Package 'BayesianMCPMod'

May 14, 2026

Title Simulate, Evaluate, and Analyze Dose Finding Trials with Bayesian MCPMod

Version 1.3.2

Description Bayesian MCPMod (Fleischer et al. (2022) <doi:10.1002/pst.2193>) is an innovative method that improves the traditional MCPMod by systematically incorporating historical data, such as previous placebo group data. This package offers functions for simulating, analyzing, and evaluating Bayesian MCPMod trials with normally and binary distributed endpoints. It enables the assessment of trial designs incorporating historical data across various true dose-response relationships and sample sizes. Robust mixture prior distributions, such as those derived with the Meta-Analytic-Predictive approach (Schmidli et al. (2014) <doi:10.1111/biom.12242>), can be specified for each dose group. Resulting mixture posterior distributions are used in the Bayesian Multiple Comparison Procedure and modeling steps. The modeling step also includes a weighted model averaging approach (Pinheiro et al. (2014) <doi:10.1002/sim.6052>). Estimated dose-response relationships can be bootstrapped and visualized.

License Apache License (>= 2)

URL https://boehringer-ingelheim.github.io/BayesianMCPMod/, https://github.com/Boehringer-Ingelheim/BayesianMCPMod

BugReports https://github.com/Boehringer-Ingelheim/BayesianMCPMod/issues

Depends R (>= 4.2)

Imports checkmate, DoseFinding (>= 1.1-1), dplyr, ggplot2, logistf, methods, nloptr, RBesT, stats, tidyr

Suggests clinDR, doFuture, future.apply, kableExtra, knitr, MCPModPack, reactable, rmarkdown, spelling, testthat (>= 3.0.0), tibble

VignetteBuilder knitr

Config/testthat/edition 3

Encoding UTF-8

Language en-US

RoxygenNote 7.3.2

NeedsCompilation no

Author Boehringer Ingelheim Pharma GmbH & Co. KG [cph, fnd],
Stephan Wojciekowski [aut, cre],
Lars Andersen [aut],
Jonas Schick [ctb],
Sebastian Bossert [aut]

Maintainer Stephan Wojciekowski <stephan.wojciekowski@boehringer-ingelheim.com>

Repository CRAN

Date/Publication 2026-05-14 14:00:19 UTC

Contents

assessDesign 2
getBootstrapQuantiles 7
getBootstrapSamples 8
getContr 9
getCritProb 11
getESS 12
getMED 13
getModelFits 14
getPosterior 16
performBayesianMCP 18
performBayesianMCPMod 19
plot.modelFits 21
predict.modelFits 23
simulateData 24

Index **26**

`assessDesign` *assessDesign*

Description

This function performs simulation based trial design evaluations for a set of specified dose-response models

Usage

```
assessDesign(
  n_patients,
  mods,
  prior_list,
  sd = NULL,
  contr = NULL,
  dr_means = NULL,
  data_sim = NULL,
  estimates_sim = NULL,
  n_sim = 1000,
  alpha_crit_val = 0.05,
  modeling = FALSE,
  simple = TRUE,
  avg_fit = TRUE,
  reestimate = FALSE,
  delta = NULL,
  evidence_level = NULL,
  n_bs_samples = 1000,
  med_selection = c("avgFit", "bestFit"),
  probability_scale = FALSE
)
```

Arguments

n_patients	Vector specifying the planned number of patients per dose group. A minimum of 2 patients are required in each group.
mods	An object of class Mods as specified in the DoseFinding package.
prior_list	A prior_list object specifying the utilized prior for the different dose groups
sd	A positive value, specification of assumed sd. Not required if either data_sim or estimates_sim is provided. Also not required in case of binary endpoint. Default NULL
contr	An object of class optContr as created by the DoseFinding::getContr function. Allows specification of a fixed contrasts matrix. Default NULL.
dr_means	A vector, allows specification of individual (not model based) assumed effects per dose group. Default NULL.
data_sim	An optional data frame for custom simulated data. Must follow the data structure as provided by simulateData(). Default NULL.
estimates_sim	An optional named list of 1) list of vectors for the estimated means per dose group (estimates_sim$mu_hats) and 2) of list of matrices for the covariance matrices specifying the (estimated) variabilities (estimates_sim$S_hats). Dimensions of entries must match the number of dose levels. Default NULL.
n_sim	Number of simulations to be performed
alpha_crit_val	(Un-adjusted) Critical value to be used for the MCP testing step. Passed to the getCritProb function for the calculation of adjusted critical values (on the probability scale). Default 0.05.

`modeling` Boolean variable defining whether the Mod part of Bayesian MCP-Mod will be performed in the assessment. More heavy on resources. Default FALSE.

`simple` Boolean variable defining whether simplified fit will be applied, see `?getModelFits`. Set automatically to TRUE if argument `delta` is provided. Passed to `getModelFits` Default TRUE.

`avg_fit` Boolean variable, defining whether an average fit (based on generalized AIC weights) should be performed in addition to the individual models. Default TRUE.

`reestimate` Boolean variable defining whether critical value should be calculated with re-estimated contrasts (see `getCritProb` function for more details). Default FALSE.

`delta` A numeric value for the threshold Delta for the MED assessment. If NULL, no MED assessment is performed. Default NULL.

`evidence_level` A numeric value between 0 and 1 for the evidence level gamma for the MED assessment. Only required for Bayesian MED assessment, see `?getMED` for details. If NULL, MED assessment will be performed on the fitted model according to the argument `med_selection`. Default NULL.

`n_bs_samples` Number of bootstrap samples for the MED assessment if `evidence_level` is provided. Note that more extreme quantiles (i.e., quantiles closer to 0 or 1) tend to require more bootstrap samples to maintain precision. Default 1000.

`med_selection` A string, either `"avgFit"` or `"bestFit"`, for the method of MED selection. Default `"avgFit"`.

`probability_scale`
A boolean to specify if the trial has a continuous or a binary outcome. Setting to TRUE will transform calculations from the logit scale to the probability scale, which can be desirable for a binary outcome. Default FALSE.

Details

For binary endpoints, if separation occurs (for example, when one treatment arm has only responders or only non-responders), we use penalized logistic regression with Firth's correction to prevent convergence issues and obtain more stable estimates. See the references for additional details.

Value

Returns success probabilities for the different assumed dose-response shapes, attributes also includes information around average success rate (across all assumed models) and prior Effective sample size.

References

Heinze, G. & Schemper, M. (2002). A solution to the problem of separation in logistic regression. *Statistics in Medicine*, 21(16), 2409–2419.

Liu et al. (2022). Commentary: analyzing binary data using MCPMod when zero counts are expected. *arXiv*, 2202.08781, https://arxiv.org/abs/2202.08781

Examples

```
n_patients  <- c(40, 60, 60, 60, 60)
dose_levels <- c(0, 0.5, 2, 4, 8)

mods <- DoseFinding::Mods(linear      = NULL,
                          emax        = c(0.5, 1.2),
                          exponential = 2,
                          betaMod     = c(1, 1),
                          doses       = dose_levels,
                          maxEff      = 6)

sd <- 12
prior_list <- list(Ctrl = RBesT::mixnorm(comp1 = c(w = 1, m = 0, s = 12), sigma = 2),
                   DG_1 = RBesT::mixnorm(comp1 = c(w = 1, m = 1, s = 12), sigma = 2),
                   DG_2 = RBesT::mixnorm(comp1 = c(w = 1, m = 1.2, s = 11), sigma = 2) ,
                   DG_3 = RBesT::mixnorm(comp1 = c(w = 1, m = 1.3, s = 11), sigma = 2) ,
                   DG_4 = RBesT::mixnorm(comp1 = c(w = 1, m = 2, s = 13), sigma = 2))

success_probabilities <- assessDesign(
  n_patients  = n_patients,
  mods        = mods,
  prior_list  = prior_list,
  sd          = sd,
  n_sim       = 1e2) # speed up example run time

success_probabilities

## Analysis with custom data
data_sim <- simulateData(
  n_patients        = n_patients,
  dose_levels       = dose_levels,
  sd                = sd,
  mods              = mods,
  n_sim             = 10)

success_probabilities_cd <- assessDesign(
  n_patients  = n_patients,
  mods        = mods,
  prior_list  = prior_list,
  data_sim    = data_sim,
  sd          = sd,
  n_sim       = 1e2) # speed up example run time

success_probabilities_cd

## Analysis with custom dose response relationship
custom_dr_means <- c(1, 2, 3, 4, 5)

success_probs_custom_dr <- assessDesign(
  n_patients  = n_patients,
  mods        = mods,
  prior_list  = prior_list,
```

```
  dr_means    = custom_dr_means,
  sd          = sd,
  n_sim       = 1e2) # speed up example run time

success_probs_custom_dr

## Analysis with custom estimates for means and variabilies
## No simulated data, only simulated model estimates
estimates_sim <- list(mu_hats = replicate(1e2, list(c(1, 2, 3, 4, 5) + rnorm(5, 0, 1))),
                      S_hats  = list(diag(1, 5)))

success_probs_custom_est <- assessDesign(
  n_patients    = n_patients,
  mods          = mods,
  prior_list    = prior_list,
  estimates_sim = estimates_sim)

success_probs_custom_est

if (interactive()) { # takes typically > 5 seconds

# with MED estimation without bootstrapping
# see ?getMED for details

success_probabilities <- assessDesign(
  n_patients     = n_patients,
  mods           = mods,
  prior_list     = prior_list,
  sd             = sd,
  modeling       = TRUE,
  n_sim          = 10, # speed up example run time
  delta          = 3)

  success_probabilities

# with MED estimation with bootstrapping

success_probabilities <- assessDesign(
  n_patients     = n_patients,
  mods           = mods,
  prior_list     = prior_list,
  sd             = sd,
  modeling       = TRUE,
  n_sim          = 10, # speed up example run time
  delta          = 3,
  evidence_level = 0.8)

  success_probabilities

}
```

getBootstrapQuantiles *getBootstrapQuantiles*

Description

A function for the calculation of bootstrapped model predictions. Samples from the posterior distribution are drawn (via the RBesT function rmix()) and for every sample the simplified fitting step (see getModelFits() function) and a prediction is performed. These fits are then used to identify the specified quantiles. This approach can be considered as the Bayesian equivalent of the frequentist bootstrap approach described in O'Quigley et al. (2017). Instead of drawing n bootstrap samples from the sampling distribution of the trial dose-response estimates, here the samples are directly taken from the posterior distribution.

Usage

```
getBootstrapQuantiles(
  model_fits,
  quantiles,
  n_samples = 1000,
  doses = NULL,
  probability_scale = attr(model_fits, "probability_scale")
)
```

Arguments

`model_fits`	An object of class modelFits, i.e. information about fitted models & corresponding model coefficients as well as the posterior distribution that was the basis for the model fitting
`quantiles`	A vector of quantiles that should be evaluated
`n_samples`	Number of samples that should be drawn as basis for the bootstrapped quantiles
`doses`	A vector of doses for which a prediction should be performed. If NULL, the dose levels of the model_fits will be used. Default NULL.
`probability_scale`	A boolean variable to specify if the trial has a continuous or a binary outcome. Setting to TRUE will transform predictions from the logit scale to the probability scale, which can be desirable for a binary outcome. Default `attr(model_fits, "probability_scale")`.

Value

A tibble with columns for model, dose, and bootstrapped samples

References

O'Quigley J, Iasonos A, Bornkamp B. 2017. Handbook of Methods for Designing, Monitoring, and Analyzing Dose-Finding Trials (1st ed.). Chapman and Hall/CRC. doi:10.1201/9781315151984

Examples

```
posterior_list <- list(Ctrl = RBesT::mixnorm(comp1 = c(w = 1, m = 0, s = 1), sigma = 2),
                       DG_1 = RBesT::mixnorm(comp1 = c(w = 1, m = 3, s = 1.2), sigma = 2),
                      DG_2 = RBesT::mixnorm(comp1 = c(w = 1, m = 4, s = 1.5), sigma = 2) ,
                      DG_3 = RBesT::mixnorm(comp1 = c(w = 1, m = 6, s = 1.2), sigma = 2) ,
                     DG_4 = RBesT::mixnorm(comp1 = c(w = 1, m = 6.5, s = 1.1), sigma = 2))
models         <- c("exponential", "linear")
dose_levels    <- c(0, 1, 2, 4, 8)
model_fits     <- getModelFits(models      = models,
                               posterior   = posterior_list,
                               dose_levels = dose_levels,
                               simple      = TRUE)

bs_quantiles <- getBootstrapQuantiles(model_fits = model_fits,
                                      quantiles  = c(0.025, 0.5, 0.8, 0.975),
                                      n_samples  = 10, # speeding up example run time
                                      doses      = c(0, 6, 8))

bs_quantiles
```

`getBootstrapSamples` *getBootstrapSamples*

Description

A function to return bootstrap samples from the fitted dose-response models. Samples from the posterior distribution are drawn (via the RBesT function rmix()) and for every sample the simplified fitting step (see getModelFits() function) and a prediction is performed. These samples are returned by this function. This approach can be considered as the Bayesian equivalent of the frequentist bootstrap approach described in O'Quigley et al. (2017). Instead of drawing n bootstrap samples from the sampling distribution of the trial dose-response estimates, here the samples are directly taken from the posterior distribution.

Usage

```
getBootstrapSamples(
  model_fits,
  n_samples = 1000,
  doses = NULL,
  probability_scale = attr(model_fits, "probability_scale")
)
```

Arguments

`model_fits`	An object of class modelFits, i.e. information about fitted models & corresponding model coefficients as well as the posterior distribution that was the basis for the model fitting
`n_samples`	Number of samples that should be drawn

doses A vector of doses for which a prediction should be performed

probability_scale
A boolean variable to specify if the trial has a continuous or a binary outcome. Setting to TRUE will transform predictions from the logit scale to the probability scale, which can be desirable for a binary outcome. Default `attr(model_fits, "probability_scale")`.

Value

A tibble with columns for sample_id, model, dose, sample, and sample_diff

References

O'Quigley J, Iasonos A, Bornkamp B. 2017. Handbook of Methods for Designing, Monitoring, and Analyzing Dose-Finding Trials (1st ed.). Chapman and Hall/CRC. doi:10.1201/9781315151984

Examples

```
posterior_list <- list(Ctrl = RBesT::mixnorm(comp1 = c(w = 1, m = 0, s = 1), sigma = 2),
                       DG_1 = RBesT::mixnorm(comp1 = c(w = 1, m = 3, s = 1.2), sigma = 2),
                      DG_2 = RBesT::mixnorm(comp1 = c(w = 1, m = 4, s = 1.5), sigma = 2) ,
                      DG_3 = RBesT::mixnorm(comp1 = c(w = 1, m = 6, s = 1.2), sigma = 2) ,
                     DG_4 = RBesT::mixnorm(comp1 = c(w = 1, m = 6.5, s = 1.1), sigma = 2))
models         <- c("exponential", "linear")
dose_levels    <- c(0, 1, 2, 4, 8)
model_fits     <- getModelFits(models      = models,
                               posterior   = posterior_list,
                               dose_levels = dose_levels,
                               simple      = TRUE)

bs_samples <- getBootstrapSamples(model_fits = model_fits,
                                  n_samples  = 10, # speeding up example run time
                                  doses      = c(0, 6, 8))

bs_samples
```

---

getContr *getContr*

Description

This function calculates contrast vectors that are optimal for detecting certain alternatives via applying the function optContr() of the DoseFinding package. Hereby, 4 different options can be distinguished that are automatically executed based on the input that is provided

1. Bayesian approach: If dose_weights and a prior_list are provided an optimized contrasts for the posterior sample size is calculated. In detail, in a first step the dose_weights (typically the number of patients per dose group) and the prior information is combined by calculating for each dose group a posterior effective sample. Based on this posterior effective sample sizes

the allocation ratio is derived, which allows for a calculation on pseudo-optimal contrasts via regular MCPMod are calculated from the regular MCPMod for these specific weights

2. Frequentist approach: If only dose_weights are provided optimal contrast vectors are calculated from the regular MCPMod for these specific weights
3. Bayesian approach + re-estimation: If only a cov_posterior (i.e. variability of the posterior distribution) is provided, pseudo-optimal contrasts based on these posterior weights will be calculated
4. Frequentist approach+re-estimation: If only a cov_new_trial (i.e. the estimated variability of a new trial) is provided, optimal contrast vectors are calculated from the regular MCPMod for this specific covariance matrix.

Usage

```
getContr(
  mods,
  dose_levels,
  dose_weights = NULL,
  prior_list = NULL,
  cov_posterior = NULL,
  cov_new_trial = NULL
)
```

Arguments

mods	An object of class 'Mods' as created by the function 'DoseFinding::Mods()'
dose_levels	Vector containing the different dosage levels.
dose_weights	Vector specifying weights for the different doses. Please note that in case this information is provided together with a prior (i.e. Option 1) is planned these two inputs should be provided on the same scale (e.g. patient numbers). Default NULL
prior_list	A list of objects of class 'normMix' as created with 'RBesT::mixnorm()'. Only required as input for Option 1. Default NULL
cov_posterior	A covariance matrix with information about the variability of the posterior distribution, only required for Option 3. Default NULL
cov_new_trial	A covariance matrix with information about the observed variability, only required for Option 4. Default NULL

Value

An object of class 'optContr' as provided by the function 'DoseFinding::optContr()'.

Examples

```
dose_levels  <- c(0, 0.5, 2, 4, 8)
mods <- DoseFinding::Mods(
  linear      = NULL,
  emax        = c(0.5, 1.2),
```

```
  exponential = 2,
  doses       = dose_levels,
  maxEff      = 6)
cov_posterior <- diag(c(2.8, 3, 2.5, 3.5, 4)^2)

contr_mat <- getContr(
  mods         = mods,
  dose_levels  = dose_levels,
  cov_posterior = cov_posterior)
```

---

getCritProb *getCritProb*

Description

This function calculates multiplicity adjusted critical values. The critical values are calculated in such a way that when using non-informative priors the actual error level for falsely declaring a significant trial in the Bayesian MCPMod is controlled (by the specified alpha level). Hereby optimal contrasts of the frequentist MCPMod are applied and two options can be distinguished

1. Frequentist approach: If only dose_weights are provided optimal contrast vectors are calculated from the regular MCPMod for these specific weights and the corresponding critical value for this set of contrasts is calculated via the critVal() function of the DoseFinding package.
2. Frequentist approach + re-estimation: If only a cov_new_trial (i.e. the covariance matrix of a new trial) is provided, optimal contrast vectors are calculated from the regular MCPMod for this specific matrix. Here as well the critical value for this set of contrasts is calculated via the critVal() function of the DoseFinding package.

Usage

```
getCritProb(
  mods,
  dose_levels,
  dose_weights = NULL,
  cov_new_trial = NULL,
  alpha_crit_val = 0.025
)
```

Arguments

mods	An object of class "Mods" as specified in the DoseFinding package.
dose_levels	Vector containing the different dosage levels.
dose_weights	Vector specifying weights for the different doses, only required for Option i). Default NULL
cov_new_trial	A covariance matrix, only required for Option ii). Default NULL
alpha_crit_val	Significance level. Default set to 0.025.

Value

Multiplicity adjusted critical value on the probability scale.

Examples

```
mods <- DoseFinding::Mods(linear      = NULL,
                          emax        = c(0.5, 1.2),
                          exponential = 2,
                          doses       = c(0, 0.5, 2,4, 8))
dose_levels <- c(0, 0.5, 2, 4, 8)
critVal <- getCritProb(
  mods           = mods,
  dose_weights   = c(50,50,50,50,50), #reflecting the planned sample size
  dose_levels    = dose_levels,
  alpha_crit_val = 0.05)
```

---

getESS *getESS*

Description

This function calculates the effective sample size for every dose group via `RBesT::ess()`.

Usage

```
getESS(post_list, method = c("elir", "moment", "morita"), n_digits = 1, ...)
```

Arguments

`post_list`	A posterior list object, for which the effective sample size for each dose group should be calculated
`method`	A string specifying the method of ESS calculation, see `?RBesT::ess()`.
`n_digits`	An integer for the number of digits the result should be rounded to.
`...`	Optional arguments applicable to specific methods, see `?RBesT::ess()`.

Value

A vector of the effective sample sizes for each dose group

getMED *getMED*

Description

This function provides information on the minimally efficacious dose (MED). The MED evaluation can either be based on the fitted model shapes (model_fits) or on bootstrapped quantiles (bs_quantiles).

Usage

```
getMED(
  delta,
  evidence_level = 0.5,
  dose_levels = NULL,
  model_fits = NULL,
  bs_quantiles = NULL,
  probability_scale = attr(model_fits, "probability_scale")
)
```

Arguments

`delta`	A numeric value for the threshold Delta.
`evidence_level`	A numeric value between 0 and 1 for the evidence level gamma. Used for the bs_quantiles-based evaluation and not used for the model_fits-based evaluation. Default 0.5.
`dose_levels`	A vector of numerics containing the different dosage levels. Default NULL.
`model_fits`	An object of class modelFits as created with getModelFits(). Default NULL.
`bs_quantiles`	A dataframe created with getBootstrapQuantiles(). Default NULL.
`probability_scale`	A boolean variable to specify if the trial has a continuous or a binary outcome. Setting to TRUE will transform predictions from the logit scale to the probability scale, which can be desirable for a binary outcome. Default `attr(model_fits, "probability_scale")`.

Details

The function assumes that the 1st dose group is the control dose group.

The bootstrap approach allows for an MED based on decision rules of the form

$$\widehat{\text{MED}} = \arg\min_{d \in \{d_1,\ldots,d_k\}} \left\{ \Pr\left(f(d,\hat{\theta}) - f(d_1,\hat{\theta}) > \Delta \right) > \gamma \right\}.$$

The model-shape approach takes the point estimate of the model into account.

Value

A matrix with rows for MED reached, MED, and MED index in the vector of dose levels and columns for the dose-response shapes.

Examples

```
posterior_list <- list(Ctrl = RBesT::mixnorm(comp1 = c(w = 1, m = 0, s = 1), sigma = 2),
                       DG_1 = RBesT::mixnorm(comp1 = c(w = 1, m = 3, s = 1.2), sigma = 2),
                      DG_2 = RBesT::mixnorm(comp1 = c(w = 1, m = 4, s = 1.5), sigma = 2) ,
                      DG_3 = RBesT::mixnorm(comp1 = c(w = 1, m = 6, s = 1.2), sigma = 2) ,
                     DG_4 = RBesT::mixnorm(comp1 = c(w = 1, m = 6.5, s = 1.1), sigma = 2))
models         <- c("exponential", "linear")
dose_levels    <- c(0, 1, 2, 4, 8)
model_fits     <- getModelFits(models      = models,
                               posterior   = posterior_list,
                               dose_levels = dose_levels,
                               simple      = TRUE)

# MED based on the model_fit:
getMED(delta = 5, model_fits = model_fits)

# MED based on bootstrapped quantiles
bs_quantiles <- getBootstrapQuantiles(model_fits = model_fits,
                                      quantiles  = c(0.025, 0.2, 0.5, 0.8),
                                      n_samples  = 100) # speeding up example run time

getMED(delta          = 5,
       evidence_level = 0.8,
       bs_quantiles   = bs_quantiles)

# MED on the probability scale
getMED(delta = 0.1, model_fits = model_fits, probability_scale = TRUE)
```

`getModelFits` *getModelFits*

Description

Fits dose-response curves for the specified dose-response models, based on the posterior distributions. For the simplified fit, multivariate normal distributions will be approximated and reduced by one-dimensional normal distributions. For the default case, the Nelder-Mead algorithm is used. In detail, for both approaches the mean vector θ^Y and the covariance Σ of the (mixture) posterior distributions and the corresponding posterior weights $\tilde{\omega}_l$ for $l \in 1, ..., L$ are used as basis For the full fit a GLS estimator is used to minimize the following expression for the respective dose-response models m

$$\hat{\theta}_m = \arg\min_{\theta_m} \sum_{l=1}^{L} \tilde{\omega}_l (\theta_{l_i}^Y - f(dose_i, \hat{\theta}_m))' \Sigma_l^{-1} (\theta_{l_i}^Y - f(dose_i, \hat{\theta}_m))$$

Therefore the function nloptr of the nloptr package is utilized. In the simplified case $L = 1$, as the dimension of the posterior is reduced to 1 first. The generalized AIC values are calculated via the formula

$$gAIC_m = \sum_{l=1}^{L} \tilde{\omega}_l \sum_{i=0}^{K} \frac{1}{\Sigma_{l_{i,i}}} (\theta_{l_i}^Y - f(dose_i, \hat{\theta}_m))^2 + 2p$$

where p denotes the number of estimated parameters and K the number of active dose levels. Here as well for the simplified case the formula reduces to one summand as $L = 1$. Corresponding gAIC based weights for model M are calculated as outlined in Schorning et al. (2016)

$$\Omega_I(M) = \frac{\exp(-0.5 gAIC_M)}{\sum_{m=1}^{Q} \exp(-0.5 gAIC_m)}$$

where Q denotes the number of models included in the averaging procedure.

Usage

```
getModelFits(
  models,
  dose_levels,
  posterior,
  avg_fit = TRUE,
  simple = FALSE,
  probability_scale = FALSE
)
```

Arguments

`models`	A Mods object as created with `DoseFinding::Mods()` or a vector of model names for which a fit will be performed. Implemented model shapes are `"linear"`, `"exponential"`, `"logistic"`, `"emax"`, `"sigEmax"`, `"quadratic"`, and `"betaMod"`.
`dose_levels`	A vector containing the different dosage levels.
`posterior`	A getPosterior object, containing the (multivariate) posterior distribution per dosage level.
`avg_fit`	Boolean variable, defining whether an average fit (based on generalized AIC weights) should be performed in addition to the individual models. Default TRUE.
`simple`	Boolean variable, defining whether simplified fit will be applied. Default FALSE.
`probability_scale`	A boolean variable to specify if the predicted dose-response should be on the logit scale or the probability scale. Setting to TRUE will transform predictions from the logit scale to the probability scale, which can be desirable for a binary outcome. Default FALSE.

Value

An object of class modelFits. A list containing information about the fitted model coefficients, the prediction per dose group as well as maximum effect and generalized AIC (and corresponding weight) per model.

References

Schorning K, Bornkamp B, Bretz F, Dette H. 2016. Model selection versus model averaging in dose finding studies. Stat Med; 35; 4021-4040.

Examples

```
posterior_list <- list(Ctrl = RBesT::mixnorm(comp1 = c(w = 1, m = 0, s = 1), sigma = 2),
                       DG_1 = RBesT::mixnorm(comp1 = c(w = 1, m = 3, s = 1.2), sigma = 2),
                       DG_2 = RBesT::mixnorm(comp1 = c(w = 1, m = 4, s = 1.5), sigma = 2),
                       DG_3 = RBesT::mixnorm(comp1 = c(w = 1, m = 6, s = 1.2), sigma = 2),
                      DG_4 = RBesT::mixnorm(comp1 = c(w = 1, m = 6.5, s = 1.1), sigma = 2))
models         <- c("emax", "exponential", "sigEmax", "linear")
dose_levels    <- c(0, 1, 2, 4, 8)

fit        <- getModelFits(models      = models,
                           posterior   = posterior_list,
                           dose_levels = dose_levels)

fit

fit_simple <- getModelFits(models      = models,
                           posterior   = posterior_list,
                           dose_levels = dose_levels,
                           simple      = TRUE)

fit_simple
```

`getPosterior` *getPosterior*

Description

Either the patient level data or both mu_hat as well as S_hat must to be provided. If patient level data is provided mu_hat and S_hat are calculated within the function using a linear model. This function calculates the posterior distribution. Depending on the input for S_hat this step is either performed for every dose group independently via the RBesT function postmix() or the mvpostmix() function of the DoseFinding package is utilized. In the latter case conjugate posterior mixture of multivariate normals are calculated (DeGroot 1970, Bernardo and Smith 1994)

Usage

```
getPosterior(
  prior_list,
  data = NULL,
  mu_hat = NULL,
  S_hat = NULL,
  calc_ess = FALSE,
  probability_scale = attr(data, "probability_scale")
)
```

Arguments

prior_list	a prior list with information about the prior to be used for every dose group
data	dataframe containing the information of dose and response. Also a simulateData object can be provided. Default NULL.
mu_hat	vector of estimated mean values (per dose group). Default NULL.
S_hat	covariance matrix specifying the (estimated) variability. The variance-covariance matrix should be provided and the dimension of the matrix needs to match the number of dose groups. Default NULL.
calc_ess	boolean variable, indicating whether effective sample size should be calculated. Default FALSE.
probability_scale	A boolean to specify if the trial has a continuous or a binary outcome. Setting to TRUE will transform calculations from the logit scale to the probability scale, which can be desirable for a binary outcome. Default `attr(data, "probability_scale")`.

Details

Kindly note that one can sample from the `posterior_list` with `lapply(posterior_list, RBesT::rmix, n = 10)`.

For binary endpoints, if separation occurs (for example, when one treatment arm has only responders or only non-responders), we use penalized logistic regression with Firth's correction to prevent convergence issues and obtain more stable estimates. See the references for additional details.

Value

posterior_list, a posterior list object is returned with information about (mixture) posterior distribution per dose group (more detailed information about the conjugate posterior in case of covariance input for S_hat is provided in the attributes)

References

BERNARDO, Jl. M., and Smith, AFM (1994). Bayesian Theory. 81.

Heinze, G. & Schemper, M. (2002). A solution to the problem of separation in logistic regression. *Statistics in Medicine*, 21(16), 2409–2419.

Liu et al. (2022). Commentary: analyzing binary data using MCPMod when zero counts are expected. *arXiv*, 2202.08781, https://arxiv.org/abs/2202.08781

Examples

```
prior_list <- list(Ctrl = RBesT::mixnorm(comp1 = c(w = 1, m = 0, s = 5), sigma = 2),
                   DG_1 = RBesT::mixnorm(comp1 = c(w = 1, m = 1, s = 12), sigma = 2),
                   DG_2 = RBesT::mixnorm(comp1 = c(w = 1, m = 1.2, s = 11), sigma = 2) ,
                   DG_3 = RBesT::mixnorm(comp1 = c(w = 1, m = 1.3, s = 11), sigma = 2) ,
                   DG_4 = RBesT::mixnorm(comp1 = c(w = 1, m = 2, s = 13), sigma = 2))

mu_hat <- c(0, 1, 1.5, 2, 2.5)
```

```
S_hat  <- diag(c(5, 4, 6, 7, 8)^2)

posterior_list <- getPosterior(
   prior_list = prior_list,
   mu_hat     = mu_hat,
   S_hat      = S_hat)

summary(posterior_list)
```

---

performBayesianMCP *performBayesianMCP*

Description

Performs Bayesian MCP Test step, as described in Fleischer et al. (2022). Tests for a dose-response effect using a model-based multiple contrast test based on the (provided) posterior distribution. In particular for every dose-response candidate the posterior probability is calculated that the contrast is bigger than 0 (based on the posterior distribution of the dose groups). In order to obtain significant test decision we consider the maximum of the posterior probabilities across the different models. This maximum is compared with a (multiplicity adjusted) critical value (on the probability scale).

Usage

```
performBayesianMCP(posterior_list, contr, crit_prob_adj)
```

Arguments

`posterior_list`	An object derived with getPosterior with information about the (mixture) posterior distribution per dose group
`contr`	An object of class 'optContr' as created by the getContr() function. It contains the contrast matrix to be used for the testing step.
`crit_prob_adj`	A getCritProb object, specifying the critical value to be used for the testing (on the probability scale)

Value

Bayesian MCP test result, with information about p-values for the individual dose-response shapes and overall significance

References

Fleischer F, Bossert S, Deng Q, Loley C, Gierse J. 2022. Bayesian MCPMod. Pharmaceutical Statistics. 21(3): 654-670. doi:10.1002/pst.2193

Examples

```
mods <- DoseFinding::Mods(linear      = NULL,
                          emax        = c(0.5, 1.2),
                          exponential = 2,
                          doses       = c(0, 0.5, 2,4, 8))
dose_levels  <- c(0, 0.5, 2, 4, 8)
sd_posterior <- c(2.8,3,2.5,3.5,4)
contr_mat <- getContr(
  mods          = mods,
  dose_levels   = dose_levels,
  cov_posterior = diag(sd_posterior)^2)
critVal <- getCritProb(
  mods           = mods,
  dose_weights   = c(50, 50, 50, 50, 50), #reflecting the planned sample size
  dose_levels    = dose_levels,
  alpha_crit_val = 0.05)
prior_list <- list(Ctrl = RBesT::mixnorm(comp1 = c(w = 1, m = 0, s = 5), sigma = 2),
                   DG_1 = RBesT::mixnorm(comp1 = c(w = 1, m = 1, s = 12), sigma = 2),
                   DG_2 = RBesT::mixnorm(comp1 = c(w = 1, m = 1.2, s = 11), sigma = 2) ,
                   DG_3 = RBesT::mixnorm(comp1 = c(w = 1, m = 1.3, s = 11), sigma = 2) ,
                   DG_4 = RBesT::mixnorm(comp1 = c(w = 1, m = 2, s = 13), sigma = 2))
mu <- c(0, 1, 1.5, 2, 2.5)
S_hat <- diag(c(5, 4, 6, 7, 8)^2)
posterior_list <- getPosterior(
  prior_list = prior_list,
  mu_hat     = mu,
  S_hat      = S_hat,
  calc_ess   = TRUE)

performBayesianMCP(posterior_list = posterior_list,
                   contr          = contr_mat,
                   crit_prob_adj  = critVal)
```

---

performBayesianMCPMod *performBayesianMCPMod*

Description

Performs Bayesian MCP Test step and modeling in a combined fashion. See performBayesian-MCP() function for MCP Test step and getModelFits() for the modeling step

Usage

```
performBayesianMCPMod(
  posterior_list,
  contr,
  crit_prob_adj,
  simple = FALSE,
```

```
  avg_fit = TRUE,
  delta = NULL,
  evidence_level = NULL,
  med_selection = c("avgFit", "bestFit"),
  n_samples = 1000,
  probability_scale = FALSE
)
```

Arguments

posterior_list	An object of class 'postList' or a list of 'postList' objects as created by getPosterior() containing information about the (mixture) posterior distribution per dose group
contr	An object of class 'optContr' as created by the getContr() function. It contains the contrast matrix to be used for the testing step.
crit_prob_adj	A getCritProb object, specifying the critical value to be used for the testing (on the probability scale).
simple	Boolean variable, defining whether simplified fit will be applied. Passed to the getModelFits() function. Default FALSE.
avg_fit	Boolean variable, defining whether an average fit (based on generalized AIC weights) should be performed in addition to the individual models. Default TRUE.
delta	A numeric value for the threshold Delta for the MED assessment. If NULL, no MED assessment is performed. Default NULL.
evidence_level	A numeric value between 0 and 1 for the evidence level gamma for the MED assessment. Only required for Bayesian MED assessment, see ?getMED for details. Default NULL.
med_selection	A string, either "avgFit" or "bestFit" based on the lowest gAIC, for the method of MED selection. Default "avgFit".
n_samples	A numerical for the number of bootstrapped samples in case the Bayesian MED assessment is performed. Default 1e3.
probability_scale	A boolean to specify if the trial has a continuous or a binary outcome. Setting to TRUE will transform calculations from the logit scale to the probability scale, which can be desirable for a binary outcome. Default FALSE.

Value

Bayesian MCP test result as well as modeling result.

Examples

```
mods <- DoseFinding::Mods(linear      = NULL,
                          emax        = c(0.5, 1.2),
                          exponential = 2,
                          doses       = c(0, 0.5, 2,4, 8))
dose_levels  <- c(0, 0.5, 2, 4, 8)
```

```
sd_posterior <- c(2.8, 3, 2.5, 3.5, 4)
contr_mat <- getContr(
  mods          = mods,
  dose_levels   = dose_levels,
  cov_posterior = diag(sd_posterior)^2)
critVal <- getCritProb(
  mods           = mods,
  dose_weights   = c(50, 50, 50, 50, 50), #reflecting the planned sample size
  dose_levels    = dose_levels,
  alpha_crit_val = 0.6) # unreasonable alpha chosen for this example, rather choose 0.05
prior_list <- list(Ctrl = RBesT::mixnorm(comp1 = c(w = 1, m = 0, s = 5), sigma = 2),
                   DG_1 = RBesT::mixnorm(comp1 = c(w = 1, m = 1, s = 12), sigma = 2),
                   DG_2 = RBesT::mixnorm(comp1 = c(w = 1, m = 1.2, s = 11), sigma = 2) ,
                   DG_3 = RBesT::mixnorm(comp1 = c(w = 1, m = 1.3, s = 11), sigma = 2) ,
                   DG_4 = RBesT::mixnorm(comp1 = c(w = 1, m = 2, s = 13), sigma = 2))
mu <- c(0, 1, 1.5, 2, 2.5)
S_hat <- diag(c(5, 4, 6, 7, 8)^2)
posterior_list <- getPosterior(
  prior_list = prior_list,
  mu_hat     = mu,
  S_hat      = S_hat,
  calc_ess   = TRUE)

performBayesianMCPMod(posterior_list = posterior_list,
                      contr          = contr_mat,
                      crit_prob_adj  = critVal,
                      simple         = FALSE,
                      delta          = 1.1)
```

`plot.modelFits` *plot.modelFits*

Description

Plot function based on the ggplot2 package. Providing visualizations for each model and a average Fit. Black lines show the fitted dose response models and an AIC based average model. Dots indicate the posterior median and vertical lines show corresponding credible intervals (i.e. the variability of the posterior distribution of the respective dose group). To assess the uncertainty of the model fit one can in addition visualize credible bands (default coloring as orange shaded areas). The calculation of these bands is performed via the `getBootstrapQuantiles()` function. The default setting is that these credible bands are not calculated.

Usage

```
## S3 method for class 'modelFits'
plot(
  x,
  probability_scale = attr(x, "probability_scale"),
```

```
  gAIC = TRUE,
  cr_intv = TRUE,
  alpha_CrI = 0.05,
  cr_bands = FALSE,
  alpha_CrB = c(0.05, 0.2, 1),
  n_bs_smpl = 1000,
  acc_color = "orange",
  plot_res = 100,
  plot_diffs = FALSE,
  ...
)
```

Arguments

x	An object of type `modelFits`
probability_scale	A boolean to specify if the trial has a continuous or a binary outcome. Setting to TRUE will transform the output from the logit scale to the probability scale, which can be desirable for a binary outcome. Default `attr(x, "probability_scale")`.
gAIC	Logical value indicating whether gAIC values are shown in the plot. Default TRUE
cr_intv	Logical value indicating whether credible intervals are included in the plot. Default TRUE
alpha_CrI	Numerical value of the width of the credible intervals. Default is set to 0.05 (i.e 95% CI are shown).
cr_bands	Logical value indicating whether bootstrapped based credible bands are shown in the plot. Default FALSE
alpha_CrB	Numerical vector of the width of the credible bands. Default is set to `c(0.05, 0.2, 1)`, i.e, the 95% CB, 80% CB and bootstrapped median are shown.
n_bs_smpl	Number of bootstrap samples being used. Default 1000.
acc_color	Color of the credible bands. Default `"orange"`.
plot_res	Number of plotted doses within the range of the dose levels, i.e., the resolution of the plot. Default 100.
plot_diffs	Logical flag. Plot differences to the control group. Default FALSE.
...	optional parameter to be passed to `plot()`.

Value

A ggplot2 object

Examples

```
posterior_list <- list(Ctrl = RBesT::mixnorm(comp1 = c(w = 1, m = 0, s = 1), sigma = 2),
                       DG_1 = RBesT::mixnorm(comp1 = c(w = 1, m = 3, s = 1.2), sigma = 2),
                      DG_2 = RBesT::mixnorm(comp1 = c(w = 1, m = 4, s = 1.5), sigma = 2) ,
                      DG_3 = RBesT::mixnorm(comp1 = c(w = 1, m = 6, s = 1.2), sigma = 2) ,
```

```
                    DG_4 = RBesT::mixnorm(comp1 = c(w = 1, m = 6.5, s = 1.1), sigma = 2))
models <- c("exponential", "linear", "emax")
dose_levels <- c(0, 1, 2, 4, 8)
model_fits <- getModelFits(models      = models,
                           posterior   = posterior_list,
                           dose_levels = dose_levels,
                           simple      = TRUE)

plot(model_fits)

# plot with credible bands
plot(model_fits,
     cr_bands          = TRUE,
     plot_diffs        = FALSE,
     probability_scale = FALSE,
     n_bs_smpl         = 1e2) # speeding up example run-time
```

`predict.modelFits` *predict.modelFits*

Description

This function performs model predictions based on the provided model and dose specifications

Usage

```
## S3 method for class 'modelFits'
predict(
  object,
  doses = NULL,
  probability_scale = attr(object, "probability_scale"),
  ...
)
```

Arguments

`object`	A modelFits object containing information about the fitted model coefficients
`doses`	A vector specifying the doses for which a prediction should be done
`probability_scale`	A boolean variable to specify if the trial has a continuous or a binary outcome. Setting to TRUE will transform predictions from the logit scale to the probability scale, which can be desirable for a binary outcome. Default FALSE.
`...`	Currently without function

Value

a list with the model predictions for the specified models and doses

Examples

```
posterior_list <- list(Ctrl = RBesT::mixnorm(comp1 = c(w = 1, m = 0, s = 1), sigma = 2),
                       DG_1 = RBesT::mixnorm(comp1 = c(w = 1, m = 3, s = 1.2), sigma = 2),
                      DG_2 = RBesT::mixnorm(comp1 = c(w = 1, m = 4, s = 1.5), sigma = 2) ,
                      DG_3 = RBesT::mixnorm(comp1 = c(w = 1, m = 6, s = 1.2), sigma = 2) ,
                     DG_4 = RBesT::mixnorm(comp1 = c(w = 1, m = 6.5, s = 1.1), sigma = 2))
models         <- c("emax", "exponential", "sigEmax", "linear", "betaMod")
dose_levels    <- c(0, 1, 2, 4, 8)
fit            <- getModelFits(models      = models,
                               posterior   = posterior_list,
                               dose_levels = dose_levels)

predict(fit, doses = c(0, 1, 3, 4, 6, 8))
```

`simulateData` *simulateData*

Description

Function to simulate patient level data for a normally distributed endpoint

Usage

```
simulateData(
  n_patients,
  dose_levels,
  sd = NULL,
  mods = NULL,
  n_sim = 1000,
  true_model = NULL,
  dr_means = NULL,
  probability_scale = FALSE
)
```

Arguments

`n_patients`	Vector containing number of patients as a numerical value per dose-group.
`dose_levels`	Vector containing the different dosage levels.
`sd`	Standard deviation on patient level. Can be NULL if `probability_scale` is TRUE. Default NULL.
`mods`	An object of class "Mods" as specified in the DoseFinding package. Can be NULL if ´dr_means´ is not NULL. Default NULL.
`n_sim`	Number of simulations to be performed, Default is 1000
`true_model`	A character for model name, e.g. "emax". Assumed true underlying model. If NULL, all dose-response models included in the mods input parameter will be used. Default NULL.

dr_means an optional vector, with information about assumed effects per dose group. Default NULL.

probability_scale
A boolean to specify if the trial has a continuous or a binary outcome. Setting to TRUE will transform calculations from the logit scale to the probability scale, which can be desirable for a binary outcome. Default FALSE.

Value

A list object, containing patient level simulated data for all assumed true models. Also providing information about simulation iteration, patient number as well as dosage levels.

Examples

```
models <- DoseFinding::Mods(linear      = NULL,
                            linlog      = NULL,
                            emax        = c(0.5, 1.2),
                            exponential = 2,
                            doses       = c(0, 0.5, 2,4, 8),
                            maxEff      = 6)
dose_levels <- c(0, 0.5, 2, 4, 8)
sd          <- 12
n_patients  <- c(40, 60, 60, 60, 60)

sim_data <- simulateData(n_patients  = n_patients,
                         dose_levels = dose_levels,
                         sd          = sd,
                         mods        = models)

head(sim_data)

# custom response "model" shape
custom_dose_response <- c(1, 2, 3, 4, 5)
sim_data_custom_dr   <- simulateData(n_patients  = n_patients,
                                     dose_levels = dose_levels,
                                     sd          = sd,
                                     dr_means    = custom_dose_response)

head(sim_data_custom_dr)
```

Index

assessDesign, 2

getBootstrapQuantiles, 7
getBootstrapSamples, 8
getContr, 9
getCritProb, 11
getESS, 12
getMED, 13
getModelFits, 14
getPosterior, 16

performBayesianMCP, 18
performBayesianMCPMod, 19
plot.modelFits, 21
predict.modelFits, 23

simulateData, 24