Package: bhmbasket (via r-universe)

May 14, 2026

Title Bayesian Hierarchical Models for Basket Trials

Version 1.1.0

Description Provides functions for the evaluation of basket trial designs with binary endpoints. Operating characteristics of a basket trial design are assessed by simulating trial data according to scenarios, analyzing the data with Bayesian hierarchical models (BHMs), and assessing decision probabilities on stratum and trial-level based on Go / No-go decision making. The package is build for high flexibility regarding decision rules, number of interim analyses, number of strata, and recruitment. The BHMs proposed by Berry et al. (2013) <doi:10.1177/1740774513497539> and Neuenschwander et al. (2016) <doi:10.1002/pst.1730>, as well as a model that combines both approaches are implemented. Functions are provided to implement Bayesian decision rules as for example proposed by Fisch et al. (2015) <doi:10.1177/2168479014533970>. In addition, posterior point estimates (mean/median) and credible intervals for response rates and some model parameters can be calculated. For simulated trial data, bias and mean squared errors of posterior point estimates for response rates can be provided.

License GPL-3

URL https://CRAN.R-project.org/package=bhmbasket, https://github.com/Boehringer-Ingelheim/bhmbasket

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

Depends R (>= 3.6.0)

Imports foreach, doRNG, rjags, checkmate

Suggests doFuture, future, knitr, rmarkdown, RBesT, testthat (>= 3.0.0)

VignetteBuilder knitr

Encoding UTF-8

RoxygenNote 7.3.2

Language en-US

SystemRequirements JAGS 4.x.y (http://mcmc-jags.sourceforge.net)

Config/testthat/edition 3

Config/pak/sysreqs jags

Repository https://boehringer-ingelheim.r-universe.dev

Date/Publication 2026-04-14 11:40:22 UTC

RemoteUrl https://github.com/boehringer-ingelheim/bhmbasket

RemoteRef HEAD

RemoteSha d9d79abcc885c98d84cc7eb2b766988474c8a1f7

Contents

combinePriorParameters *combinePriorParameters*

Description

This function combines prior parameters from different sources and returns them for use in performAnalyses.

Usage

```
combinePriorParameters(list_of_prior_parameters)
```

Arguments

list_of_prior_parameters	A list of items with class prior_parameters_list

Details

This function is intended to combine the prior parameters set with the functions setPriorParametersBerry, setPriorParametersExNex, setPriorParametersExNexAdj, setPriorParametersPooled, or setPriorParametersStratified, in case more than one analysis method should be applied with performAnalyses.

Value

A list with prior parameters of class prior_parameters_list

Author(s)

Stephan Wojciekowski

See Also

performAnalyses setPriorParametersBerry setPriorParametersExNex setPriorParametersExNexAdj setPriorParametersPooled setPriorParametersStratified getPriorParameters getMuVar

Examples

```
prior_parameters_stratified <- setPriorParametersStratified(c(1, 2), c(3, 4))
prior_parameters_berry      <- setPriorParametersBerry(0, 1, 2)

prior_parameters_list       <- combinePriorParameters(
  list(prior_parameters_berry,
       prior_parameters_stratified))
```

`continueRecruitment` *continueRecruitment*

Description

This function continues the recruitment of subjects for a set of scenarios based on the Go / NoGo decisions in the simulated trial outcomes of said scenarios.

Usage

```
continueRecruitment(n_subjects_add_list, decisions_list, method_name = NULL)
```

Arguments

`n_subjects_add_list`	A list that contains for each scenario an integer vector for the number of subjects per cohort to be additionally recruited.
`decisions_list`	A list with decisions per scenario created with `getGoDecisions`
`method_name`	A string for the method name of the analysis the decisions are based on. Can be `NULL` if only one method has been used for analysis, Default: `NULL`

Details

This function is intended to be used for analyses with the following work flow:
`simulateScenarios()` -> `performAnalyses()` -> `getGoDecisions()`->
`continueRecruitment()` -> `performAnalyses()` -> `getGoDecisions()`->
`continueRecruitment()` -> ...

Note that `n_subjects_add_list` takes the additional number of subjects to be recruited, not the overall number of subjects. This way the work flow can be repeated as often as required, which can be useful e.g. for interim analyses.

Value

An object of class `scenario_list` with the scenario data for each specified scenario.

Author(s)

Stephan Wojciekowski

See Also

`simulateScenarios` `performAnalyses` `getGoDecisions`

Examples

```
interim_scenarios <- simulateScenarios(
  n_subjects_list     = list(c(10, 20, 30)),
  response_rates_list = list(rep(0.9, 3)),
  n_trials            = 10)

interim_analyses <- performAnalyses(
  scenario_list       = interim_scenarios,
  target_rates        = rep(0.5, 3),
  n_mcmc_iterations   = 100)

interim_gos <- getGoDecisions(
  analyses_list       = interim_analyses,
  cohort_names        = c("p_1", "p_2", "p_3"),
  evidence_levels     = c(0.5, 0.8, 0.5),
  boundary_rules      = quote(c(x[1] > 0.8, x[2] > 0.6, x[3] > 0.7)))

scenarios_list <- continueRecruitment(
  n_subjects_add_list = list(c(30, 20, 10)),
  decisions_list      = interim_gos,
  method_name         = "exnex_adj")
```

`createTrial` *createTrial*

Description

This function creates an object of class `scenario_list` for a single trial outcome, which can subsequently be analyzed with other functions of `bhmbasket`, e.g. `performAnalyses`

Usage

```
createTrial(n_subjects, n_responders)
```

Arguments

`n_subjects`	A vector of integers for the number of subjects in the trial outcome
`n_responders`	A vector of integers for the number of responders in the trial outcome

Details

This function is a wrapper for `simulateScenarios` with

```
simulateScenarios(
 n_subjects_list     = list(n_subjects),
 response_rates_list = list(n_responders),
 n_trials            = 1)
```

Value

An object of class `scenario_list` with the scenario data for a single trial outcome.

Author(s)

Stephan Wojciekowski

See Also

`simulateScenarios` `performAnalyses`

Examples

```
trial_outcome <- createTrial(n_subjects   = c(10, 20, 30, 40),
                             n_responders = c( 1,  2,  3,  4))
```

`getAverageNSubjects` *getAverageNSubjects*

Description

This function calculates the average number of subjects per scenario.

Usage

```
getAverageNSubjects(scenario_list)
```

Arguments

`scenario_list`	An object of class `scenario_list`, as created with `simulateScenarios` or `continueRecruitment`

Details

This function can be useful to assess decision rules with regard to the average number of subjects across scenarios when performing interim analyses.

Value

A named list of vectors for the average number of subjects in each scenario.

Author(s)

Stephan Wojciekowski

See Also

`simulateScenarios` `continueRecruitment`

Examples

```
interim_scenarios <- simulateScenarios(
  n_subjects_list     = list(c(10, 20, 30)),
  response_rates_list = list(rep(0.9, 3)),
  n_trials            = 10)

interim_analyses <- performAnalyses(
  scenario_list       = interim_scenarios,
  target_rates        = rep(0.5, 3),
  n_mcmc_iterations   = 100)

interim_gos <- getGoDecisions(
  analyses_list       = interim_analyses,
  cohort_names        = c("p_1", "p_2", "p_3"),
  evidence_levels     = c(0.5, 0.8, 0.5),
  boundary_rules      = quote(c(x[1] > 0.8, x[2] > 0.6, x[3] > 0.7)))

scenarios_list <- continueRecruitment(
  n_subjects_add_list = list(c(30, 20, 10)),
  decisions_list      = interim_gos,
  method_name         = "exnex_adj")

getAverageNSubjects(scenarios_list)
```

`getEstimates` *getEstimates*

Description

This function calculates the point estimates and credible intervals per cohort, as well as estimates of the biases and the mean squared errors of the point estimates per cohort.

Usage

```
getEstimates(
  analyses_list,
  add_parameters = NULL,
  point_estimator = "median",
  alpha_level = 0.05
)
```

Arguments

`analyses_list`	An object of class `analysis_list`, as created with `performAnalyses`
`add_parameters`	A vector of strings naming additional parameters from the Bayesian hierarchical models, e.g. `c('mu', 'tau')`. If `NULL`, no additional parameters will be evaluated, Default: `NULL`

`point_estimator`
: A string indicating the type of estimator used for calculation of bias and MSE. Must be one of `'median'` or `'mean'`

`alpha_level`
: A numeric in (0, 1) for the level of the credible interval. Only values corresponding to quantiles saved in `performAnalyses` will work, Default: `0.05`

Details

Bias and MSE will only be calculated for response rate estimates of simulated trials. For additional parameters, bias and MSE will not be calculated.

Possible additional parameters for the Bayesian hierarchical models are `c('mu', 'tau')` for `'berry'`, `'exnex'`, `'exnex_mix'`, `'exnex_adj'`, and `'exnex_adj_mix'`. The ExNex-based models can also access posterior weights. `paste0("w_", seq_len(n_cohorts))`.

Value

A named list of matrices of estimates of response rates and credible intervals. Estimates of bias and MSE are included for response rate estimates of simulated trials.

Author(s)

Stephan Wojciekowski

See Also

`createTrial` `simulateScenarios` `performAnalyses`

Examples

```
scenarios_list <- simulateScenarios(
  n_subjects_list     = list(c(10, 20, 30)),
  response_rates_list = list(c(0.1, 0.2, 3)),
  n_trials            = 10)

analyses_list <- performAnalyses(
  scenario_list       = scenarios_list,
  target_rates        = c(0.1, 0.1, 0.1),
  calc_differences    = matrix(c(3, 2, 2, 1), ncol = 2),
  n_mcmc_iterations   = 100)

getEstimates(analyses_list)
getEstimates(analyses_list   = analyses_list,
             add_parameters  = c("mu", "tau", "w_1", "w_2", "w_3"),
             point_estimator = "mean",
             alpha_level     = 0.1)

outcome <- createTrial(
  n_subjects          = c(10, 20, 30),
  n_responders        = c( 1,  2,  3))

outcome_analysis <- performAnalyses(
```

```
  scenario_list       = outcome,
  target_rates        = c(0.1, 0.1, 0.1),
  n_mcmc_iterations   = 100)

getEstimates(outcome_analysis)
getEstimates(analyses_list  = outcome_analysis,
             add_parameters = c("mu", "w_1", "w_2", "w_3"))
```

---

getGoDecisions *getGoDecisions*

Description

This function applies decision rules to the analyzed trials. The resulting `decision_list` can be further processed with `getGoProbabilities` or `continueRecruitment`.

Usage

```
getGoDecisions(
  analyses_list,
  cohort_names,
  evidence_levels,
  boundary_rules,
  overall_min_gos = 1
)
```

Arguments

`analyses_list`	An object of class `analysis_list`, as created with `performAnalyses`
`cohort_names`	A vector of strings with the names of the cohorts, e.g. `c('p_1', 'p_2')`
`evidence_levels`	A vector of numerics in `(0, 1)` for the posterior probability thresholds for the cohorts. Will be recycled to match the number of methods in the `analyses_list`
`boundary_rules`	A quote of a vector for the boundary rules, `quote(c(...))`, see details. The number of decisions to be taken must match the number of cohorts. Will be recycled to match the number of methods in the `analyses_list`
`overall_min_gos`	A positive integer for the minimum number of cohort-wise go decisions required for an overall go decision Default: 1

Details

This function applies decision rules of the following type to the outcomes of (simulated) basket trials with binary endpoints:

$$P(p_j|data > p_{B,j}) > \gamma,$$

where $p_j|data$ is the posterior response rate of cohort j, $p_{B,j}$ is the response rate boundary of cohort j, and γ is the evidence level. This rule can equivalently be written as

$$q_{1-\gamma,j} > p_{B,j},$$

where $q_{1-\gamma,j}$ is the $1-\gamma$-quantile of the posterior response rate of cohort j.

The arguments `cohort_names` and `evidence_levels` determine $q_{1-\gamma,j}$, where the entries of `cohort_names` and `evidence_levels` are matched corresponding to their order.

The argument `boundary_rules` provides the rules that describe what should happen with the posterior quantiles $q_{1-\gamma,j}$. The first posterior quantile determined by the first items of `cohort_names` and `evidence_levels` is referred to as `x[1]`, the second as `x[2]`, etc. Using the `quote(c(...))`-notation, many different rules can be implemented. A decision rule for only one cohort would be `boundary_rules = quote(c(x[1] > 0.1))`, `cohort_names = 'p_1'`, and `evidence_levels = 0.5`, which implements the rule $P(p_1|data > 0.1) > 0.5$. The number of decisions to be taken must match the number of cohorts, i.e. for each cohort there must be a decision rule in the vector separated by a comma. See the example section for a decision rule for more than one cohort and the example of `negateGoDecisions` for the implementation of a more complex decision rule.

Value

An object of class `decision_list`

Author(s)

Stephan Wojciekowski

See Also

`performAnalyses` `getGoProbabilities` `negateGoDecisions` `continueRecruitment`

Examples

```
scenarios_list <- simulateScenarios(
  n_subjects_list     = list(c(10, 20, 30)),
  response_rates_list = list(c(0.1, 0.1, 0.9)),
  n_trials            = 10)

analyses_list <- performAnalyses(
  scenario_list      = scenarios_list,
  target_rates       = rep(0.5, 3),
  n_mcmc_iterations  = 100)

## Decision rule for more than one cohort
decisions_list <- getGoDecisions(
  analyses_list   = analyses_list,
  cohort_names    = c("p_1", "p_2", "p_3"),
  evidence_levels = c(0.5, 0.5, 0.8),
  boundary_rules  = quote(c(x[1] > 0.7, x[2] < 0.3, x[3] < 0.6)))

## Decision rule for only two of the three cohorts
decisions_list <- getGoDecisions(
```

```
  analyses_list   = analyses_list,
  cohort_names    = c("p_1", "p_3"),
  evidence_levels = c(0.5, 0.8),
  boundary_rules  = quote(c(x[1] > 0.7, TRUE, x[3] < 0.6)),
  overall_min_gos = 2L)

## Different decision rules for each method
## This works the same way for the different evidence_levels
decisions_list <- getGoDecisions(
  analyses_list   = analyses_list,
  cohort_names    = c("p_1", "p_2", "p_3"),
  evidence_levels = c(0.5, 0.5, 0.8),
  boundary_rules  = list(
    quote(c(x[1] > 0.1,  x[2] < 0.5,  x[3] < 0.1)),   # "berry"
    quote(c(x[1] > 0.2,  x[2] < 0.4,  x[3] < 0.2)),   # "exnex"
    quote(c(x[1] > 0.25, x[2] < 0.35, x[3] < 0.25)),  # "exnex_adj"
    quote(c(x[1] > 0.3,  x[2] < 0.3,  x[3] < 0.3)),   # "exnex_adj_mix"
    quote(c(x[1] > 0.35, x[2] < 0.25, x[3] < 0.35)),  # "exnex_mix"
    quote(c(x[1] > 0.4,  x[2] < 0.2,  x[3] < 0.4)),   # "pooled"
    quote(c(x[1] > 0.45, x[2] < 0.15, x[3] < 0.45)),  # "stratified"
    quote(c(x[1] > 0.5,  x[2] < 0.1,  x[3] < 0.5))    # "stratified_mix"
  ))
```

`getGoProbabilities` *getGoProbabilities*

Description

Calculates the Go probabilities for given decisions

Usage

```
getGoProbabilities(go_decisions_list, nogo_decisions_list = NULL)
```

Arguments

`go_decisions_list`
: An object of class `decision_list`, as returned by `getGoDecisions`

`nogo_decisions_list`
: An object of class `decision_list`, as returned by `getGoDecisions`, Default: `NULL`

Details

If only `go_decisions_list` is provided (i.e. `nogo_decisions_list` is `NULL`), only Go probabilities will be calculated. If both `go_decisions_list` and `nogo_decisions_list` are provided, Go, Consider, and NoGo probabilities will be calculated.

Value

A list of matrices of Go (and Consider and NoGo) probabilities

Author(s)

Stephan Wojciekowski

See Also

getGoDecisions

Examples

```
scenarios_list <- simulateScenarios(
  n_subjects_list     = list(c(10, 20)),
  response_rates_list = list(rep(0.9, 2)),
  n_trials            = 10)

analyses_list <- performAnalyses(
  scenario_list       = scenarios_list,
  target_rates        = rep(0.5, 2),
  n_mcmc_iterations   = 100)

go_decisions_list <- getGoDecisions(
  analyses_list       = analyses_list,
  cohort_names        = c("p_1", "p_2"),
  evidence_levels     = c(0.5, 0.8),
  boundary_rules      = quote(c(x[1] > 0.8, x[2] > 0.6)))

nogo_decisions_list <- getGoDecisions(
  analyses_list       = analyses_list,
  cohort_names        = c("p_1", "p_2"),
  evidence_levels     = c(0.5, 0.8),
  boundary_rules      = quote(c(x[1] < 0.5, x[2] < 0.3)))

getGoProbabilities(go_decisions_list)
getGoProbabilities(go_decisions_list, nogo_decisions_list)
```

---

getMuVar *getMuVar*

Description

This function returns the variance of μ that is worth a certain number of subjects for the distribution of the response rates.

Usage

```
getMuVar(response_rate, tau_scale, n_worth = 1)
```

Arguments

response_rate	A numeric for the response rate
tau_scale	A numeric for the scale parameter of the Half-normal distribution of τ
n_worth	An integer for the number of subjects the variance of μ should be worth with regard to the variability of the distribution of the response rate, Default: 1

Details

Calculates the variance mu_var in

$$logit(p) = \theta\ N(\mu, \tau), \mu\ N(mu_mean, mu_var), \tau\ HN(tau_scale),$$

for n_worth number of observations, as in Neuenschwander et al. (2016).

Value

Returns a numeric for the variance of μ

Author(s)

Stephan Wojciekowski

References

Neuenschwander, Beat, et al. "Robust exchangeability designs for early phase clinical trials with multiple strata." *Pharmaceutical statistics* 15.2 (2016): 123-134.

Examples

```
getMuVar(response_rate = 0.3,
         tau_scale     = 1)
getMuVar(response_rate = 0.3,
         tau_scale     = 1,
         n_worth       = 2)
```

---

getPriorParameters *getPriorParameters*

Description

This function provides default prior parameters for the analysis methods that can be used in performAnalyses.

Usage

```
getPriorParameters(
  method_names,
  target_rates,
  n_worth = 1,
  tau_scale = 1,
  w_j = 0.5
)
```

Arguments

method_names	A vector of strings for the names of the methods to be used. Available methods: c("berry", "exnex", "exnex_adj", "exnex_mix", "exnex_adj_mix", "pooled", "stratified",
target_rates	A vector of numerics in (0, 1) for the target rate of each cohort
n_worth	An integer for the number of subjects the variability of the prior should reflect response rate scale, Default: 1
tau_scale	A numeric for the scale parameter of the Half-normal distribution of τ in the methods "berry", "exnex", "exnex"_mix, "exnex_adj" and "exnex_adj_mix", Default: 1
w_j	A numeric in (0, 1) for the weight of the Ex component in the methods "exnex" and "exnex_adj", Default: 0.5

Details

Regarding the default prior parameters for "berry", "exnex", "exnex_mix", "exnex_adj" and "exnex_adj_mix":

- "berry": The mean of μ is set to 0. Its variance is calculated as proposed in "Robust exchangeability designs for early phase clinical trials with multiple strata" (Neuenschwander et al. (2016)) with regard to n_worth. The scale parameter of τ is set to tau_scale.
- "exnex": The weight of the Ex component is set to w_j. For the Ex component: The target rate that results in the greatest variance is determined. The mean of μ is set to that target rate. The variance of μ is calculated as proposed in "Robust exchangeability designs for early phase clinical trials with multiple strata" (Neuenschwander et al. (2016)) with regard to n_worth. The scale parameter of τ is set to tau_scale.

  For the Nex components: The means of μ_j are set to the respective target rates. The variances of τ_j are calculated as proposed in "Robust exchangeability designs for early phase clinical trials with multiple strata" (Neuenschwander et al. (2016)) with regard to n_worth, see also getMuVar.
- "exnex_adj": The weight of the Ex component is set to w_j. For the Ex component: The target rate that results in the greatest variance is determined. The mean of μ is set to 0. The variance of μ is calculated as proposed in "Robust exchangeability designs for early phase clinical trials with multiple strata" (Neuenschwander et al. (2016)) with regard to n_worth, see also getMuVar. The scale parameter of τ is set to tau_scale.

  For the Nex components: The means of μ_j are set to the 0. The variances of τ_j are calculated as proposed in "Robust exchangeability designs for early phase clinical trials with multiple strata" (Neuenschwander et al. (2016)) with regard to n_worth, see also getMuVar.

- "exnex_mix": Uses the same default Ex prior construction as "exnex". The Nex part default parameters are specified as a one-component mixture prior with w_nex = 1, mean_nex = matrix(logit(target_rates), nrow = 1), and sd_nex = matrix(sqrt(getMuVar(target_rates, 0, n_worth)), nrow = 1). This keeps the default mixture representation compatible with the getPriorParameter()'s input.
- "exnex_adj_mix": Uses the same default Ex prior construction as "exnex_adj". The Nex part is specified as a one-component mixture prior with w_nex = 1, mean_nex = matrix(logit(target_rates), nrow = 1), and sd_nex = matrix(sqrt(getMuVar(target_rates, 0, n_worth)), nrow = 1). The Ex component is centered as in "exnex_adj".
- "pooled": The target rate that results in the greatest variance is determined. The scale parameter α is set to that target rate times n_worth. The scale parameter β is set to 1 - that target rate times n_worth.
- "stratified": The scale parameters α_j are set to target_rates * n_worth. The scale parameters β_j are set to (1 - target_rates) * n_worth.
- "stratified_mix": A two-component beta mixture prior is created by default for each cohort. The first component uses a_j = target_rates * n_worth and b_j = (1 - target_rates) * n_worth. The second component is a vague prior with a_j = 1 and b_j = 1. The default mixture weights are c(0.8, 0.2).

Value

A list with prior parameters of class prior_parameters_list

Author(s)

Stephan Wojciekowski

References

Berry, Scott M., et al. "Bayesian hierarchical modeling of patient subpopulations: efficient designs of phase II oncology clinical trials." *Clinical Trials* 10.5 (2013): 720-734.

Neuenschwander, Beat, et al. "Robust exchangeability designs for early phase clinical trials with multiple strata." *Pharmaceutical statistics* 15.2 (2016): 123-134.

See Also

performAnalyses setPriorParametersBerry setPriorParametersExNex setPriorParametersExNexAdj setPriorParametersPooled setPriorParametersStratified setPriorParametersStratifiedMix combinePriorParameters getMuVar

Examples

```
prior_parameters_list <- getPriorParameters(
  method_names = c("berry", "exnex", "exnex_mix", "exnex_adj",
                   "exnex_adj_mix", "pooled", "stratified", "stratified_mix"),
  target_rates = c(0.1, 0.2, 0.3))
```

invLogit *invLogit*

Description

This function returns the inverse logit of the input argument.

Usage

```
invLogit(theta)
```

Arguments

theta	A numeric

Details

This function is an alias for ‘stats::binomial()$linkinv‘

Value

Inverse logit of theta

Author(s)

Stephan Wojciekowski

See Also

family

Examples

```
invLogit(logit(0.3))
invLogit(c(-Inf, 0, Inf))
```

loadAnalyses *loadAnalyses*

Description

This function loads an analysis performed with performAnalyses

Usage

```
loadAnalyses(
  scenario_numbers,
  analysis_numbers = rep(1, length(scenario_numbers)),
  load_path = tempdir()
)
```

Arguments

scenario_numbers	A (vector of) positive integer(s) for the scenario number(s)
analysis_numbers	A (vector of) positive integer(s) for the analysis number(s), Default: rep(1, length(scenario_numbers))
load_path	A string providing a path where the scenarios are being stored, Default: tempfile

Value

Returns an object of class analysis_list

Author(s)

Stephan Wojciekowski

See Also

performAnalyses saveAnalyses tempfile

Examples

```
trial_data <- createTrial(
  n_subjects   = c(10, 20, 30),
  n_responders = c(1, 2, 3))

analysis_list <- performAnalyses(
  scenario_list      = trial_data,
  target_rates       = rep(0.5, 3),
  n_mcmc_iterations  = 100)

save_info     <- saveAnalyses(analysis_list)
```

```
analysis_list <- loadAnalyses(scenario_numbers = save_info$scenario_numbers,
                              analysis_numbers = save_info$analysis_numbers,
                              load_path        = save_info$path)
```

---

loadScenarios *loadScenarios*

Description

This function loads scenarios saved with saveScenarios

Usage

```
loadScenarios(scenario_numbers, load_path = tempdir())
```

Arguments

scenario_numbers	A vector of integers naming the scenario to be loaded
load_path	A string for the directory where the scenarios are being stored, Default: tempfile

Value

Returns an object of class scenario_list

Author(s)

Stephan Wojciekowski

See Also

simulateScenarios saveScenarios tempfile

Examples

```
scenarios_list <- simulateScenarios(
  n_subjects_list     = list(c(10, 20, 30)),
  response_rates_list = list(rep(0.9, 3)),
  n_trials            = 10)

save_info      <- saveScenarios(scenarios_list)
scenarios_list <- loadScenarios(scenario_numbers = save_info$scenario_numbers,
                                load_path        = save_info$path)
```

`logit` *logit*

Description

This function returns the logit of the input argument.

Usage

```
logit(p)
```

Arguments

p	A numeric in (0, 1)

Details

This function is an alias for ‘stats::binomial()$linkfun‘

Value

logit of p

Author(s)

Stephan Wojciekowski

See Also

`family`

Examples

```
logit(invLogit(0.3))
logit(c(0, 0.5, 1))
```

`negateGoDecisions` *negateGoDecisions*

Description

Negates the go decisions derived with `getGoDecisions`.

Usage

```
negateGoDecisions(go_decisions_list, overall_min_nogos = "all")
```

Arguments

`go_decisions_list`	An object of class `decision_list`, as returned by `getGoDecisions`
`overall_min_nogos`	Either a non-negative integer or the string `all` for the minimum number of cohort-level NoGo decisions required for an overall NoGo decision, Default: `all`

Details

This function is intended for implementing decision rules with a consider zone as e.g. proposed in "Bayesian design of proof-of-concept trials" by Fisch et al. (2015). This approach involves two criteria, Significance and Relevance.

- Significance: high evidence that the treatment effect is greater than some smaller value (e.g. treatment effect under H0)
- Relevance: moderate evidence that the treatment effect is greater than some larger value (e.g. treatment effect under a certain alternative)

The decision for a cohort is then taken as follows:

- Go decision: Significance and Relevance
- Consider decision: either Significance, or Relevance, but not both
- NoGo decision: no Significance and no Relevance

In the example below, the following criteria for are implemented for each of the three cohorts:

- Significance: $P(p_j > 0.4) > 0.95$
- Relevance: $P(p_j > 0.8) > 0.5$

Value

A list of NoGo decisions of class `decision_list`

Author(s)

Stephan Wojciekowski

References

Fisch, Roland, et al. "Bayesian design of proof-of-concept trials." *Therapeutic innovation & regulatory science* 49.1 (2015): 155-162.

See Also

getGoDecisions

Examples

```
scenarios_list <- simulateScenarios(
  n_subjects_list     = list(c(10, 20, 30)),
  response_rates_list = list(rep(0.9, 3)),
  n_trials            = 10)

analysis_list <- performAnalyses(
  scenario_list      = scenarios_list,
  target_rates       = rep(0.5, 3),
  n_mcmc_iterations  = 100)

go_decisions_list <- getGoDecisions(
  analyses_list   = analysis_list,
  cohort_names    = c("p_1", "p_2", "p_3",
                      "p_1", "p_2", "p_3"),
  evidence_levels = c(0.5,  0.5,  0.5,
                      0.95, 0.95, 0.95),
  boundary_rules  = quote(c(x[1] > 0.8 & x[4] > 0.4,
                            x[2] > 0.8 & x[5] > 0.4,
                            x[3] > 0.8 & x[6] > 0.4)))

nogo_decisions <- negateGoDecisions(getGoDecisions(
  analyses_list   = analysis_list,
  cohort_names    = c("p_1", "p_2", "p_3",
                      "p_1", "p_2", "p_3"),
  evidence_levels = c(0.5,  0.5,  0.5,
                      0.95, 0.95, 0.95),
  boundary_rules  = quote(c(x[1] > 0.8 | x[4] > 0.4,
                            x[2] > 0.8 | x[5] > 0.4,
                            x[3] > 0.8 | x[6] > 0.4))))

getGoProbabilities(go_decisions_list, nogo_decisions)
```

`performAnalyses` *performAnalyses*

Description

This function performs the analysis of simulated or observed trial data with the specified methods and returns the quantiles of the posterior response rates

Usage

```
performAnalyses(
  scenario_list,
  evidence_levels = c(0.025, 0.05, 0.5, 0.8, 0.9, 0.95, 0.975),
 method_names = c("berry", "exnex", "exnex_mix", "exnex_adj", "exnex_adj_mix", "pooled",
    "stratified", "stratified_mix"),
  target_rates = NULL,
  prior_parameters_list = NULL,
  calc_differences = NULL,
  n_mcmc_iterations = 10000,
  n_cores = 1,
  seed = 1,
  verbose = TRUE
)
```

Arguments

`scenario_list`	An object of class `scenario_list`, as e.g. created with `simulateScenarios`
`evidence_levels`	A vector of numerics in `(0, 1)` for the `1-evidence_levels`-quantiles of the posterior response rates to be saved. Default: `c(0.025, 0.05, 0.5, 0.8, 0.9, 0.95, 0.975)`
`method_names`	A vector of strings for the names of the methods to be used. Must be one of the default values, Default: `c("berry", "exnex", "exnex_mix", "exnex_adj", "exnex_adj_mix", "pooled", "stratified", "stratified_mix")`
`target_rates`	A vector of numerics in `(0, 1)` for the target rates of each cohort, Default: `NULL`
`prior_parameters_list`	An object of class `prior_parameters_list`, as e.g. created with `getPriorParameters`
`calc_differences`	A matrix of positive integers with 2 columns. For each row the differences will be calculated. Also a vector of positive integers can be provided for a single difference. The integers are the numbers for the cohorts to be subtracted from one another. E.g. providing `c(2, 1)` calculates the difference between cohort 2 and cohort 1. If `NULL`, no subtractions are performed, Default: `NULL`
`n_mcmc_iterations`	A positive integer for the number of MCMC iterations, see Details, Default: `10000`. If `n_mcmc_iterations` is present in `.GlobalEnv` and `missing(n_mcmc_iterations)`, the globally available value will be used.
`n_cores`	Argument is deprecated and does nothing as of version 0.9.3. A positive integer for the number of cores for the parallelization, Default: `1`
`seed`	Argument is deprecated and does nothing as of version 0.9.3. A numeric for the random seed, Default: `1`
`verbose`	A logical indicating whether messages should be printed, Default: `TRUE`

Details

This function applies the following analysis models to (simulated) scenarios of class `scenario_list`:

- Bayesian hierarchical model (BHM) proposed by Berry et al. (2013): `"berry"`
- BHM proposed by Neuenschwander et al. (2016): `"exnex"`
- BHM that combines above approaches: `"exnex_adj"`
- Pooled beta-binomial approach: `"pooled"`
- Stratified beta-binomial approach: `"stratified"`
- BHM with mixture prior on the Nex component: `"exnex_mix"`
- Adjusted BHM with mixture prior on the Nex component: `"exnex_adj_mix"`
- Stratified beta-binomial approach with mixture beta prior: `"stratified_mix"`

The posterior distributions of the BHMs are approximated with Markov chain Monte Carlo (MCMC) methods implemented in JAGS. Two independent chains are used with each `n_mcmc_iterations` number of MCMC iterations. The first `floor(n_mcmc_iterations / 3)` number of iterations are discarded as burn-in period. No thinning is applied.

Note that the value for `n_mcmc_iterations` required for a good approximation of the posterior distributions depends on the analysis model, the investigated scenarios, and the use case. The default value might be a good compromise between run-time and approximation for the estimation of decision probabilities, but it should definitively be increased for the analysis of a single trial's outcome.

The analysis models will only be applied to the unique trial realizations across all simulated scenarios. The models can be applied in parallel by registering a parallel backend for the 'foreach' framework, e.g. with `doFuture::registerDoFuture()` and `future::plan(future::multisession)`. The parallelization is nested, so that the resources of a HPC environment can be used efficiently. For more on this topic, kindly see the respective vignette. The tasks that are to be performed in parallel are chunked according to the number of workers determined with `foreach::getDoParWorkers()`.

The JAGS code for the BHM `"exnex"` was taken from Neuenschwander et al. (2016). The JAGS code for the BHM `"exnex_adj"` is based on the JAGS code for `"exnex"`. The JAGS code for the BHM `"exnex_mix"` extends the `"exnex"` model by using a mixture prior for the Nex component. The JAGS code for the BHM `"exnex_adj_mix"` extends the `"exnex_adj"` model by using a mixture prior for the Nex component.

Value

An object of class `analysis_list`.

Author(s)

Stephan Wojciekowski

References

Berry, Scott M., et al. "Bayesian hierarchical modeling of patient subpopulations: efficient designs of phase II oncology clinical trials." *Clinical Trials* 10.5 (2013): 720-734.

Neuenschwander, Beat, et al. "Robust exchangeability designs for early phase clinical trials with multiple strata." *Pharmaceutical statistics* 15.2 (2016): 123-134.

Plummer, Martyn. "JAGS: A program for analysis of Bayesian graphical models using Gibbs sampling." *Proceedings of the 3rd international workshop on distributed statistical computing.* Vol. 124. No. 125.10. 2003.

See Also

`simulateScenarios` `createTrial` `getPriorParameters`

Examples

```
trial_data <- createTrial(
  n_subjects   = c(10, 20, 30),
  n_responders = c(1, 2, 3))

analysis_list <- performAnalyses(
  scenario_list      = trial_data,
  target_rates       = rep(0.5, 3),
  calc_differences   = matrix(c(3, 2, 1, 1), ncol = 2),
  n_mcmc_iterations  = 100)
```

---

saveAnalyses *saveAnalyses*

Description

This function saves an object of class `analysis_list`

Usage

```
saveAnalyses(analyses_list, save_path = tempdir(), analysis_numbers = NULL)
```

Arguments

`analyses_list`	An object of class `analysis_list`, as created with `performAnalyses`
`save_path`	A string for the path where the scenarios are being stored, Default: `tempfile`
`analysis_numbers`	A positive integer naming the analysis number. If `NULL`, the function will look for the number of saved analyses of the scenario in the directory and add 1, Default: `NULL`

Value

A named list of length 3 of vectors with scenario and analysis numbers and the `save_path`

Author(s)

Stephan Wojciekowski

See Also

performAnalyses loadAnalyses tempfile

Examples

```
trial_data <- createTrial(
  n_subjects   = c(10, 20, 30),
  n_responders = c(1, 2, 3))

analysis_list <- performAnalyses(
  scenario_list      = trial_data,
  target_rates       = rep(0.5, 3),
  n_mcmc_iterations  = 100)

save_info     <- saveAnalyses(analysis_list)
analysis_list <- loadAnalyses(scenario_numbers = save_info$scenario_numbers,
                              analysis_numbers = save_info$analysis_numbers,
                              load_path        = save_info$path)
```

---

saveScenarios *saveScenarios*

Description

Saves the scenario data in a newly created or existing directory

Usage

```
saveScenarios(scenario_list, save_path = tempdir())
```

Arguments

scenario_list	An object of class scenario_list, e.g. created with simulateScenarios()
save_path	A string providing the path of the directory in which the scenario data should be saved, Default: tempfile

Value

A named list of length 2 with the scenario numbers and the save_path

Author(s)

Stephan Wojciekowski

See Also

simulateScenarios loadScenarios tempfile

Examples

```
scenarios_list <- simulateScenarios(
  n_subjects_list     = list(c(10, 20, 30)),
  response_rates_list = list(rep(0.9, 3)),
  n_trials            = 10)

save_info      <- saveScenarios(scenarios_list)
scenarios_list <- loadScenarios(scenario_numbers = save_info$scenario_numbers,
                                load_path        = save_info$path)
```

`scaleRoundList` *scaleRoundList*

Description

This function applies scaling and rounding to each item of a list of numerics

Usage

```
scaleRoundList(list, scale_param = 1, round_digits = NULL)
```

Arguments

`list`	The list to which the scaling and rounding should be applied to.
`scale_param`	A numeric for the scaling of each item of the list, Default: ‘1‘
`round_digits`	An integer for the number of digits. If ‘NULL‘, no rounding will be applied, Default: ‘NULL‘

Value

A list of scaled and rounded numerics

Author(s)

Stephan Wojciekowski

Examples

```
some_list <- as.list(runif(5))
scaleRoundList(some_list, scale_param = 100, round_digits = 2)

scenarios_list <- simulateScenarios(
  n_subjects_list     = list(c(10, 20, 30)),
  response_rates_list = list(c(0.1, 0.2, 0.3)),
  n_trials            = 10)

analyses_list <- performAnalyses(
  scenario_list       = scenarios_list,
```

```
  target_rates        = rep(0.5, 3),
  n_mcmc_iterations   = 100)

scaleRoundList(
  list         = getEstimates(analyses_list),
  scale_param  = 100,
  round_digits = 2)
```

`setPriorParametersBerry` *setPriorParametersBerry*

Description

This function sets prior parameters for the analysis method `"berry"` for use in `performAnalyses`.

Usage

```
setPriorParametersBerry(mu_mean, mu_sd, tau_scale)
```

Arguments

`mu_mean`	A numeric for the mean of μ
`mu_sd`	A positive numeric for the standard deviation of μ
`tau_scale`	A positive numeric for the scale parameter of τ

Details

This function sets the prior parameters for the method proposed by Berry et al. (2013). Note that the implemented distribution of τ is half-normal.

Value

A list with prior parameters of class `prior_parameters_list`

Author(s)

Stephan Wojciekowski

References

Berry, Scott M., et al. "Bayesian hierarchical modeling of patient subpopulations: efficient designs of phase II oncology clinical trials." *Clinical Trials* 10.5 (2013): 720-734.

See Also

`performAnalyses` `getPriorParameters` `combinePriorParameters` `setPriorParametersExNex` `setPriorParametersExNexAdj` `setPriorParametersPooled` `setPriorParametersStratified` `getMuVar`

Examples

```
prior_parameters_berry <- setPriorParametersBerry(0, 1, 2)
```

`setPriorParametersExNex` *setPriorParametersExNex*

Description

This function sets prior parameters for the analysis method `"exnex"` for use in `performAnalyses`.

It supports two specifications for the Nex part:

- the standard ExNex specification with cohort-specific Nex priors via `mu_j` and `tau_j`
- an extended specification with a mixture prior on the Nex part via `w_nex`, `mean_nex`, and `sd_nex`

Usage

```
setPriorParametersExNex(
  mu_mean,
  mu_sd,
  tau_scale,
  mu_j = NULL,
  tau_j = NULL,
  w_j,
  w_nex = NULL,
  mean_nex = NULL,
  sd_nex = NULL
)
```

Arguments

`mu_mean`	A numeric for the mean of μ
`mu_sd`	A positive numeric for the standard deviation of μ
`tau_scale`	A positive numeric for the scale parameter of τ
`mu_j`	A vector of numerics for the means μ_j of the standard Nex priors. Ignored if `w_nex`, `mean_nex`, and `sd_nex` are provided.
`tau_j`	A vector of positive numerics for the standard deviations τ_j of the standard Nex priors. Ignored if `w_nex`, `mean_nex`, and `sd_nex` are provided.
`w_j`	A numeric in `(0, 1)` for the weight of the Ex component, or a numeric vector of mixture weights summing to 1 if multiple Ex components are specified.
`w_nex`	An optional numeric vector of mixture weights in $[0, 1]$ summing to 1 for the Nex mixture prior.
`mean_nex`	An optional numeric matrix of Nex mixture means with one row per Nex mixture component and one column per cohort.
`sd_nex`	An optional positive numeric matrix of Nex mixture standard deviations with one row per Nex mixture component and one column per cohort.

Details

This function sets the prior parameters for the method proposed by Neuenschwander et al. (2016). If `w_nex`, `mean_nex`, and `sd_nex` are all `NULL`, the standard ExNex formulation is used. Otherwise, the Nex part is specified as a finite mixture prior.

Value

A list with prior parameters of class `prior_parameters_list`

Author(s)

Stephan Wojciekowski

References

Neuenschwander, Beat, et al. "Robust exchangeability designs for early phase clinical trials with multiple strata." *Pharmaceutical statistics* 15.2 (2016): 123-134.

See Also

`performAnalyses` `getPriorParameters` `combinePriorParameters` `setPriorParametersBerry` `setPriorParametersExNexAdj` `setPriorParametersPooled` `setPriorParametersStratified` `getMuVar`

Examples

```
## standard ExNex
prior_parameters_exnex <- setPriorParametersExNex(
  mu_mean   = 0,
  mu_sd     = 1,
  tau_scale = 2,
  mu_j      = c(4, 5),
  tau_j     = c(6, 7),
  w_j       = 0.8
)

## ExNex with Nex mixture prior
prior_parameters_exnex_mix <- setPriorParametersExNex(
  mu_mean   = 0,
  mu_sd     = 1,
  tau_scale = 2,
  w_j       = 0.8,
  w_nex     = c(0.7, 0.3),
  mean_nex  = rbind(c(4, 5), c(2, 3)),
  sd_nex    = rbind(c(6, 7), c(8, 9))
)
```

setPriorParametersExNexAdj *setPriorParametersExNexAdj*

Description

This function sets prior parameters for the analysis method `"exnex_adj"` for use in `performAnalyses`.

It supports two specifications for the Nex part:

- the standard ExNex Adjusted specification with cohort-specific Nex priors via `mu_j` and `tau_j`
- an extended specification with a mixture prior on the Nex part via `w_nex`, `mean_nex`, and `sd_nex`

Usage

```
setPriorParametersExNexAdj(
  mu_mean,
  mu_sd,
  tau_scale,
  mu_j = NULL,
  tau_j = NULL,
  w_j,
  w_nex = NULL,
  mean_nex = NULL,
  sd_nex = NULL
)
```

Arguments

`mu_mean`	numeric Mean of μ
`mu_sd`	numeric Positive standard deviation of μ
`tau_scale`	numeric Positive scale parameter of τ
`mu_j`	numeric Vector of means μ_j for the standard Nex priors. Ignored if `w_nex`, `mean_nex`, and `sd_nex` are provided.
`tau_j`	numeric Vector of positive standard deviations τ_j for the standard Nex priors. Ignored if `w_nex`, `mean_nex`, and `sd_nex` are provided.
`w_j`	numeric Weight of the Ex component in `(0, 1)`, or a numeric vector of mixture weights summing to 1 if multiple Ex components are specified.
`w_nex`	numeric Optional vector of mixture weights in $[0, 1]$ summing to 1 for the Nex mixture prior.
`mean_nex`	numeric Optional matrix of Nex mixture means with one row per Nex mixture component and one column per cohort.
`sd_nex`	numeric Optional positive matrix of Nex mixture standard deviations with one row per Nex mixture component and one column per cohort.

Details

This function sets prior parameters for the ExNex Adjusted method, which combines the approach proposed by Neuenschwander et al. (2016) and the approach proposed by Berry et al. (2013). If `w_nex`, `mean_nex`, and `sd_nex` are all `NULL`, the standard ExNex Adjusted formulation is used. Otherwise, the Nex part is specified as a finite mixture prior.

Value

A list with prior parameters of class `prior_parameters_list`

Author(s)

Stephan Wojciekowski

References

Neuenschwander, Beat, et al. "Robust exchangeability designs for early phase clinical trials with multiple strata." *Pharmaceutical statistics* 15.2 (2016): 123-134.

See Also

`performAnalyses` `getPriorParameters` `combinePriorParameters` `setPriorParametersBerry` `setPriorParametersExNex` `setPriorParametersPooled` `setPriorParametersStratified` `getMuVar`

Examples

```
## standard ExNex Adjusted
prior_parameters_exnex_adj <- setPriorParametersExNexAdj(
  mu_mean   = 0,
  mu_sd     = 1,
  tau_scale = 2,
  mu_j      = c(4, 5),
  tau_j     = c(6, 7),
  w_j       = 0.8
)

## ExNex Adjusted with Nex mixture prior
prior_parameters_exnex_adj_mix <- setPriorParametersExNexAdj(
  mu_mean   = 0,
  mu_sd     = 1,
  tau_scale = 2,
  w_j       = 0.8,
  w_nex     = c(0.7, 0.3),
  mean_nex  = rbind(c(0, 0), c(-1, -1)),
  sd_nex    = rbind(c(6, 7), c(8, 9))
)
```

setPriorParametersPooled *setPriorParametersPooled*

Description

This function sets prior parameters for the analysis method "pooled" for use in performAnalyses.

Usage

```
setPriorParametersPooled(a, b)
```

Arguments

a	A positive numeric for α
b	A positive numeric for β

Details

The method "pooled" is a beta-binomial model that pools all cohorts. The prior parameters are the scale parameters of the beta prior distribution.

Value

A list with prior parameters of class prior_parameters_list

Author(s)

Stephan Wojciekowski

See Also

performAnalyses getPriorParameters combinePriorParameters setPriorParametersBerry setPriorParametersExNex setPriorParametersExNexAdj setPriorParametersStratified getMuVar

Examples

```
prior_parameters_pooled <- setPriorParametersPooled(1, 2)
```

setPriorParametersStratified
setPriorParametersStratified

Description

This function sets prior parameters for the analysis method "stratified" for use in performAnalyses.

Usage

```
setPriorParametersStratified(a_j, b_j)
```

Arguments

a_j	A vector of positive numerics for α
b_j	A vector of positive numerics for β

Details

The method "stratified" is a beta-binomial model that assesses each cohort individually. The prior parameters are the scale parameters of the beta prior distributions.

Value

A list with prior parameters of class prior_parameters_list

Author(s)

Stephan Wojciekowski

See Also

performAnalyses getPriorParameters combinePriorParameters setPriorParametersBerry setPriorParametersExNex setPriorParametersExNexAdj setPriorParametersPooled getMuVar

Examples

```
prior_parameters_pooled <- setPriorParametersStratified(c(1, 2), c(3, 4))
```

setPriorParametersStratifiedMix *setPriorParametersStratifiedMix*

Description

This function sets prior parameters for the analysis method `"stratified_mix"` for use in `performAnalyses`.

Usage

```
setPriorParametersStratifiedMix(w, a_j, b_j)
```

Arguments

`w`	A numeric vector of mixture weights in $[0, 1]$ summing to 1.
`a_j`	A positive numeric matrix of beta shape parameters α_j, with rows per mixture component and columns per cohort.
`b_j`	A positive numeric matrix of beta shape parameters β_j, with rows per mixture component and columns per cohort.

Details

The method `"stratified_mix"` is a beta-binomial model that assesses each cohort individually with a finite mixture beta prior. See also the R package `RBesT`.

Value

A list with prior parameters of class `prior_parameters_list`

Author(s)

Stephan Wojciekowski

See Also

`performAnalyses` `getPriorParameters` `combinePriorParameters` `setPriorParametersStratified`

Examples

```
prior_parameters_stratified_mix <- setPriorParametersStratifiedMix(
  w   = c(0.8, 0.2),
  a_j = rbind(c(2, 3), c(1, 1)),
  b_j = rbind(c(8, 7), c(1, 1))
)
```

simulateScenarios *simulateScenarios*

Description

This function creates scenarios for the analysis with `performAnalyses`.

Usage

```
simulateScenarios(
  n_subjects_list,
  response_rates_list,
  scenario_numbers = seq_along(response_rates_list),
  n_trials = 10000
)
```

Arguments

`n_subjects_list`	A list that contains for each scenario a vector for the number of subjects per cohort. A single vector can be provided if all scenarios should have the same number of subjects.
`response_rates_list`	A list that contains for each scenario a vector for the response rates per cohort.
`scenario_numbers`	A vector of positive integers naming the scenarios, Default: `seq_along(response_rates_list)`
`n_trials`	An integer indicating the number of trial simulations per response rates, Default: `10000`. If `n_trials` is present in `.GlobalEnv` and `missing(n_trials)`, the globally available value will be used.

Details

The function simulates trials with binary outcome for each scenario. Integer values for the response rates will be treated as observed outcomes.

Value

An object of class `scenario_list` with the scenario data for each specified scenario.

Author(s)

Stephan Wojciekowski

See Also

`saveScenarios` `createTrial` `performAnalyses`

Examples

```
n_subjects     <- c(10, 20, 30)

rr_negative    <- rep(0.1, 3)
rr_nugget      <- c(0.9, 0.1, 0.1)
rr_positive    <- rep(0.9, 3)

scenarios_list <- simulateScenarios(
  n_subjects_list     = list(n_subjects,
                             n_subjects,
                             n_subjects),
  response_rates_list = list(rr_negative,
                             rr_nugget,
                             rr_positive))
```

Index