Package 'MCPModGeneral'

August 29, 2024

Title A Supplement to the 'DoseFinding' Package for the General Case

Version 0.1-3

Description Analyzes non-normal data via the Multiple Comparison Procedures and Modeling approach (MCP-Mod). Many functions rely on the 'DoseFinding' package. This package makes it so the user does not need to provide or calculate the mu vector and S matrix. Instead, the user typically supplies the data in its raw form, and this package will calculate the needed objects and passes them into the 'DoseFinding' functions. If the user wishes to primarily use the functions provided in the 'DoseFinding' package, a singular function (prepareGen()) will provide mu and S. The package currently handles power analysis and the MCP-Mod procedure for negative binomial, Poisson, and binomial data. The MCP-Mod procedure can also be applied to survival data, but power analysis is not available.
Bretz, F., Pinheiro, J. C., and Branson, M. (2005) <doi:10.1111/j.1541-0420.2005.00344.x>.
Buckland, S. T., Burnham, K. P. and Augustin, N. H. (1997) <doi:10.2307/2533961>.
Pinheiro, J. C., Bornkamp, B., Glimm, E. and Bretz, F. (2014) <doi:10.1002/sim.6052>.

License GPL-3

Encoding UTF-8

Depends DoseFinding

Imports stats, MASS, mvtnorm

RoxygenNote 7.3.2

Suggests testthat (>= 2.1.0), knitr, rmarkdown, survival

VignetteBuilder knitr

NeedsCompilation no

Author Ian Laga [aut, cre],
Francis Boateng [aut]

Maintainer Ian Laga <ilaga25@gmail.com>

Repository CRAN

Date/Publication 2024-08-29 16:50:04 UTC

Contents

`MCPModGeneral-package` *A Supplement to the* `DoseFinding` *Package for the General Case*

Description

Analyzes non-normal data via the Multiple Comparison Procedures and Modeling approach ('MCP-Mod'). Many functions rely on the 'DoseFinding' package. This package makes it so the user does not need to prespecify or calculate the μ vector and S matrix. Instead, the user typically supplies the data in its raw form, and this package will calculate the needed objects and pass them into the 'DoseFinding' functions. If the user wishes to primarily use the functions provided in the 'DoseFinding' package, a singular function ('prepareGen') will provide mu and S. The package currently handles power analysis and the 'MCP-Mod' procedure for negative binomial, Poisson, and binomial data. The 'MCP-Mod' procedure can also be applied to survival data, but power analysis is not available.

Details

Package:	MCPModGeneral
Type:	Package
Version:	0.1-1
Date:	2020-2-9
License:	GPL-3

The main functions are:
`prepareGen`: Calculates the μ vector and S matrix to be supplied to regular `MCPMod` functions (e.g. `MCPMod`, `MCTtest`, `planMod`)
`MCPModGen`: Implements the full `MCPMod` procedure for raw negative binomial and binary data.
`planModPrepare`: Calculate the theoretical covariance matrix S.
`powMCTGen`: Calculates the power of the multiple contrast test.
`sampSizeMCTGen`: Calculates the sample size needed to reach the target power.

The secondary functions are:
`MCPModSurv`: Implements the full `MCPMod` procedure for basic survival data. This includes a Cox-PH model and parametric survival regression models. Power analysis is not available for the survival data.
`simS`: A simulation based method for estimating the theoretical covariance matrices.

Author(s)

Ian Laga

Maintainer: Ian Laga <ilaga25@gmail.com>

References

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

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

Examples

```
# Analyze the binary migraine data from the DoseFinding package.
data(migraine)
models = Mods(linear = NULL, emax = 1, quadratic = c(-0.004),
              doses = migraine$dose)

powMCTGen(migraine$ntrt, "binomial", "logit",
          Ntype = "actual", altModels = models)
sampSizeMCTGen("binomial", "logit", altModels = models, power = 0.8,
               Ntype = "arm", upperN = 30, verbose = TRUE)

# Now analyze using binomial weights
PFrate <- migraine$painfree/migraine$ntrt
migraine$pfrat = migraine$painfree / migraine$ntrt
MCPModGen("binomial","logit",returnS = TRUE, w = "ntrt", dose = "dose",
   resp = "pfrat", data = migraine, models = models, selModel = "aveAIC",
   Delta = 0.2)
```

MCPModGen *MCPModGen - Multiple Comparison and Modeling (General Case)*

Description

This function allows the user to implement the MCPMod function on negative binomial, Poisson, and binary data, without having to write any additional code. If analyzing survival data, see the MCPModSurv function.

Usage

```
MCPModGen(
  family = c("negative binomial", "binomial"),
 link = c("log", "logit", "probit", "cauchit", "cloglog", "identity", "log risk ratio",
    "risk ratio"),
 returnS = FALSE,
 w = NULL,
```

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

Arguments

family	A character string containing the error distribution to be used in the model.
link	A character string for the model link function.
returnS	Logical determining whether muHat and SHat should be returned, in additional to the MCPMod output.
w	Either a numeric vector of the same length as dose and resp, or a character vector denoting the column name in the data.
dose, resp	Either vectors of equal length specifying dose and response values, or character vectors specifying the names of variables in the data frame specified in data.
data	Data frame with names specified in ‘dose‘, ‘resp‘, and optionally ‘w‘. If data is not specified, it is assumed that ‘dose‘ and ‘resp‘ are numerical vectors
models	An object of class ‘"Mods"’, see Mods for details
addCovars	Formula specifying additive linear covariates (e.g. ‘~ factor(gender)‘).
placAdj	Logical specifying whether the provided by ‘resp‘ are to be treated as placebo-adjusted estimates.
selModel	Optional character vector specifying the model selection criterion for dose estimation. Possible values are • AIC: Selects model with smallest AIC (this is the default) • maxT: Selects the model corresponding to the largest t-statistic. • aveAIC: Uses a weighted average of the models corresponding to the significant contrasts. The model weights are chosen by the formula: $w_i =$

$\exp(-0.5AIC_i)/\sum_i(\exp(-0.5AIC_i))$ See Buckland et al. (1997) for details.

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

`alpha` Significance level for the multiple contrast test

`df` An optional numeric value specifying the degrees of freedom. Infinite degrees of freedom ('df=Inf', the default), correspond to the multivariate normal distribution.

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

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

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

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

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

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

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

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

`offset` Either a numeric vector of the same length as dose and resp, or a character vector denoting the column name in the data.

`...` Additional arguments to be passed to `glm` or `glm.nb`. This is especially useful when a fitting error is returned. In these cases, it may be useful to supply a `start` vector for the parameters.

Details

This function works by first fitting a glm with the chosen family and link. The μ vector and S matrix are extracted from the glm, and these values are supplied to the MCPMod function, along with all user-defined arguments.
Currently, the function can take the negative binomial and Poisson family with a log, sqrt, identity, risk ratio, and log risk ratio links, or a bernoulli family with a log, logit, probit, cauchit, cloglog-link, identity, risk ratio, and log risk ratio links.

Value

An object of class MCPMod if 'returnS = FALSE'. Otherwise, a list containing an object of class MCPMod, the numeric vector μ, and the numeric matrix S.

References

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

Examples

```
# Analyze the binary migraine data from the DoseFinding package.
data(migraine)
models = Mods(linear = NULL, emax = 1, quadratic = c(-0.004), doses = migraine$dose)

# Now analyze using binomial weights
PFrate <- migraine$painfree/migraine$ntrt
migraine$pfrat = migraine$painfree / migraine$ntrt
MCPModGen("binomial","logit",returnS = TRUE, w = "ntrt", dose = "dose",
   resp = "pfrat", data = migraine, models = models, selModel = "aveAIC",
   Delta = 0.2)
```

`MCPModSurv` *MCPModSurv - Multiple Comparison and Modeling for Coxph and Parametric Survival Models*

Description

This function allows the user to implement the MCPMod function on a Cox proportional hazards regression model and a parametric survival model. The function works very similarly to `MCPModGen`, but is unique enough in terms of the data and the parameters to warrant its own function.

Usage

```
MCPModSurv(
  model = c("coxph", "parametric"),
  dist = NULL,
  returnS = FALSE,
  dose,
  resp,
  status,
  data = NULL,
  models,
  placAdj = FALSE,
  selModel = c("AIC", "maxT", "aveAIC"),
  alpha = 0.025,
  df = NULL,
  critV = NULL,
```

```
  doseType = c("TD", "ED"),
  Delta,
  p,
  pVal = TRUE,
  alternative = c("one.sided", "two.sided"),
  na.action = na.fail,
  mvtcontrol = mvtnorm.control(),
  bnds,
  control = NULL,
  ...
)
```

Arguments

model	A character string containing the survival regression model.
dist	A character string for the distribution, in the case when model is "parametric". Must be one of "weibull", "exponential", "gaussian", "logistic", "lognormal", or "loglogistic".
returnS	Logical determining whether muHat and SHat should be returned, in additional to the MCPMod output.
dose, resp, status	Either character strings specifying the names of the respective columns in the data data frame, or numeric vectors of equal length containing their respective values. status refers to whether an observation was censored or not. If no observations were censored, status should be a vector of 1s.
data	Data frame with names specified in 'dose', 'resp', and optionally 'w'. If data is not specified, it is assumed that 'dose' and 'resp' are numerical vectors
models	An object of class '"Mods"', see Mods for details
placAdj	Logical specifying whether the provided by 'resp' are to be treated as placebo-adjusted estimates.
selModel	Optional character vector specifying the model selection criterion for dose estimation. Possible values are • AIC: Selects model with smallest AIC (this is the default) • maxT: Selects the model corresponding to the largest t-statistic. • aveAIC: Uses a weighted average of the models corresponding to the significant contrasts. The model weights are chosen by the formula: $w_i = \exp(-0.5AIC_i)/\sum_i(\exp(-0.5AIC_i))$ See Buckland et al. (1997) for details. For 'type = "general"' the "gAIC" is used.
alpha	Significance level for the multiple contrast test
df	An optional numeric value specifying the degrees of freedom. Infinite degrees of freedom ('df=Inf', the default), correspond to the multivariate normal distribution.
critV	Supply a pre-calculated critical value. If this argument is NULL, no critical value will be calculated and the test decision is based on the p-values. If 'critV = TRUE' the critical value will be calculated.

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

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

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

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

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

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

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

`...`
: Additional arguments to be passed to coxph or survreg. This is especially useful when a fitting error is returned.

Details

'MCPModSurv' works by making calls to 'coxph', 'survreg', and 'Surv' from the 'survival' package. After retrieving coefficient estimates and the estimated covariance matrix, values are passed into the 'MCPMod' function from the 'DoseFinding' package.

Value

An object of class MCPMod if returnS = FALSE. Otherwise, a list containing an object of class MCPMod, the numeric vector μ, and the numeric matrix S.

`planModPrepare` *Return the S Matrix for a Theoretical DR-Curve*

Description

This function is useful for several DoseFinding functions, but particular for planMod. Given the true dose-response curve at specified doses, this function will calculate and return the S matrix associated with the specified distribution. If an object of class Mods is provided in the models argument, then a list of S matrices will be returned.

Usage

```
planModPrepare(
  nSample,
  family = c("negative binomial", "binomial", "poisson"),
  link = c("log", "logit", "sqrt", "probit", "cauchit", "cloglog", "identity",
    "risk ratio", "log risk ratio"),
  modelPar = NULL,
  theoResp = NULL,
  doses = NULL,
  Ntype = c("arm", "total", "actual"),
  alRatio = NULL,
  placEff = NULL,
  models = NULL,
  verbose = FALSE,
  offset = NULL
)
```

Arguments

nSample	An integer if Ntype is 'arm' or 'total', or a numerical vector of patient allocations for each arm if Ntype is 'actual'.
family	A character string containing the error distribution to be used in the model.
link	A character string for the model link function.
modelPar	A numeric vector containing the additional parameters for the family argument. If the family is negative binomial, the dispersion parameter should be supplied. If the family is binomial, no model parameter should be supplied.
theoResp	A numerical vector of theoretical response values, on the transformed scale (e.g. on the log-scale for the negative binomial family). This should be the same length as the doses argument.
doses	A numerical vector of doses, corresponding to the theoretical response values provided.
Ntype	One of 'arm', 'total', or 'actual'. See documentation for Ntype in powMCT for descriptions of 'arm' and 'total'. For 'actual', the nSample should be a numerical vector containing the actual patient allocation for each dose provided.
alRatio	A numeric vector specifying the ratios between the patient allocation for the specified doses.
placEff	A numeric value of the placebo effect. This is needed only when the link is risk ratio.
models	Instead of supplying a theoretical dose-response curve and doses, an object of class Mods may be provided. The doses will be pulled from this object, along with the responses at the doses.
verbose	A logical specifying whether the patient allocation should be printed, in addition to the results.

`offset` A positive numeric value specifying the offset term for the negative binomial distribution. If offset = NULL (the default), then the offset has no effect. Theoretically, the offset should be a numeric vector the same length as the number of observations, but for planning purposes, it is unlikely to know the individual offsets in advance.

Value

A numeric S matrix, or a list S matrices, for each model is `models`.

Examples

```
dose.vec = c(0, 5, 10, 20, 30, 40)
models.full = Mods(doses = dose.vec, linear = NULL,
      sigEmax = rbind(c(9, 2), c(6, 3)),
      emax = 0.8,
      quadratic = -0.02,
      placEff = 0, maxEff = 2)
planModPrepare(30, 'negative binomial', 'log', 0.3, getResp(models.full)[,3],
      dose.vec, 'arm')
```

---

powMCTGen *Calculate Power for Multiple Contrast Test (General Case)*

Description

Like the powMCT function, this function allows the user to calculate power for a multiple contrast test for a set of specified alternatives for the general case, but without specifying and S matrix. The user supplies the patient allocation, the alternative models, and any parameters needed for the distribution family (e.g. the dispersion parameter for the negative binomial distribution). The function works by calculating the μ and S for each model in the alternative models and supplying the calculated values to the powMCT function from the DoseFinding package, forwarding relevant arguments.

This function also allows a new Ntype, namely 'actual'. If nSample is a vector and Ntype is 'actual', the function interprets nSample to be the exact patient allocation. This is useful for slightly modifying patient allocation and avoiding messy ratios.

Furthermore, the function also accepts a theoResp and doses argument, which together describe the theoretical dose-response relationship. The returned power is the probability of accepting at least one of the models specified in altModels.

Usage

```
powMCTGen(
  nSample,
  family = c("negative binomial", "binomial", "poisson"),
  link = c("log", "logit", "sqrt", "probit", "cauchit", "cloglog", "identity",
    "risk ratio", "log risk ratio"),
  modelPar = NULL,
```

```
  placEff = NULL,
  theoResp = NULL,
  doses = NULL,
  Ntype = c("arm", "total", "actual"),
  alRatio = NULL,
  altModels,
  alpha = 0.025,
  df = NULL,
  critV = TRUE,
  alternative = c("one.sided", "two.sided"),
  verbose = FALSE,
  offset = NULL
)
```

Arguments

nSample	An integer if Ntype is 'arm' or 'total', or a numerical vector of patient allocations for each arm if Ntype is 'actual'.
family	A character string containing the error distribution to be used in the model.
link	A character string for the model link function.
modelPar	A numeric vector containing the additional parameters for the family argument. If the family is negative binomial, the dispersion parameter should be supplied. If the family is binomial, no model parameter should be supplied.
placEff	A numeric value specifying the mean response at the placebo This is required if link = 'risk ratio' and ignored otherwise.
theoResp	A numerical vector of theoretical response values, on the transformed scale (e.g. on the log-scale for the negative binomial family). This should be the same length as the doses argument.
doses	A numerical vector of doses, corresponding to the theoretical response values provided.
Ntype	One of 'arm', 'total', or 'actual'. See documentation for Ntype in powMCT for descriptions of 'arm' and 'total'. For 'actual', the nSample should be a numerical vector containing the actual patient allocation for each dose provided.
alRatio	Vector describing the relative patient allocations to the dose groups up to proportionality, e.g. rep(1, length(doses)) corresponds to balanced allocations.
altModels	An object of class Mods, defining the mean vectors under which the power should be calculated.
alpha	Significance level to use.
df	Degrees of freedom to assume.
critV	Critical value, if equal to TRUE the critical value will be calculated. Otherwise one can directly specify the critical value here.
alternative	Character determining the alternative for the multiple contrast trend test.
verbose	A logical specifying whether the patient allocation should be printed, in addition to the results.

offset — A positive numeric value specifying the offset term for the negative binomial distribution. If offset = 1 (the default), then the offset has no effect. Theoretically, the offset should be a numeric vector the same length as the number of observations, but for planning purposes, it is unlikely to know the individual offsets in advance.

Value

Numeric containing the calculated power values

Examples

```
dose.vec = c(0, 5, 10, 20, 30, 40)
models.full = Mods(doses = dose.vec, linear = NULL,
      sigEmax = rbind(c(9, 2), c(6, 3)),
      emax = 0.8,
      quadratic = -0.02,
      placEff = 0, maxEff = 2)
## Calculate the power using the responses and doses specified in Mods
powMCTGen(30, 'negative binomial', 'log', modelPar = 0.1, Ntype = 'arm',
      alpha = 0.05, altModels = models.full)
## Calculate the power at theoretical dose-response values
powMCTGen(30, 'negative binomial', 'log', modelPar = 0.1,
      theoResp = c(0, 0.01, 0.02, 1, 1.6, 1.8), doses = c(0, 10, 20, 30, 40, 50),
      alpha = 0.05, altModels = models.full)
```

---

prepareGen *Prepare General Data for the MCPMod Function*

Description

This function serves as an alternative for using the MCPModGen function directly for general data. The function returns the estimates for μ and S, which are needed for MCPMod.

Usage

```
prepareGen(
  family = c("negative binomial", "binomial", "poisson"),
 link = c("log", "logit", "probit", "cauchit", "cloglog", "identity", "log risk ratio",
    "risk ratio", "sqrt"),
  w = NULL,
  dose,
  resp,
  data = NULL,
  addCovars = ~1,
  placAdj = FALSE,
  offset = NULL,
  ...
)
```

Arguments

family	A character string containing the error distribution to be used in the model.
link	A character string for the model link function.
w	Either a numeric vector of the same length as dose and resp, or a character vector denoting the column name in the data.
dose, resp	Either vectors of equal length specifying dose and response values, or character vectors specifying the names of variables in the data frame specified in data.
data	Data frame with names specified in 'dose', 'resp', and optionally 'w'. If data is not specified, it is assumed that 'dose' and 'resp' are numerical vectors
addCovars	Formula specifying additive linear covariates (e.g. '~ factor(gender)').
placAdj	Logical specifying whether the provided by 'resp' are to be treated as placebo-adjusted estimates.
offset	Either a numeric vector of the same length as dose and resp, or a character vector denoting the column name in the data.
...	Additional arguments to be passed to glm or glm.nb. This is especially useful when a fitting error is returned. In these cases, it may be useful to supply a start vector for the parameters.

Value

A list containing the μ vector and S matrix.

Examples

```
# Analyze the binary migraine data from the DoseFinding package.
data(migraine)
models = Mods(linear = NULL, emax = 1, quadratic = c(-0.004), doses = migraine$dose)

# Now analyze using binomial weights
PFrate <- migraine$painfree/migraine$ntrt
migraine$pfrat = migraine$painfree / migraine$ntrt
muS = prepareGen("binomial", "logit", w = "ntrt", dose = "dose",
                 resp = "pfrat", data = migraine)
## Look at the elements of muS
muS
MCPMod(muS$data$dose, muS$data$resp, models = models, S = muS$S,
       type = "general", selModel = "aveAIC",Delta = 0.2)
```

---

sampSizeMCTGen *Sample Size Calculations (General Case)*

Description

This function build on the sampSizeMCT function in the DoseFinding package, allowing the procedure to work with the powMCTGen function for the general case.

Usage

```
sampSizeMCTGen(
  family = c("negative binomial", "binomial", "poisson"),
  link = c("log", "logit", "probit", "cauchit", "cloglog", "log risk ratio",
    "risk ratio"),
  modelPar = NULL,
  theoResp = NULL,
  doses = NULL,
  upperN,
  lowerN = floor(upperN/2),
  Ntype = c("arm", "total"),
  alRatio = NULL,
  altModels,
  alpha = 0.025,
  power = 0.8,
  sumFct = c("min", "mean", "max"),
  verbose = FALSE,
  tol = 0.001
)
```

Arguments

family	A character string containing the error distribution to be used in the model.
link	A character string for the model link function.
modelPar	A numeric vector containing the additional parameters for the family argument. If the family is negative binomial, the dispersion parameter should be supplied. If the family is binomial, no model parameter should be supplied.
theoResp	A numerical vector of theoretical response values, on the transformed scale (e.g. on the log-scale for the negative binomial family). This should be the same length as the doses argument.
doses	A numerical vector of doses, corresponding to the theoretical response values provided.
upperN, lowerN	Upper and lower bound for the power sample size. lowerN defaults to floor(upperN/2).
Ntype	One of "arm", "total", or 'actual". See documentation for Ntype in powMCT for descriptions of "arm" and 'total". For "actual", the nSample should be a numerical vector containing the actual patient allocation for each dose provided.
alRatio	A numeric vector specifying the ratios between the patient allocation for the specified doses.
altModels	An object of class Mods, defining the mean vectors under which the power should be calculated.
alpha	A numeric value specifying the significance level
power	A numeric value specifying the power power of sumFct
sumFct	Either an included character vector or a function that combines the power values under the different alternative into one value.

verbose	A logical specifying whether the patient allocation should be printed, in addition to the results.
tol	A positive numeric value specifying the tolerance level for the bisection search algorithm. Bisection is stopped if the targFunc value is within tol of power.

Value

Numeric containing the calculated power values

Examples

```
dose.vec = c(0, 5, 10, 20, 30, 40)
models.full = Mods(doses = dose.vec, linear = NULL,
      sigEmax = rbind(c(9, 2), c(6, 3)),
      emax = 0.8,
      quadratic = -0.02,
      placEff = 0, maxEff = 2)
## Now we can calculate the sample sizes needed in order to achieve a certain power
sampSizeMCTGen("negative binomial", "log", modelPar = 0.1, upperN = 50, Ntype = "arm",
      altModels = models.full, alpha = 0.05, sumFct = "min", power = 0.8)
```

---

simS *Covariance Matrix Simulation*

Description

For non-canonical links, simulating the covariance matrix is sometimes the easiest way to get an estimate of the covariance matrix. Even for the canonical links, simulating the covariance matrix may be desirable, as theoretical covariance matrices are based off of asymptotic properties which may not hold for small sample sizes.

Usage

```
simS(
  doses,
  resp,
  nSample,
  Ntype = c("arm", "total", "actual"),
  nSim = 1000,
  alRatio = NULL,
  family,
  link,
  modelPar = NULL,
  placEff = NULL,
  verbose = FALSE
)
```

Arguments

doses	A numerical vector of doses, corresponding to the theoretical response values provided.
resp	A numeric vector of response values corresponding to the doses. This should be on the link scale (e.g. resp should be on the log-scale if using the log-link).
nSample	An integer if Ntype is "arm" or "total", or a numerical vector of patient allocations for each arm if Ntype is "actual".
Ntype	One of "arm", "total", or 'actual". See documentation for Ntype in powMCT for descriptions of "arm" and 'total". For "actual", the nSample should be a numerical vector containing the actual patient allocation for each dose provided.
nSim	An integer specifying the number of simulations used to estimate the covariance matrix.
alRatio	Vector describing the relative patient allocations to the dose groups up to proportionality, e.g. rep(1, length(doses)) corresponds to balanced allocations.
family	A character string containing the error distribution to be used in the model.
link	A character string specifying the link to be using when modeling the glm.
modelPar	A numeric vector containing the additional parameters for the family argument. If the family is negative binomial, the dispersion parameter should be supplied. If the family is binomial, no model parameter should be supplied.
placEff	A numeric value specifying the mean response at the placebo This is required if link = "risk ratio" or "log risk ratio" and ignored otherwise.
verbose	A logical specifying whether the patient allocation should be printed, in addition to the results.

Value

Numeric containing the estimated covariance matrix.

Examples

```
data(migraine)
models = Mods(linear = NULL, emax = 1, quadratic = c(-0.004), doses = migraine$dose)
simS(migraine$dose, getResp(models)[,1], 30, "arm", 10, family = "binomial",
     link = "logit", verbose = TRUE)
```

Index