporation of a usermodel
## simulate dose response data
dats <- genDFdata("sigEmax", c(e0 = 0, eMax = 1, ed50 = 2, h = 2),
                 n = 50, sigma = 1, doses = 0:10)
## define usermodel
userMod <- function(dose, a, b, d){
  a + b*dose/(dose + d)
}
## define gradient
userModGrad <-
  function(dose, a, b, d) cbind(1, dose/(dose+d), -b*dose/(dose+d)^2)
## name starting values for nls
start <- list(userMod=c(a=1, b=2, d=2))
models <- list(userMod=c(0, 1, 1), linear = NULL)
dfe6 <- MCPMod(resp ~ dose, dats, models, clinRel = 0.4, selModel="AIC",
              start = start, uGrad = userModGrad)

## (iv) Contrast matrix and critical value handed over and not calculated
## simulate dose response data
dat <- genDFdata(mu = (0:4)/4, n = 20,
                sigma = 1, doses = (0:4)/4)
## construct optimal contrasts and critical value with planMM
doses <- (0:4)/4
mods <- list(linear = NULL, quadratic = -0.7)
pM <- planMM(mods, doses, 20)
fit <- MCPMod(resp ~ dose, dat, models = NULL, clinRel = 0.3,
              contMat = pM$contMat, critV = pM$critVal)

## (v) Use self constructed contrasts (and hand them over)
mu1 <- c(1, 2, 2, 2, 2)
mu2 <- c(1, 1, 2, 2, 2)
mu3 <- c(1, 1, 1, 2, 2)
mMat <- cbind(mu1, mu2, mu3)
dimnames(mMat)[[1]] <- doses
pM <- planMM(muMat = mMat, doses = doses, n = 20, cV = FALSE)
## use MCPMod only for testing part (see also MCPtest function)
fit <- MCPMod(resp ~ dose, dat, models = NULL, contMat = pM$contMat,
              pVal = TRUE, testOnly = TRUE, critV=TRUE)
summary(fit)

```

Description

Perform a multiple contrast test with model-based (optimal) contrasts for the dose variable (or contrasts specified via `contMat`) and possible additional covariates.

Usage

```
MCPtest(formula, data, models, addCovars = ~1,
         alpha = 0.025, contMat = NULL, critV = NULL, pVal = TRUE,
         alternative = c("one.sided", "two.sided"),
         direction = c("increasing", "decreasing"),
         na.action = na.fail, mvtcontrol = mvtnorm.control(),
         std = TRUE, off, scal)
```

Arguments

<code>formula</code>	A formula object specifying the response and the dose variable (in the form <code>response ~ dose</code>). Additional covariates need to be specified via the <code>addCovars</code> argument, see below for details.
<code>data</code>	Dose Response data frame containing the variables needed for performing the multiple contrast test.
<code>models</code>	A candidate models list. A list specifying the model shapes to be included in the contrast matrix. The names of the list entries should be equal to the names of the model functions. The list entries should be equal to prior estimates for standardized model parameters. See the MCPMod function for details on how to specify candidate model shapes, and DR-Models for the definition of the implemented dose-response models.
<code>addCovars</code>	Formula specifying additional (linear) covariates
<code>alpha</code>	Significance level for the multiple contrast test
<code>contMat</code>	Optional matrix containing the optimal contrasts in the columns. If specified the code does not calculate the optimal contrasts.
<code>critV</code>	Critical value, if <code>NULL</code> , no critical value will be calculated, and the test decision will be based on the p-values. If <code>critV = TRUE</code> the critical value will be calculated (the test decision will be based on the critical value). If <code>critV</code> is equal to a numerical value it will be assumed that the critical value is pre-specified and it will not be calculated by the code (the test decision will then also be based on the critical value).
<code>pVal</code>	Optional logical determining whether p-values should be calculated, defaults to <code>TRUE</code> . If the critical value is supplied, p-values will not be calculated.
<code>alternative</code>	Character determining the alternative for the multiple contrast trend test.
<code>direction</code>	Character determining the trend direction of the data, which one wants to investigate (e.g., if one wants to investigate whether the response gets larger with increasing dose direction should be equal to "increasing"). When the contrast matrix is handed over via the <code>'contMat'</code> argument the <code>direction</code> argument is ignored (the direction is implicit in the contrast matrix).
<code>na.action</code>	A function which indicates what should happen when the data contain NAs.

<code>mvtcontrol</code>	A list specifying additional control parameters for the <code>qmv</code> and <code>pmvt</code> calls in the code, see also <code>mvtnorm.control</code> for details.
<code>std</code>	Optional logical value determining, whether standardized versions should be assumed for calculation of the optimal contrasts. If <code>FALSE</code> all model parameters need to be specified in the <code>models</code> argument (also location and scale parameters).
<code>off</code>	Fixed offset parameter needed when the linear in log model is used. See also documentation of the linear in log model: "linlog". When <code>off = NULL</code> by default (maximum dose)*0.1 is used for <code>off</code> .
<code>scal</code>	Fixed scale parameter needed when the beta model is used. See also documentation of the beta model: "betaMod". When <code>scal = NULL</code> by default (maximum dose)*1.2 is used for <code>scal</code> .

Value

An object of class `MCPtest` containing a list with values

<code>contMat</code>	The contrast matrix (either calculated by <code>MCPtest</code> or handed over as an argument). The contrasts are in the columns of the matrix.
<code>tStat</code>	The individual contrast t-statistics (with associated p-values)
<code>alpha</code>	The significance level
<code>twoSide</code>	Numeric specifying, whether two-sided or one-sided testing has been performed
<code>critVal</code>	The critical value (either calculated by <code>MCPtest</code> or handed over as an argument)

Author(s)

Bjoern Bornkamp

See Also

[critVal](#)

Examples

```
## example without covariates
data(biom)
modlist <- list(emax = 0.05, linear = NULL, logistic = c(0.5, 0.1))
fit1 <- MCPtest(resp ~ dose, biom, modlist)
## now calculate critical value (but not p-values)
fit2 <- MCPtest(resp ~ dose, biom, modlist, critV = TRUE, pVal = FALSE)
## now hand over critical value
fit3 <- MCPtest(resp ~ dose, biom, modlist, critV = 2.24)

## example with covariates
data(IBScovars)
modlist <- list(emax = 0.05, linear = NULL, logistic = c(0.5, 0.1))
MCPtest(resp ~ dose, IBScovars, modlist, addCovars = ~gender)

## example with contrast matrix handed over
data(biom)
```

```

## calculate a contrast matrix
mu1 <- c(1, 2, 2, 2, 2)
mu2 <- c(1, 1, 2, 2, 2)
mu3 <- c(1, 1, 1, 2, 2)
mMat <- cbind(mu1, mu2, mu3)
dimnames(mMat)[[1]] <- sort(unique(biom$dose))
pM <- planMM(muMat = mMat, doses = doses, n = 20, cV = FALSE)
## perform MCP analysis
MCPtest(resp~dose, data = biom, contMat = pM$contMat)

```

MED.DRMod

Calculate MED for a DRMod object

Description

This function calculates the minimum effective dose (MED) for a DRMod object. The MED is the smallest dose that achieves a clinical relevant effect. See Bretz et al. (2005) for a detailed definition.

Usage

```

## S3 method for class 'DRMod'
MED(object, type = c("MED2", "MED1", "MED3"),
     clinRel, gamma = 0.05, old = FALSE,
     direction = c("increasing", "decreasing"),
     doseSeq = NULL, lenDose = 101, data = getData(object), uGrad, ...)

```

Arguments

object	An object of class DRMod
type	The type of MED estimator to be used. "MED1"- "MED3" are generalization of the estimators described in Bretz et al. (2005), for the case of no covariates. To obtain the estimators described in Bretz et al. (2005), set the argument old below to TRUE. See also the examples below for details.
clinRel	Numeric specifying the clinical threshold(s) to be used for MED estimation. Should be of the same length as gamma, or of length 1.
gamma	Numeric specifying the gamma parameter needed for the confidence interval used in the MED estimate. Should be in (0,0.5] as 1-2*gamma confidence intervals are used. gamma should be of the same length as clinRel, or of length 1.
old	Logical indicating, whether old versions of the MED estimates should be calculated.
direction	Character specifying the direction of the dose effect.
doseSeq	Sequence of values allowed for the dose estimator. Should include 0 (ie placebo) as a first value.

lenDose	If doseSeq is not specified an equally spaced grid of lenDose values between placebo and the maximal dose in the study is used.
data	Data frame containing the variables on which the DRMod object was fitted
uGrad	Function to return the gradient of a user defined model, see Examples of the fitDRModel function.
...	Additional arguments

Value

Returns a numeric containing the MED estimates, if the MED cannot be determined within the dose-range NA is returned.

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

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., 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

[fitDRModel](#), [ED](#)

Examples

```
data(biom)
fit <- fitDRModel(resp ~ dose, biom, "logistic")
## MED2 estimates MED as smallest dose where the dose-effect
## curve is larger than clinRel and the lower bound of the
## 1-2*gamma confidence interval for the mean is larger than 0
## select gamma 0.05, 0.1 and 0.5
## (for gamma=0.5 CI is irrelevant for definition of MED estimate)
MED(fit, "MED2", clinRel = 0.2, gamma = c(0.05, 0.1, 0.5))

## option "old" uses the definition given in Bretz et al. (2005)
## to calculate the MED based on the full dose-response function
## this is included for back-compatibility with the old MCPMod function.
## This option is not available if covariates are used
MED(fit, "MED2", clinRel = 0.2, gamma = c(0.05, 0.1, 0.5), old = TRUE)
```

migraine	<i>Migraine Dose Response data</i>
----------	------------------------------------

Description

An example data set obtained from clinicaltrials.gov. 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>

modelMeans	<i>Calculate mean vectors for a given candidate set</i>
------------	---------------------------------------------------------

Description

Calculates the mean or standardized mean vectors for a candidate set of models. This function is mainly for internal use.

Usage

```
modelMeans(models, doses, std = TRUE, off = 0.1 * max(doses),
            scal = 1.2 * max(doses))
```

Arguments

models	A list of candidate models, or the output of the fullMod function (depending on the value of std).
doses	A numeric vector giving the doses to be administered.
std	Logical indicating whether standardized or non-standardized version of model function should be used.
off	Offset parameter for linear in log model.
scal	Scale parameter for beta model.

Value

Matrix with standardized or non-standardized model means.

Examples

```
doses <- c(0, 10, 25, 50, 100, 150)
models <- list(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))
modelMeans(models, doses, std = TRUE)

# now non-standardized means
Models <- fullMod(models, doses, base = 0, maxEff = 0.4, scal = 200)
modelMeans(Models, doses, std = FALSE)
```

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 always the "GenzBretz" algorithm is used, as only this allows for calculating multivariate t-distribution integrals. 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.

See Also

[pmvt](#), [qmvt](#)

 planMM

Calculate optimal contrasts and critical value for MCP test

Description

Calculates the optimal model contrasts, the critical value and the contrast correlation matrix, i.e. the quantities necessary to conduct a multiple contrast test for a given candidate set of dose-response models (see also [MCPtest](#)).

Usage

```
planMM(models, doses, n, off = 0.1 * max(doses), scal = 1.2 * max(doses),
        std = TRUE, alpha = 0.025, alternative = c("one.sided", "two.sided"),
        direction = c("increasing", "decreasing"),
        control = mvtnorm.control(), cV = TRUE, muMat = NULL, vCov = NULL)
```

Arguments

models	A list of candidate models
doses	A numeric vector giving the doses to be administered.
n	The vector of sample sizes per group. In case just one number is specified, it is assumed that all group sample sizes are equal to this number.
off	Offset parameter for the linear in log model (default 10 perc of the maximum dose).
scal	Scale parameter for the beta model (default 20 perc. larger than maximum dose).
std	Optional logical indicating, whether standardized version of the models should be assumed.
alpha	Level of significance (default: 0.025)
alternative	Character determining the type of alternative used for the multiple contrast trend test. When muMat is specified or std = F, this argument is ignored.
direction	Character determining the trend direction of the data, which one wants to investigate (e.g., if one wants to investigate whether the response gets larger with increasing dose direction should be equal to "increasing").
control	A list of options for the pmvt and qmvt functions as produced by mvtnorm.control
cV	Logical indicating whether critical value should be calculated
muMat	An optional matrix with means in the columns and given dimnames (dose levels and names of contrasts). If specified the models argument should not be specified, see examples below.
vCov	Covariance matrix of mean vectors assumed for calculation of optimal contrasts. When vCov is given it is assumed that the estimates are asymptotically normal.

Value

An object of class planMM with the following components:

contMat	Matrix of optimal contrasts.
critVal	The critical value for the test (if calculated)
muMat	Matrix of (non-normalized) model means
corMat	Matrix of the contrast correlations.

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

Bretz, F., Pinheiro, J., 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

See Also

[critVal](#), [MCPtest](#)

Examples

```
## Not run:
## Example from JBS paper
doses <- c(0,10,25,50,100,150)
models <- list(linear = NULL, emax = 25,
               logistic = c(50, 10.88111), exponential= 85,
               betaMod=matrix(c(0.33,2.31,1.39,1.39), byrow=TRUE, nrow=2))
pLM <- planMM(models, doses, n = rep(50,6), alpha = 0.05, scal=200)
plot(pLM)

## example, where means are directly specified
## doses
dvec <- c(0, 10, 50, 100)
## mean vectors
mu1 <- c(1, 2, 2, 2)
mu2 <- c(1, 1, 2, 2)
mu3 <- c(1, 1, 1, 2)
mMat <- cbind(mu1, mu2, mu3)
dimnames(mMat)[[1]] <- dvec
planMM(muMat = mMat, doses = dvec, n = 30)

## End(Not run)
```

plot.fullMod	<i>Plot method for fullMod objects</i>
--------------	----------------------------------------

Description

Plot method for fullMod objects.

Usage

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

Arguments

x	fullMod object
...	Additional arguments for the plotModels function

See Also

[plotModels](#), [fullMod](#)

plot.LP	<i>Plot method for LP objects</i>
---------	-----------------------------------

Description

Graphically displays an LP object.

Usage

```
## S3 method for class 'LP'
plot(x, line = TRUE, type = NULL, spldf = 5, ...)
```

Arguments

x	LP object as obtained from a call to the LP function
line	Logical indicating whether the power values should be smoothed.
type	One of "LP1", "LP2" or "both", availability depending on whether the corresponding values have been calculated in the call to the LP function.
spldf	Numeric determining the degrees of freedom for the smoothing spline which is plotted if line==TRUE. Note that spldf should be larger than 1 and smaller than len (default: 5).
...	Additional arguments.

Details

The function produces a trellis display of the loss in power for different values of the standardized model parameter. A smoothing spline (with `spldf` degrees of freedom) is fit to these points to give a smooth impression of the loss in power curve. For models with two prior parameters a trellis display is shown with the number of panels equal to `len[2]`. The number of points on which the power is evaluated is equal to `len[1]` in each panel, where `len` is an argument of the LP function.

See Also

[LP](#)

Examples

```
## Not run:
doses <- c(0,10,25,50,100,150)
models <- list(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))

# Examples from JBS paper, p.654
LPobj <- LP(models, model = "emax", type = "both", paramRange = c(10,70),
           doses = doses, base = 0, maxEff = 0.4, sigma = 1, n = 60,
           alpha = 0.05, len = 15, scal = 200)

plot(LPobj)
plot(LPobj, line = FALSE, type = "LP1")
plot(LPobj, type = "LP1", spldf = 9)

## End(Not run)
```

plot.MCPMod

Plot MCPMod model fits

Description

The function plots the model(s) used for dose estimation. When there are no covariates in the model the full dose-response model is displayed (possibly including the dose-response data). When there are covariates in the model the plot method displays the effect curves. If it is desired to have a more detailed plot, it might be better to extract the fitted dose-response models from the MCPMod object, and further work with them (see also the plot method for DRMod objects [plot.DRMod](#)).

Usage

```
## S3 method for class 'MCPMod'
plot(x, complData = FALSE, CI = FALSE, clinRel = FALSE, doseEst = FALSE,
     gamma = NULL, models = "all", nrDoseGam = 1,
     colors = c("black", "blue", "black", "gray", "blue"),
     uGrad = NULL, ...)
```

Arguments

x	A MCPMod object.
complData	Logical indicating whether complete data set or group means should be plotted.
CI	Logical indicating whether a confidence interval should be plotted along the model fit(s).
clinRel	Logical indicating, whether clinical relevance threshold should be included in plot.
doseEst	Logical determining whether dose estimate should be included in plot.
gamma	Numeric giving the value for the $1-2*\gamma$ pointwise CI around the predicted mean. if equal to NULL the value determined in the MCPMod call is used. In case a vector of gamma values was used nrDoseGam determines which is used.
models	Character vector determining, which of the used models should be plotted (only available if model averaging was used)
nrDoseGam	In case a vector is specified for dePar in the MCPMod function (and gamma in the plot.MCPMod function is NULL), nrDoseGam determines which of these values should be used for the conf. interval and the dose estimate (if doseEst = T).
colors	Vector of length 5 with the names of the colors for: predictions, CI, data, clinical relevance threshold, dose estimator
uGrad	If a user defined model has been used for dose estimation, the gradient function needs to be handed over via uGrad.
...	Additional arguments to xyplot.

See Also

[MCPMod](#), [plot.DRMod](#)

plot.planMM

Plotting a planMM object

Description

This function displays the contrasts or model means obtained from a planMM object.

Usage

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

Arguments

x	A planMM object.
superpose	Logical, indicating if lines should be superposed.
xlab	Label for x-axis
ylab	Label for y-axis
resp	One of "contrasts" or "means". Determines, whether contrasts or normalized means are plotted.
...	Additional arguments to the xyplot function call.

See Also

[planMM](#)

Examples

```
## Not run:
doses <- c(0, 10, 25, 50, 100, 150)
models <- list(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))
pM <- planMM(models, doses, 50, scal = 200)
plot(pM)
plot(pM, superpose=FALSE, xlab="Different axis name")
plot(pM, resp = "means")
# example with muMat
dvec <- c(0, 10, 50, 100)
mu1 <- c(1, 2, 2, 2)
mu2 <- c(1, 1, 2, 2)
mu3 <- c(1, 1, 1, 2)
mMat <- cbind(mu1, mu2, mu3)
dimnames(mMat)[[1]] <- dvec
pM <- planMM(muMat = mMat, doses = dvec, n = 30)
plot(pM)
plot(pM, superpose=FALSE, xlab="Different axis name")

## End(Not run)
```

plot.powerMM

Plot method for powerMM objects

Description

This function plots the result of the powerMM function call in a trellis display.

Usage

```
## S3 method for class 'powerMM'
plot(x, superpose = TRUE, line.at = NULL, models = "all",
     summ = NULL, perc = FALSE, xlab = NULL,
     ylab = ifelse(perc, "Power (%)", "Power"), ...)
```

Arguments

x	A powerMM object, i.e. a matrix with power values for different sample sizes and models
superpose	Logical, indicating if lines should be superposed.
line.at	A value, or a vector of values, between 0 and 1, to be drawn as horizontal line in the plot (default: not drawn).
models	Character determining which of the models should be included in the plot, "all" and "none" are accepted, else names (or numbers) of models.
summ	Summaries to be included in plot; by default the mean, the minimum and the maximum value are displayed.
perc	Logical indicating if power values should be in percentage.
xlab	Label for x-axis.
ylab	Label for y-axis.
...	Additional arguments for the xypLOT function.

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

[powerMM](#)

Examples

```
## Not run:
# Example from JBS paper
doses <- c(0,10,25,50,100,150)
models <- list(linear = NULL, emax = 25,
              logistic = c(50, 10.88111), exponential= 85,
              betaMod=matrix(c(0.33,2.31,1.39,1.39), byrow=TRUE, nrow=2))
pM <- powerMM(models, doses, base = 0, maxEff = 0.4, sigma = 1,
              lower = 10, upper = 100, step = 20, scal = 200)

pM
plot(pM)
plot(pM, line.at = 0.8, model = c("emax", "linear"), summ = "mean")
plot(pM, line.at = 0.8, model = "none", summ = c("median", "min"))

## End(Not run)
```

plotModels

Plot candidate models

Description

Produces a trellis display of the model functions in the candidate set. The location and scale parameters of the models are determined by the `base` and `maxEff` arguments.

Usage

```
plotModels(models, doses, base = 0, maxEff = 1, nPoints = 200,
           off = 0.1 * max(doses), scal = 1.2 * max(doses),
           superpose = FALSE, ylab = "Model means",
           xlab = "Dose", ...)
```

Arguments

<code>models</code>	A list specifying the candidate models. This can also be a <code>fullMod</code> object, then the arguments <code>base</code> , <code>maxEff</code> , <code>off</code> and <code>scal</code> are ignored.
<code>doses</code>	Dose levels to be administered
<code>base</code>	Expected placebo effect
<code>maxEff</code>	Expected maximum change from placebo (in considered dose range)
<code>nPoints</code>	Number of points for plotting
<code>off</code>	Offset parameter for the linear in log model (default: 10 percent of maximum dose)
<code>scal</code>	Scale parameter for the beta model (default: 20 percent larger than maximum dose)
<code>superpose</code>	Logical determining, whether model plots should be superposed
<code>ylab</code> , <code>xlab</code>	Label for y-axis and x-axis.
<code>...</code>	Additional arguments to the <code>xypLOT</code> call.

References

Bornkamp B., Pinheiro J. C., 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., 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

[guesst](#), [fullMod](#)

Examples

```
## JBS example
doses <- c(0,10,25,50,100,150)
models <- list(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))
plotModels(models, doses, base = 0, maxEff = 0.4, scal = 200)
## all models in one panel
plotModels(models, doses, base = 0, maxEff = 0.4, scal = 200,
           superpose = TRUE)

## plotModels can also be called using a fullMod object
fM <- fullMod(models, doses, base = 0, maxEff = 0.4, scal = 200)
plotModels(fM)
## or even easier
plot(fM)
```

powCalc

Calculate the power for the multiple contrast test

Description

Given the contrast matrix, the sample size and a certain ‘alternative’ (i.e. a mean vector and sigma), the function calculates the power to detect this alternative. See Pinheiro et al. (2006) for details. The function is the building block for the functions `powerScenario`, `powerMM`, `sampSize` and `LP`. Numerical integration routines from the `mvtnorm` package are used to calculate the underlying multivariate integrals.

Usage

```
powCalc(cMat, n, alpha = 0.025, delta = NULL, mu = NULL,
        sigma = NULL, cVal = NULL, corMat = NULL,
        alternative = c("one.sided", "two.sided"),
        control = mvtnorm.control())
```

Arguments

<code>cMat</code>	Matrix with the contrasts in the columns
<code>n</code>	Numeric vector of sample sizes per group. In case just one number is specified, it is assumed that all group sample sizes are equal to this number
<code>alpha</code>	Level of significance (defaults to 0.025)
<code>delta</code>	Non-centrality vector of the distribution of the test statistic under the alternative.
<code>mu</code>	Mean vector under the alternative. The function then calculates the non-centrality vector itself. Ignored if <code>delta</code> is specified.
<code>sigma</code>	Expected standard deviation of the response. Only necessary if the non-centrality vector is to be calculated by the function (i.e. if <code>delta</code> is <code>NULL</code>).

cVal	Optional numeric vector giving the critical value, if specified the argument alpha is ignored.
corMat	An optional matrix giving the correlations of the contrasts specified in cMat.
alternative	Character determining the type of alternative used for the multiple contrast trend test.
control	A list of options for the pmvt and qmvt functions as produced by mvtnorm. control.

Value

The function returns the power value.

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

[planMM](#), [LP](#), [sampSize](#), [powerMM](#), [powerScenario](#)

Examples

```
doses <- c(0,10,25,50,100,150)
models <- list(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))

## calculate optimal contrasts and critical value
plMM <- planMM(models, doses, 50, scal = 200, alpha = 0.05)

## calculate mean vectors
compMod <- fullMod(models, doses, base = 0, maxEff = 0.4, scal = 200)
muMat <- modelMeans(compMod, doses, FALSE, scal = 200)

## calculate power to detect mean vectors
## Power for linear model
powCalc(plMM$contMat, 50, mu = muMat[,1], sigma = 1, cVal = plMM$critVal)
## Power for emax model
powCalc(plMM$contMat, 50, mu = muMat[,2], sigma = 1, cVal = plMM$critVal)
## Power for logistic model
powCalc(plMM$contMat, 50, mu = muMat[,3], sigma = 1, cVal = plMM$critVal)
## compare with JBS 16, p. 650
```

powerMM

*Calculate power for different sample sizes***Description**

Calculates the power under the assumed candidate set for different sample sizes.

To investigate the power of the chosen candidate set against alternatives not included in the candidate model set, see the [powerScenario](#) function.

Usage

```
powerMM(models, doses, base, maxEff, sigma, lower, upper, step,
        sumFct = c("min", "mean", "max"), off = 0.1 * max(doses),
        scal = 1.2 * max(doses), alpha = 0.025,
        alternative = c("one.sided", "two.sided"),
        control = mvtnorm.control(), muMat = NULL, alRatio = NULL,
        typeN = c("arm", "total"), vCov = NULL, ...)
```

Arguments

models	A list specifying the candidate models. This can also be a fullMod object, then the arguments base, maxEff, off and scal are ignored.
doses	Dose levels to be administered
base	Expected placebo effect
maxEff	Expected maximum change from placebo (within considered dose range)
sigma	Expected standard deviation
lower, upper	Maximum and minimum group sample size for which the power is calculated.
step	Stepsize for the sample size at which the power is calculated. It is calculated at seq(lower, upper, by=step).
sumFct	A character vector giving the names of the summary functions used to combine the power values into one value. By default the minimum, the mean and the maximum are used.
off	Offset parameter for the linear in log model (default 10 perc. of maximum dose).
scal	Scale parameter for the beta model (default 20 perc. larger than maximum dose).
alpha	Level of significance (default: 0.025)
alternative	Character determining the type of alternative used for the multiple contrast trend test.
control	A list of options for the pmvt and qmvt functions as produced by mvtnorm.control.
muMat	An optional matrix with means in the columns, dimnames should be given (dose levels and names of contrasts), if specified the the models argument should not be specified, see examples below.

alRatio	Vector describing the relative patient allocations to the dose groups. See examples below, e.g. <code>c(1,2,2)</code> corresponds to allocating twice as many patients in dose groups two and three. Per default balanced allocations are assumed.
typeN	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. See examples below.
vCov	Covariance matrix of mean vectors assumed for calculation of optimal contrasts.
...	Possible additional arguments for <code>sumFct</code> .

Details

Given the candidate set of models and associated guesstimates the function calculates the power to detect every model in the candidate set for different group sample sizes. Additionally summary functions can be specified to calculate the combined power (by default the minimum, mean and maximum). The location and scale parameters are determined by forcing the model function to go through (0,base) and (dmax,maxEff), see Pinheiro et al. (2006) for details. There exists a plot method for the output of the powerMM function. See the examples below.

Value

A powerMM object, i.e. a matrix containing the power values for different sample sizes and models

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., 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

[plot.powerMM](#), [powCalc](#), [powerScenario](#)

Examples

```
## Not run:
doses <- c(0,10,25,50,100,150)
models <- list(linear = NULL, emax = 25,
               logistic = c(50, 10.88111), exponential= 85,
               betaMod=matrix(c(0.33,2.31,1.39,1.39), byrow=TRUE, nrow=2))
pM <- powerMM(models, doses, base = 0, maxEff = 0.4, sigma = 1,
              alpha = 0.05, lower = 10, upper = 100, step = 20, scal = 200)

pM
## a graphical display provides plot method
plot(pM)
## reproduces plot in JBS 16, p.651
plot(pM, line.at = 0.8, models = "none")

## the same with fullMod object and default alpha
```

```

fMod <- fullMod(models, doses, base = 0, maxEff = 0.4, scal=200)
pM <- powerMM(fMod, sigma = 1, lower = 10, upper = 100,
              step = 20, scal = 200)
pM

## using unbalanced (but fixed) allocations
pM <- powerMM(models, doses, base = 0, maxEff = 0.4, sigma = 1,
              lower = 10, upper = 100, step = 20, scal = 200,
              alRatio = c(3, 2, 2, 1, 1, 1), typeN = "arm")
plot(pM, summ = "mean")

## example, where means are directly specified
## doses
dvec <- c(0, 10, 50, 100)
## mean vectors
mu1 <- c(1, 2, 2, 2)
mu2 <- c(1, 1, 2, 2)
mu3 <- c(1, 1, 1, 2)
mMat <- cbind(mu1, mu2, mu3)
dimnames(mMat)[[1]] <- dvec
pM <- powerMM(muMat = mMat, doses = dvec, sigma = 2, lower = 10,
              upper = 100, step = 20)
pM

## End(Not run)

```

powerScenario

Calculates the power for an planMM object under a particular alternative scenario

Description

Given a calculated planMM object (containing the selected candidate models set and sample size) the powerScenario function calculates the power under a particular alternative scenario, ie an alternative mean vector (not included in the candidate set) and an alternative standard deviation.

The function `powerMM` calculates the power to detect the different models within in the candidate set for different sample sizes.

Usage

```

powerScenario(planMMObj, muAlt = NULL, sigmaAlt = NULL,
              rowNames = NULL, colNames = NULL,
              control = mvtnorm.control())

```

Arguments

`planMMObj` A planMM object with calculated critical value. Be sure you selected right direction for the trend in planMM object.

muAlt	A vector with the alternative mean or a matrix with the alternative mean vectors in the rows. The matrix needs to have as many columns as there are doses (in the planMM object).
sigmaAlt	A vector of alternative sigmas to investigate.
rowNames, colNames	Names for the output matrix, if NULL a default naming convention is used.
control	List of control options for the mvtnorm related functions, see mvtnorm.control for details.

Value

The calculated power.

Author(s)

Bjoern Bornkamp

See Also

[planMM](#), [powCalc](#), [powerMM](#)

Examples

```
doses <- c(0,10,25,50,100,150)
models <- list(linear = NULL, emax = 25,
               logistic = c(50, 10.88111), exponential= 85,
               betaMod=matrix(c(0.33,2.31,1.39,1.39), byrow=TRUE, nrow=2))
pLM <- planMM(models, doses, n = rep(50,6), alpha = 0.05, scal=200)
muAlt <- 0.5*(0:5)/5+1
sigmaAlt <- 2
powerScenario(pLM, muAlt, sigmaAlt)

## now try 2 mean vectors and 3 sigmas
muAlt <- rbind(0.5*(0:5)/5+1, c(1,1.3,1.3,1.3,1.3,1.3))
sigmaAlt <- c(1.5,2,2.5)
powerScenario(pLM, muAlt, sigmaAlt)
```

predict.MCPMod

Predict a MCPMod object.

Description

Predict the dose-response curve of a MCPMod object (in case of model averaging this will be an average of the significant and converged dose-response models)

Usage

```
## S3 method for class 'MCPMod'
predict(object, type = c("fullModel", "EffectCurve"), newdata = NULL,
        doseSeq = NULL, lenSeq = 101, uGrad = NULL, ...)
```

Arguments

object	A MCPMod object
type	Predictions for all variables or only the Effect Curve?
newdata	Data frame containing values where to predict when using type="fullModel", if missing use the data, where the model was fitted on.
doseSeq	Numeric specifying doses where to predict in case of type="EffectCurve", if missing use lenSeq equally spaced values between smallest and largest dose in the data.
lenSeq	If doseSeq is not specified a equally spaced grid of "lenSeq" values between placebo and the maximal dose in the study is used.
uGrad	Function to return the gradient of a user defined model, see Examples of the fitDRModel function.
...	Additional arguments

Value

A numeric consisting out of the predictions.

Author(s)

Bjoern Bornkamp

See Also

[fitDRModel](#)

Examples

```
data(IBScovars)
models <- list(emax = 0.2, quadratic = -0.2, linlog = NULL)
dfe <- MCPMod(resp ~ dose, IBScovars, models, addCovars = ~gender,
             alpha = 0.05, pVal = TRUE,
             selModel = "aveAIC", clinRel = 0.25, off = 1)
# predict only effect curve
predict(dfe, type = "EffectCurve", doseSeq = 0:4)

# predict full model, specify where to predict via newdata
preddat <- data.frame(dose = 0:4, gender = as.factor(rep(1, 5)))
predict(dfe, type = "fullModel", newdata = preddat)
```

rndDesign	<i>Round a continuous design to integer values.</i>
-----------	-----------------------------------------------------

Description

Round a continuous design to integer values, using the rounding technique described in Pukelsheim (1993)

Usage

```
rndDesign(w, N, eps = 0.0001)
```

Arguments

w	Vector of design weights or an object of class "design".
N	Total sample size.
eps	Value under which elements of w will be regarded as 0.

Author(s)

Bjoern Bornkamp

References

Pukelsheim, F. (1993). Optimal Design of Experiments, Wiley, Chapter 12.

Examples

```
# breaks ties at random  
rndDesign(rep(1/4, 4), 21)
```

sampSize	<i>Sample size calculations for MCPMod</i>
----------	--------------------------------------------

Description

Given a candidate set, the baseline effect, the maximum effect and the standard deviation, the `sampSize` function returns the smallest sample size achieving a certain combined power value for the associated multiple contrast test. See Pinheiro et al. (2006) for details.

Usage

```
sampSize(models, doses, base, maxEff, sigma, upperN,
         lowerN = floor(upperN/2), power = 0.8, alRatio = NULL,
         sumFct = mean, off = 0.1*max(doses), scal = 1.2 * max(doses),
         alpha = 0.025, alternative = c("one.sided", "two.sided"),
         tol = 0.001, verbose = FALSE,
         control = mvtnorm.control(), muMat = NULL,
         typeN = c("arm", "total"), vCov = NULL, ...)
```

Arguments

models	A list specifying the candidate models. This can also be a fullMod object, then the arguments base, maxEff, off and scal are ignored
doses	Dose levels to be administered
base	Expected placebo effect
maxEff	Expected maximum change from placebo (in considered dose range)
sigma	Expected standard deviation
upperN, lowerN	Upper and lower bound for the target sample size. lowerN defaults to floor(upperN/2).
power	Desired combined power value, defaults to 0.8.
alRatio	Vector describing the relative patient allocations to the dose groups. See Examples below.
sumFct	A function to combine the power values under the different models into one value. By default the arithmetic mean is used.
off	Offset parameter for the linear in log model (default 10 perc. of maximum dose).
scal	Scale parameter for the beta model (default 20 perc. larger than maximum dose).
alpha	Level of significance (default: 0.025)
alternative	Character determining the type of alternative used for the multiple contrast trend test. Argument is ignored when muMat is specified.
tol	A positive numeric value specifying the tolerance level for the bisection search algorithm.
verbose	Logical value indicating if a trace of the iteration progress of the bisection search algorithm should be displayed.
control	A list of options for the pmvt and qmvt functions as produced by mvtnorm.control
muMat	An optional matrix with means as columns and given dimnames (dose levels and names of contrasts). If specified the the models argument should not be specified, see examples below.
typeN	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. See examples below.
vCov	Covariance matrix of mean vectors assumed for calculation of optimal contrasts.
...	Possible additional arguments for sumFct

Details

Calculates the sample size necessary to achieve a desired combined power value for the multiple contrast test. A summary function is used to combine the individual power values. The allocation ratios for the dose groups need to be predefined and fixed (by default balanced allocations are assumed).

The function implements a simple bisection search algorithm to determine the target sample size. In case the upper and lower bound (upperN, lowerN) do not contain the target sample size the algorithm automatically adjusts these boundaries, but outputs a warning message.

Value

An object of class `sampSize`, with the following components:

`samp.size` Vector of target sample size(s)
`approx.power` Combined Power achieved under the assumed scenario and sample size.

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., 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

[MCPtest](#), [powCalc](#), [powerMM](#)

Examples

```
## example from JBS paper p.651
doses <- c(0,10,25,50,100,150)
models <- list(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))
sampSize(models, doses, base = 0, maxEff = 0.4, sigma = 1,
          upperN = 80, scal = 200, alpha = 0.05)
## with different summary function
## Not run:
sampSize(models, doses, base = 0, maxEff = 0.4, sigma = 1,
          upperN = 90, scal = 200, sumFct = median, alpha = 0.05)

## End(Not run)
## with unbalanced allocations (twice as many patients in placebo group
## than in active dose groups)
sampSize(models, doses, base = 0, maxEff = 0.4, sigma = 1,
          alpha = 0.05, upperN = 80, scal = 200, alRatio=c(2,1,1,1,1,1))
## iterates total sample size instead of sample size in smallest arm
## in this case no big difference
```

```
## Not run:
sampSize(models, doses, base = 0, maxEff = 0.4, sigma = 1,
          alpha = 0.05, upperN = 500, scal = 200, typeN = "total",
          alRatio=c(2,1,1,1,1,1))

## sample size calculation for general matrix of means
dvec <- c(0, 10, 50, 100)
mu1 <- c(1, 2, 2, 2)
mu2 <- c(1, 1, 2, 2)
mu3 <- c(1, 1, 1, 2)
mMat <- cbind(mu1, mu2, mu3)
dimnames(mMat)[[1]] <- dvec
sampSize(muMat = mMat, doses = dvec, sigma = 1,
          alpha = 0.05, upperN = 10, alRatio=c(2,2,1,1))

## End(Not run)
```

Index

- *Topic **datagen**
 - genDFdata, 31
 - *Topic **datasets**
 - biom, 5
 - IBScovars, 45
 - migraine, 60
 - *Topic **design**
 - critVal, 16
 - fullMod, 29
 - getPars, 35
 - LP, 46
 - planMM, 62
 - powCalc, 70
 - powerMM, 72
 - sampSize, 77
 - *Topic **hplot**
 - plot.fullMod, 64
 - plot.LP, 64
 - plot.MCPMod, 65
 - plot.planMM, 66
 - plot.powerMM, 67
 - plotModels, 69
 - *Topic **htest**
 - critVal, 16
 - MCPMod, 48
 - MCPtest, 55
 - *Topic **methods**
 - plot.fullMod, 64
 - plot.LP, 64
 - plot.MCPMod, 65
 - plot.planMM, 66
 - plot.powerMM, 67
 - *Topic **misc**
 - AIC.DRMod, 4
 - DRMod and gDRMod methods, 21
 - ED.DRMod, 24
 - getBnds, 32
 - getGrad, 33
 - getInit, 34
 - modelMeans, 60
 - mvtnorm.control, 61
 - predict.MCPMod, 75
 - *Topic **models**
 - bootMCPMod, 6
 - calcBayesEst, 7
 - calcCrit, 10
 - calcOptDesign, 12
 - DoseFinding-package, 2
 - DR-Models, 18
 - fit.control, 25
 - fitDRModel, 26
 - getPars, 35
 - getUpdDesign, 36
 - gFitDRModel, 38
 - guesst, 43
 - MCPMod, 48
 - MED.DRMod, 58
 - powerScenario, 74
 - rndDesign, 77
 - *Topic **model**
 - gMCPtest, 40
 - *Topic **nonlinear**
 - DoseFinding-package, 2
 - *Topic **package**
 - DoseFinding-package, 2
 - *Topic **regression**
 - DoseFinding-package, 2
 - fitDRModel, 26
- AIC.DRMod, 4, 28
- betaMod, 27, 28, 32, 39, 44, 49, 53
- betaMod (DR-Models), 18
- betaModGrad (DR-Models), 18
- biom, 5
- bootMCPMod, 6
- calcBayesEst, 7, 36, 37
- calcCrit, 10, 15

- calcOptDesign, [11](#), [12](#), [33](#), [36](#)
- coef.DRMod, [27](#)
- coef.DRMod (DRMod and gDRMod methods), [21](#)
- coef.gDRMod, [39](#)
- coef.gDRMod (DRMod and gDRMod methods), [21](#)
- critVal, [16](#), [57](#), [63](#)

- DoseFinding (DoseFinding-package), [2](#)
- DoseFinding-package, [2](#)
- DR-Models, [31](#)
- DR-Models, [18](#), [26](#), [38](#), [52](#), [56](#)
- DRMod and gDRMod methods, [21](#)

- ED, [59](#)
- ED (ED.DRMod), [24](#)
- ED.DRMod, [24](#), [28](#), [48](#)
- emax, [28](#), [39](#), [44](#), [53](#)
- emax (DR-Models), [18](#)
- emaxGrad (DR-Models), [18](#)
- exponential, [28](#), [39](#), [44](#), [53](#)
- exponential (DR-Models), [18](#)
- exponentialGrad (DR-Models), [18](#)

- fit.control, [6](#), [25](#), [51](#)
- fitDRModel, [5](#), [20](#), [23](#), [25](#), [26](#), [32–34](#), [39](#), [48](#), [51–53](#), [59](#), [76](#)
- fullMod, [11](#), [13](#), [29](#), [35](#), [64](#), [69](#)

- genDFdata, [31](#)
- getBnds, [9](#), [27](#), [32](#), [38](#), [51](#), [53](#)
- getGrad, [33](#)
- getInit, [34](#)
- getPars, [30](#), [35](#)
- getUpdDesign, [36](#)
- gFitDRModel, [20](#), [23](#), [38](#)
- gMCPtest, [40](#)
- guesst, [43](#), [47](#), [69](#)

- IBScovars, [45](#)
- intervals.DRMod, [33](#)
- intervals.DRMod (DRMod and gDRMod methods), [21](#)
- intervals.gDRMod (DRMod and gDRMod methods), [21](#)

- linear, [28](#), [39](#), [53](#)
- linear (DR-Models), [18](#)
- linearGrad (DR-Models), [18](#)

- linlog, [28](#), [39](#), [49](#), [53](#)
- linlog (DR-Models), [18](#)
- linlogGrad (DR-Models), [18](#)
- logistic, [28](#), [32](#), [39](#), [44](#), [53](#)
- logistic (DR-Models), [18](#)
- logisticGrad (DR-Models), [18](#)
- logLik.DRMod (AIC.DRMod), [4](#)
- LP, [30](#), [46](#), [65](#), [71](#)

- MCPMod, [6](#), [8](#), [48](#), [56](#), [66](#)
- MCPtest, [17](#), [42](#), [48](#), [51](#), [53](#), [55](#), [62](#), [63](#), [79](#)
- MED (MED.DRMod), [58](#)
- MED.DRMod, [25](#), [28](#), [48](#), [58](#)
- migraine, [60](#)
- modelMeans, [60](#)
- mvtnorm.control, [17](#), [53](#), [61](#), [75](#)

- nlminb, [13](#), [25–27](#), [39](#)
- nls, [26](#)
- nls.control, [25](#), [26](#)

- optim, [13](#)
- optimize, [25–27](#)

- planMM, [17](#), [62](#), [67](#), [71](#), [75](#)
- plot.DRMod, [28](#), [65](#), [66](#)
- plot.DRMod (DRMod and gDRMod methods), [21](#)
- plot.fullMod, [64](#)
- plot.gDRMod, [39](#)
- plot.gDRMod (DRMod and gDRMod methods), [21](#)
- plot.LP, [47](#), [64](#)
- plot.MCPMod, [53](#), [65](#)
- plot.planMM, [66](#)
- plot.powerMM, [67](#), [73](#)
- plotModels, [30](#), [44](#), [64](#), [69](#)
- pmvt, [61](#)
- powCalc, [70](#), [73](#), [75](#), [79](#)
- powerMM, [30](#), [68](#), [71](#), [72](#), [74](#), [75](#), [79](#)
- powerScenario, [71–73](#), [74](#)
- predict.DRMod, [28](#), [33](#)
- predict.DRMod (DRMod and gDRMod methods), [21](#)
- predict.gDRMod, [39](#)
- predict.gDRMod (DRMod and gDRMod methods), [21](#)
- predict.MCPMod, [6](#), [53](#), [75](#)
- print.LP (LP), [46](#)

`print.MCPMod` (MCPMod), 48
`print.planMM` (planMM), 62
`print.sampSize` (sampSize), 77
`print.summary.MCPMod` (MCPMod), 48

`qmvt`, 61

quadratic, 28, 39, 44, 53
quadratic (DR-Models), 18
quadraticGrad (DR-Models), 18

`rndDesign`, 77

`sampSize`, 30, 71, 77
`sigEmax`, 28, 32, 39, 44, 53
`sigEmax` (DR-Models), 18
`sigEmaxGrad` (DR-Models), 18
`summary.MCPMod` (MCPMod), 48

`vcov.DRMod`, 33
`vcov.DRMod` (DRMod and gDRMod methods),
21
`vcov.gDRMod`, 39
`vcov.gDRMod` (DRMod and gDRMod methods),
21