Package: basksim (via r-universe)

October 9, 2026

Type Package

Title Simulation-Based Calculation of Basket Trial Operating Characteristics

Version 2.2.0

Description Provides a unified syntax for the simulation-based comparison of different single-stage basket trial designs with a binary endpoint and equal sample sizes in all baskets. Methods include the designs by Baumann et al. (2025) <doi:10.1080/19466315.2024.2402275>, Schmitt and Baumann (2025) <doi:10.1080/19466315.2025.2486231>, Fujikawa et al. (2020) <doi:10.1002/bimj.201800404>, Berry et al. (2020) <doi:10.1177/1740774513497539>, and Neuenschwander et al. (2016) <doi:10.1002/pst.1730>. For the latter two designs, the functions are mostly wrappers for functions provided by the package 'bhmbasket'.

License GPL (>= 3)

Encoding UTF-8

RoxygenNote 7.3.3

Imports arrangements, bhmbasket (>= 1.0.0), doFuture, extraDistr, foreach, HDInterval, progressr, purrr, stats

Suggests covr, testthat (>= 3.0.0), ggplot2

Config/testthat/edition 3

URL https://github.com/lbau7/basksim

BugReports https://github.com/lbau7/basksim/issues

NeedsCompilation no

Author Lukas Baumann [aut, cre] (ORCID: <https://orcid.org/0000-0001-7931-7470>), Lukas D Sauer [aut] (ORCID: <https://orcid.org/0000-0002-1340-9994>), Sabrina Schmitt [aut] (ORCID: <https://orcid.org/0009-0008-2719-6513>)

Maintainer Lukas Baumann <baumann@imbi.uni-heidelberg.de>

Repository https://cran.r-universe.dev

Date/Publication 2026-05-12 10:02:23 UTC

RemoteUrl https://github.com/cran/basksim

RemoteRef HEAD

RemoteSha c35bfc470b5f699670812c189373c3d55772d9ae

Contents

adjust_lambda . . . 3
adjust_lambda.bhm . . . 4
adjust_lambda.default . . . 5
adjust_lambda.exnex . . . 7
ecd . . . 8
geom_borrow . . . 9
geom_borrow.fujikawa . . . 10
geom_posterior . . . 11
geom_posterior.fujikawa . . . 12
geom_prior . . . 13
geom_prior.fujikawa . . . 13
get_data . . . 14
get_details . . . 15
get_details.app . . . 16
get_details.bhm . . . 17
get_details.binomial . . . 18
get_details.cpp . . . 19
get_details.cppglobal . . . 20
get_details.cpplim . . . 21
get_details.exnex . . . 22
get_details.fujikawa . . . 24
get_details.jsdglobal . . . 25
get_details.mml . . . 27
get_details.mmlglobal . . . 28
get_evaluation . . . 29
get_evaluation.app . . . 29
get_evaluation.bhm . . . 30
get_evaluation.cpp . . . 31
get_evaluation.cpplim . . . 32
get_evaluation.exnex . . . 33
get_evaluation.fujikawa . . . 34
get_results . . . 35
get_results.app . . . 36
get_results.bhm . . . 37
get_results.cpp . . . 38
get_results.cppglobal . . . 39
get_results.cpplim . . . 40
get_results.exnex . . . 41
get_results.fujikawa . . . 43
get_results.jsdglobal . . . 44

get_results.mml 45
get_results.mmlglobal 46
get_scenarios 47
opt_design 47
setup_app 49
setup_bhm 50
setup_binomial 51
setup_cpp 51
setup_cppglobal 52
setup_cpplim 53
setup_exnex 54
setup_fujikawa 55
setup_jsdglobal 56
setup_mml 57
setup_mmlglobal 58
toer 59

Index **61**

`adjust_lambda` *Adjust Lambda*

Description

Adjust Lambda

Usage

```
adjust_lambda(design, ...)
```

Arguments

`design`	An object created with one of the `setup` functions.
`...`	Further arguments.

Details

The default method for `adjust_lambda` uses a combination of `uniroot` and grid search and calls `toer` in every iteration. For methods implemented in the `bhmbasket` package there are separate methods that are computationally more efficient.

Value

A list containing the greatest estimated value for `lambda` with `prec_digits` decimal places which controls the family wise error rate at level `alpha` (one-sided) and the estimated family wise error rate for the estimated `lambda`.

Examples

```
design <- setup_cpp(k = 3, p0 = 0.2)

# Equal sample sizes
adjust_lambda(design = design, n = 20, alpha = 0.05,
  design_params = list(tune_a = 1, tune_b = 1), iter = 1000)

# Unequal sample sizes
adjust_lambda(design = design, n = c(15, 20, 25), alpha = 0.05,
   design_params = list(tune_a = 1, tune_b = 1), iter = 1000)
```

`adjust_lambda.bhm` *Adjust Lambda for the BHM Design*

Description

Adjust Lambda for the BHM Design

Usage

```
## S3 method for class 'bhm'
adjust_lambda(
  design,
  n,
  p1 = NULL,
  alpha = 0.05,
  design_params = list(),
  iter = 1000,
  n_mcmc = 10000,
  prec_digits = 3,
  data = NULL,
  ...
)
```

Arguments

`design`	An object created with one of the `setup` functions.
`n`	The sample sizes of the baskets. A vector must be used for varying sample sizes.
`p1`	Probabilities used for the simulation. If `NULL` then all probabilities are set to p0.
`alpha`	The one-sided significance level.
`design_params`	A list of params that is specific to the class of `design`.
`iter`	The number of iterations in the simulation. Is ignored if `data` is specified.
`n_mcmc`	Number of MCMC samples.
`prec_digits`	Number of decimal places that are considered when adjusting lambda.

data	A data matrix with k column with the number of responses for each basket. Has to be generated with get_data. If data is used, then iter is ignored.
...	Further arguments.

Value

A list containing the greatest estimated value for lambda with prec_digits decimal places which controls the family wise error rate at level alpha (one-sided) and the estimated family wise error rate for the estimated lambda.

Examples

```
design <- setup_bhm(k = 3, p0 = 0.2, p_target = 0.5)

# Equal sample sizes
adjust_lambda(design = design, n = 15, design_params = list(tau_scale = 1),
  iter = 100, n_mcmc = 5000)

# Unequal sample sizes
adjust_lambda(design = design, n = c(15, 20, 25),
  design_params = list(tau_scale = 1),
  iter = 100, n_mcmc = 5000)
```

---

adjust_lambda.default *Adjust Lambda*

Description

Adjust Lambda

Usage

```
## Default S3 method:
adjust_lambda(
  design,
  n,
  p1 = NULL,
  alpha = 0.05,
  design_params = list(),
  iter = 1000,
  prec_digits = 3,
  data = NULL,
  ...
)
```

Arguments

design	An object created with one of the setup functions.
n	The sample sizes of the baskets. A vector must be used for varying sample sizes.
p1	Probabilities under the alternative hypothesis. If NULL then the type 1 error rate under the global null hypothesis is calculated.
alpha	The one-sided significance level.
design_params	A list of params that is specific to the class of design.
iter	The number of iterations in the simulation. Is ignored if data is specified.
prec_digits	Number of decimal places that are considered when adjusting lambda.
data	A data matrix with k column with the number of responses for each basket. Has to be generated with get_data. If data is used, then iter is ignored.
...	Further arguments.

Details

It is recommended to use data and then use the same simulated data set for all further calculations. If data = NULL then new data are generated in each step of the algorithm, so lambda doesn't necessarily protect the family wise error rate for different simulated data due to Monte Carlo simulation error.

Value

A list containing the greatest estimated value for lambda with prec_digits decimal places which controls the family wise error rate at level alpha (one-sided) and the estimated family wise error rate for the estimated lambda.

Examples

```
# Example for a basket trial with Fujikawa's Design
design <- setup_fujikawa(k = 3, p0 = 0.2)

# Equal sample sizes
adjust_lambda(design = design, n = 20, alpha = 0.05,
  design_params = list(epsilon = 2, tau = 0), iter = 1000)

# Unequal sample sizes
adjust_lambda(design = design, n = c(15, 20, 25), alpha = 0.05,
  design_params = list(epsilon = 2, tau = 0), iter = 1000)
```

adjust_lambda.exnex *Adjust Lambda for the EXNEX Design*

Description

Adjust Lambda for the EXNEX Design

Usage

```
## S3 method for class 'exnex'
adjust_lambda(
  design,
  n,
  p1 = NULL,
  alpha = 0.05,
  design_params = list(),
  iter = 1000,
  n_mcmc = 10000,
  prec_digits = 3,
  data = NULL,
  ...
)
```

Arguments

design	An object created with one of the setup functions.
n	The sample sizes of the baskets. A vector must be used for varying sample sizes.
p1	Probabilities used for the simulation. If NULL then all probabilities are set to p0.
alpha	The one-sided significance level.
design_params	A list of params that is specific to the class of design.
iter	The number of iterations in the simulation. Is ignored if data is specified.
n_mcmc	Number of MCMC samples.
prec_digits	Number of decimal places that are considered when adjusting lambda.
data	A data matrix with k column with the number of responses for each basket. Has to be generated with get_data. If data is used, then iter is ignored.
...	Further arguments.

Value

A list containing the greatest estimated value for lambda with prec_digits decimal places which controls the family wise error rate at level alpha (one-sided) and the estimated family wise error rate for the estimated lambda.

Examples

```
design <- setup_exnex(k = 3, p0 = 0.2)

# Equal sample sizes
adjust_lambda(design = design, n = 15,
  design_params = list(tau_scale = 1, w_j = 0.5),
  iter = 100, n_mcmc = 5000)

# Unequal sample sizes
adjust_lambda(design = design, n = c(15, 20, 25),
  design_params = list(tau_scale = 1, w_j = 0.5),
  iter = 100, n_mcmc = 5000)
```

---

ecd *Calculate the Expected Number of Correct Decisions for a Basket Trial Design*

Description

Calculate the Expected Number of Correct Decisions for a Basket Trial Design

Usage

```
ecd(
  design,
  n,
  p1,
  lambda,
  design_params = list(),
  iter = 1000,
  data = NULL,
  ...
)
```

Arguments

design	An object created with one of the setup functions.
n	The sample sizes of the baskets. A vector must be used for varying sample sizes.
p1	Probabilities used for the simulation. If NULL then all probabilities are set to p0.
lambda	The posterior probability threshold.
design_params	A list of params that is specific to the class of design.
iter	The number of iterations in the simulation. Is ignored if data is specified.
data	A data matrix with k column with the number of responses for each basket. Has to be generated with get_data. If data is used, then iter is ignored.
...	Further arguments.

Value

A numeric value.

Examples

```
# Example for a basket trial with Fujikawa's Design
design <- setup_fujikawa(k = 3, p0 = 0.2)

# Equal sample sizes
ecd(design = design, n = 20, p1 = c(0.2, 0.5, 0.5),
  lambda = 0.95, design_params = list(epsilon = 2, tau = 0),
  iter = 1000)

# Unequal sample sizes
ecd(design = design, n = c(15, 20, 25), p1 = c(0.2, 0.5, 0.5),
  lambda = 0.95, design_params = list(epsilon = 2, tau = 0),
  iter = 1000)
```

---

geom_borrow *Plot a Bayesian basket trial's posterior distribution after borrowing*

Description

Plot a Bayesian basket trial's posterior distribution after borrowing

Usage

```
geom_borrow(design, ...)
```

Arguments

design	An object created with one of the setup functions.
...	Further arguments to be passed to 'geom_function'.

Value

A list of ggplot layers of type 'geom_function'.

Examples

```
# Example for a basket trial with Fujikawa's Design
design <- setup_fujikawa(k = 3, p0 = 0.2)
n <- 20
r <- c(4, 5, 2)
epsilon <- 2
tau <- 0.5
# One facet per basket
library(ggplot2)
ggplot() +
```

```
    geom_borrow(design, n, r, epsilon, tau, logbase = exp(1)) +
    facet_wrap(vars(basket))
# Colour different baskets
ggplot() +
    geom_borrow(design, n, r, epsilon, tau,
                logbase = exp(1), aes(colour = basket))
```

`geom_borrow.fujikawa` *Plot a Fujikawa basket trial's posterior distribution after borrowing*

Description

Plot a Fujikawa basket trial's posterior distribution after borrowing

Usage

```
## S3 method for class 'fujikawa'
geom_borrow(design, n, r, epsilon, tau, logbase, ...)
```

Arguments

`design`	An object created with one of the `setup` functions.
`n`	The sample sizes of the baskets. A vector must be used for varying sample sizes.
`r`	Vector of responses.
`epsilon`	Tuning parameter that determines the amount of borrowing. See `setup_fujikawa`).
`tau`	Tuning parameter that determines how similar the baskets have to be that information is shared. See `setup_fujikawa`).
`logbase`	Tuning parameter. The base of the logarithm that is used to calculate the Jensen-Shannon divergence.
`...`	Further arguments to be passed to 'geom_function'.

Value

A list of ggplot layers of type 'geom_function'.

Examples

```
# Example for a basket trial with Fujikawa's Design
design <- setup_fujikawa(k = 3, p0 = 0.2)
n <- 20
r <- c(4, 5, 2)
epsilon <- 2
tau <- 0.5
# One facet per basket
library(ggplot2)
ggplot() +
    geom_borrow(design, n, r, epsilon, tau, logbase = exp(1)) +
```

```
    facet_wrap(vars(basket))
# Colour different baskets
ggplot() +
    geom_borrow(design, n, r, epsilon, tau,
                logbase = exp(1), aes(colour = basket))
```

`geom_posterior` *Plot a Bayesian basket trial's posterior distribution*

Description

Plot a Bayesian basket trial's posterior distribution

Usage

```
geom_posterior(design, ...)
```

Arguments

`design`	An object created with one of the `setup` functions.
`...`	Further arguments to be passed to 'geom_function'.

Value

A list of ggplot layers of type 'geom_function'.

Examples

```
# Example for a basket trial with Fujikawa's Design
design <- setup_fujikawa(k = 3, p0 = 0.2)
n <- 20
r <- c(4, 5, 2)
# One facet per basket
library(ggplot2)
ggplot() +
    geom_posterior(design, n, r) +
    facet_wrap(vars(basket))
# Colour different baskets
ggplot() +
    geom_posterior(design, n, r, aes(colour = basket))
```

geom_posterior.fujikawa *Plot a Fujikawa basket trial's posterior distribution*

Description

Plot a Fujikawa basket trial's posterior distribution

Usage

```
## S3 method for class 'fujikawa'
geom_posterior(design, n, r, ...)
```

Arguments

design	An object created with one of the setup functions.
n	The sample sizes of the baskets. A vector must be used for varying sample sizes.
r	Vector of responses.
...	Further arguments to be passed to 'geom_function'.

Value

A list of ggplot layers of type 'geom_function'.

Examples

```
# Example for a basket trial with Fujikawa's Design
design <- setup_fujikawa(k = 3, p0 = 0.2)
n <- 20
r <- c(4, 5, 2)
# One facet per basket
library(ggplot2)
ggplot() +
    geom_posterior(design, n, r) +
    facet_wrap(vars(basket))
# Colour different baskets
ggplot() +
    geom_posterior(design, n, r, aes(colour = basket))
```

geom_prior *Plot a Bayesian basket trial's prior distribution*

Description

Plot a Bayesian basket trial's prior distribution

Usage

```
geom_prior(design, ...)
```

Arguments

design	An object created with one of the setup functions.
...	Further arguments to be passed to 'geom_function'.

Value

A list of ggplot layers of type 'geom_function'.

Examples

```
# Example for a basket trial with Fujikawa's Design
design <- setup_fujikawa(k = 3, p0 = 0.2)
# One facet per basket
library(ggplot2)
ggplot() +
    geom_prior(design) +
    facet_wrap(vars(basket))
# Colour different baskets
ggplot() +
    geom_prior(design, aes(colour = basket))
```

geom_prior.fujikawa *Plot a Fujikawa basket trial's prior distribution*

Description

Plot a Fujikawa basket trial's prior distribution

Usage

```
## S3 method for class 'fujikawa'
geom_prior(design, ...)
```

Arguments

design	An object created with one of the setup functions.
...	Further arguments to be passed to 'geom_function'.

Value

A list of ggplot layers of type 'geom_function'.

Examples

```
# Example for a basket trial with Fujikawa's Design
design <- setup_fujikawa(k = 3, p0 = 0.2)
# One facet per basket
library(ggplot2)
ggplot() +
    geom_prior(design) +
    facet_wrap(vars(basket))
# Colour different baskets
ggplot() +
    geom_prior(design, aes(colour = basket))
```

---

get_data *Simulate Data Based on a Binomial Distribution*

Description

Simulate Data Based on a Binomial Distribution

Usage

```
get_data(k, n, p, iter, type = c("matrix", "bhmbasket"))
```

Arguments

k	The number of baskets.
n	The sample sizes of the baskets. A vector must be used for varying sample sizes.
p	Probabilities used to simulate the data
iter	The number of iterations in the simulation. Is ignored if data is specified.
type	Type of output. Use bhmbasket for the BHM and EXNED design and matrix for everything else.

Details

For type = "bhmbasket" this is simply a wraper for bhmbasket::simulateScenarios.

Value

If type = "matrix" then a matrix is returned, if type = "bhmbasket" then an element with class scenario_list.

Examples

```
# Equal sample sizes
get_data(k = 3, n = 20, p = c(0.2, 0.2, 0.5), iter = 1000,
  type = "matrix")

# Unequal sample sizes
get_data(k = 3, n = c(15, 20, 25), p = c(0.2, 0.2, 0.5),
  iter = 1000, type = "matrix")
```

---

get_details *Get Details of a Basket Trial Simulation*

Description

Get Details of a Basket Trial Simulation

Usage

```
get_details(design, ...)
```

Arguments

design	An object created with one of the setup functions.
...	Further arguments.

Value

A list containing the rejection probabilities, posterior means, mean squared errors of all baskets and the family-wise error rate. For some methods the mean limits of HDI intervals are also returned.

Examples

```
# Example for a basket trial with Fujikawa's Design
design <- setup_fujikawa(k = 3, p0 = 0.2)

# Equal sample sizes
get_details(design = design, n = 20, p1 = c(0.2, 0.5, 0.5),
  lambda = 0.95, epsilon = 2, tau = 0, iter = 100)

# Unequal sample sizes
get_details(design = design, n = c(15, 20, 25),
   p1 = c(0.2, 0.5, 0.5), lambda = 0.95, epsilon = 2,
   tau = 0, iter = 100)
```

`get_details.app` *Get Details of a Basket Trial Simulation with the Adaptive Power Prior Design for sequential clinical trials*

Description

Get Details of a Basket Trial Simulation with the Adaptive Power Prior Design for sequential clinical trials

Usage

```
## S3 method for class 'app'
get_details(
  design,
  n,
  p1 = NULL,
  lambda,
  level = 0.95,
  iter = 1000,
  data = NULL,
  ...
)
```

Arguments

`design`	An object of class app.
`n`	The sample sizes of the baskets. A vector must be used for varying sample sizes.
`p1`	Probabilities used for the simulation. If `NULL` then all probabilities are set to p0.
`lambda`	The posterior probability threshold.
`level`	Level of the credibility intervals.
`iter`	The number of iterations in the simulation. Is ignored if `data` is specified.
`data`	A data matrix with k column with the number of responses for each basket. Has to be generated with `get_data`. If `data` is used, then `iter` is ignored.
`...`	Further arguments.

Value

A list containing the rejection probabilities, posterior means, mean squared errors and mean limits of HDI intervals for all baskets as well as the family-wise error rate.

Examples

```
design <- setup_app(k = 3, p0 = 0.2)

# Equal sample sizes
```

```
get_details(design = design, n = 20, p1 = c(0.2, 0.5, 0.5),
 lambda = 0.95, iter = 100)

# Unequal sample sizes
get_details(design = design, n = c(15, 20, 25), p1 = c(0.2, 0.5, 0.5),
 lambda = 0.95, iter = 100)
```

`get_details.bhm` *Get Details of a BHM Basket Trial Simulation*

Description

Get Details of a BHM Basket Trial Simulation

Usage

```
## S3 method for class 'bhm'
get_details(
  design,
  n,
  p1 = NULL,
  lambda,
  level = 0.95,
  tau_scale,
  iter = 1000,
  n_mcmc = 10000,
  data = NULL,
  ...
)
```

Arguments

`design`	An object of class `bhm`.
`n`	The sample sizes of the baskets. A vector must be used for varying sample sizes.
`p1`	Probabilities used for the simulation. If `NULL` then all probabilities are set to p0.
`lambda`	The posterior probability threshold.
`level`	Level of the credibility intervals.
`tau_scale`	Standard deviation of the half normal prior distribution for the variance of the thetas.
`iter`	The number of iterations in the simulation. Is ignored if `data` is specified.
`n_mcmc`	Number of MCMC samples.
`data`	An object of class `scenario_list` as returned by the function `bhmbasket::simulateScenarios`.
`...`	Further arguments.

Value

A list containing the rejection probabilities, posterior means, mean squared errors and mean limits of HDI intervals for all baskets as well as the family-wise error rate.

Examples

```
design <- setup_bhm(k = 3, p0 = 0.2, p_target = 0.5)

# Equal sample sizes
get_details(design = design, n = 20, p1 = c(0.2, 0.5, 0.5),
  lambda = 0.95, tau_scale = 1, iter = 100)

# Unequal sample sizes
get_details(design = design, n = c(15, 20, 25), p1 = c(0.2, 0.5, 0.5),
  lambda = 0.95, tau_scale = 1, iter = 100)
```

---

get_details.binomial *Get Details of a Basket Trial with the Frequentist Binomial Design*

Description

This basic frequentist design conducts a separate binomial test for each basket. All tests are one-sided, and the alternative is greater than the null hypothesis. These details are calculated exactly, not simulated.

Usage

```
## S3 method for class 'binomial'
get_details(design, n, p1 = NULL, alpha = 0.025, ...)
```

Arguments

design	An object of class binomial.
n	The sample sizes of the baskets. A vector must be used for varying sample sizes.
p1	Probabilities used for the simulation. If NULL then all probabilities are set to p0.
alpha	The one-sided significance level.
...	Further arguments.

Value

A list containing the rejection probabilities, critical values and expected number of correct decisions. Critical values c are defined so that the null hypothesis is rejected if the observed number of responses r is greater than c, i.e. $r > c$ rejects H_0. The nominal FWER is the (theoretical) FWER of a multiple testing problem with k hypothesis tests at significance level α. The actual FWER is usually lower than the nominal FWER, as the binomial test does not exhaust its significance level.

Examples

```
design <- setup_binomial(k = 3, p0 = 0.2)
p1 <- c(0.2, 0.5, 0.5)
get_details(design = design, n = 20, p1 = p1)
```

`get_details.cpp` *Get Details of a Basket Trial Simulation with the Calibrated Power Prior Design*

Description

Get Details of a Basket Trial Simulation with the Calibrated Power Prior Design

Usage

```
## S3 method for class 'cpp'
get_details(
  design,
  n,
  p1 = NULL,
  lambda,
  level = 0.95,
  tune_a,
  tune_b,
  iter = 1000,
  data = NULL,
  ...
)
```

Arguments

`design`	An object of class `cpp`.
`n`	The sample sizes of the baskets. A vector must be used for varying sample sizes.
`p1`	Probabilities used for the simulation. If `NULL` then all probabilities are set to p0.
`lambda`	The posterior probability threshold.
`level`	Level of the credibility intervals.
`tune_a`	First tuning parameter that determines the amount of borrowing based on pairwise similarity between baskets.
`tune_b`	Second tuning parameter that determines the amount of borrowing based on pairwise similarity between baskets.
`iter`	The number of iterations in the simulation. Is ignored if `data` is specified.
`data`	A data matrix with k column with the number of responses for each basket. Has to be generated with `get_data`. If `data` is used, then `iter` is ignored.
`...`	Further arguments.

Value

A list containing the rejection probabilities, posterior means, mean squared errors and mean limits of HDI intervals for all baskets as well as the family-wise error rate.

Examples

```
design <- setup_cpp(k = 3, p0 = 0.2)

# Equal sample sizes
get_details(design = design, n = 20, p1 = c(0.2, 0.5, 0.5),
  lambda = 0.95, tune_a = 1, tune_b = 1, iter = 100)

# Unequal sample sizes
get_details(design = design, n = c(15, 20, 25), p1 = c(0.2, 0.5, 0.5),
  lambda = 0.95, tune_a = 1, tune_b = 1, iter = 100)
```

---

get_details.cppglobal *Get Details of a Basket Trial Simulation with the Global Calibrated Power Prior Design*

Description

Get Details of a Basket Trial Simulation with the Global Calibrated Power Prior Design

Usage

```
## S3 method for class 'cppglobal'
get_details(
  design,
  n,
  p1 = NULL,
  lambda,
  level = 0.95,
  tune_a,
  tune_b,
  epsilon,
  iter = 1000,
  data = NULL,
  ...
)
```

Arguments

design	An object of class cppgen.
n	The sample sizes of the baskets. A vector must be used for varying sample sizes.
p1	Probabilities used for the simulation. If NULL then all probabilities are set to p0.
lambda	The posterior probability threshold.

level	Level of the credibility intervals.
tune_a	First tuning parameter that determines the amount of borrowing based on pairwise similarity between baskets.
tune_b	Second tuning parameter that determines the amount of borrowing based on pairwise similarity between baskets.
epsilon	Tuning parameter that determines the amount of borrowing based on overall heterogeneity.
iter	The number of iterations in the simulation. Is ignored if data is specified.
data	A data matrix with k column with the number of responses for each basket. Has to be generated with get_data. If data is used, then iter is ignored.
...	Further arguments.

Value

A list containing the rejection probabilities, posterior means, mean squared errors and mean limits of HDI intervals for all baskets as well as the family-wise error rate.

Examples

```
design <- setup_cppglobal(k = 3, p0 = 0.2)
get_details(design = design, n = 20, p1 = c(0.2, 0.5, 0.5), lambda = 0.95,
  tune_a = 1, tune_b = 1, epsilon = 2, iter = 100)
```

---

get_details.cpplim *Get Details of a Basket Trial Simulation with the Limited Calibrated Power Prior Design*

Description

Get Details of a Basket Trial Simulation with the Limited Calibrated Power Prior Design

Usage

```
## S3 method for class 'cpplim'
get_details(
  design,
  n,
  p1 = NULL,
  lambda,
  level = 0.95,
  tune_a,
  tune_b,
  iter = 1000,
  data = NULL,
  ...
)
```

Arguments

design	An object of class cpplim.
n	The sample sizes of the baskets. A vector must be used for varying sample sizes.
p1	Probabilities used for the simulation. If NULL then all probabilities are set to p0.
lambda	The posterior probability threshold.
level	Level of the credibility intervals.
tune_a	First tuning parameter that determines the amount of borrowing based on pairwise similarity between baskets.
tune_b	Second tuning parameter that determines the amount of borrowing based on pairwise similarity between baskets.
iter	The number of iterations in the simulation. Is ignored if data is specified.
data	A data matrix with k column with the number of responses for each basket. Has to be generated with get_data. If data is used, then iter is ignored.
...	Further arguments.

Value

A list containing the rejection probabilities, posterior means, mean squared errors and mean limits of HDI intervals for all baskets as well as the family-wise error rate.

Examples

```
design <- setup_cpplim(k = 3, p0 = 0.2)

# Equal sample sizes
get_details(design = design, n = 20, p1 = c(0.2, 0.5, 0.5),
  lambda = 0.95, tune_a = 1, tune_b = 1, iter = 100)

# Unequal sample sizes
get_details(design = design, n = c(15, 20, 25), p1 = c(0.2, 0.5, 0.5),
  lambda = 0.95, tune_a = 1, tune_b = 1, iter = 100)
```

---

get_details.exnex *Get Details of a Basket Trial Simulation with the EXNEX Design*

Description

Get Details of a Basket Trial Simulation with the EXNEX Design

Usage

```
## S3 method for class 'exnex'
get_details(
  design,
  n,
  p1 = NULL,
  lambda,
  level = 0.95,
  tau_scale,
  w_j,
  iter = 1000,
  n_mcmc = 10000,
  data = NULL,
  ...
)
```

Arguments

design	An object of class exnex.
n	The sample sizes of the baskets. A vector must be used for varying sample sizes.
p1	Probabilities used for the simulation. If NULL then all probabilities are set to p0.
lambda	The posterior probability threshold.
level	Level of the credibility intervals.
tau_scale	Standard deviation of the half normal prior exchangeability distribution for the variance of the thetas.
w_j	Fixed prior weight for the exchangeability part of the model.
iter	The number of iterations in the simulation. Is ignored if data is specified.
n_mcmc	Number of MCMC samples.
data	An object of class scenario_list as returned by the function bhmbasket::simulateScenarios.
...	Further arguments.

Value

A list containing the rejection probabilities, posterior means, mean squared errors and mean limits of HDI intervals for all baskets as well as the family-wise error rate.

Examples

```
design <- setup_exnex(k = 3, p0 = 0.2)

# Equal sample sizes
get_details(design = design, n = 20, p1 = c(0.2, 0.5, 0.5),
  lambda = 0.95, tau_scale = 1, w_j = 0.5, iter = 100)

# Unequal sample sizes
get_details(design = design, n = c(15, 20, 25), p1 = c(0.2, 0.5, 0.5),
```

```
lambda = 0.95, tau_scale = 1, w_j = 0.5, iter = 100)
```

`get_details.fujikawa` *Get Details of a Basket Trial Simulation with Fujikawa's Design*

Description

Get Details of a Basket Trial Simulation with Fujikawa's Design

Usage

```
## S3 method for class 'fujikawa'
get_details(
  design,
  n,
  p1 = NULL,
  lambda,
  level = 0.95,
  epsilon,
  tau,
  logbase = 2,
  iter = 1000,
  data = NULL,
  use_future = FALSE,
  weight_fun = NULL,
  weight_params = list(epsilon = epsilon, tau = tau, logbase = logbase),
  ...
)
```

Arguments

`design`	An object of class `fujikawa`.
`n`	The sample sizes of the baskets. A vector must be used for varying sample sizes.
`p1`	Probabilities used for the simulation. If `NULL` then all probabilities are set to p0.
`lambda`	The posterior probability threshold.
`level`	Level of the credibility intervals.
`epsilon`	Tuning parameter that determines the amount of borrowing. See `setup_fujikawa`).
`tau`	Tuning parameter that determines how similar the baskets have to be that information is shared. See `setup_fujikawa`).
`logbase`	Tuning parameter. The base of the logarithm that is used to calculate the Jensen-Shannon divergence.
`iter`	The number of iterations in the simulation. Is ignored if `data` is specified.
`data`	A data matrix with k column with the number of responses for each basket. Has to be generated with `get_data`. If `data` is used, then `iter` is ignored.

use_future	A logical, should %dofuture% or %do% be used for the call to foreach. Default is FALSE which means that %dofuture% is not used. %dofuture% is needed for parallelization. Note that for actually using parallelized calculations, one needs to activate a future backend.
weight_fun	A function of the form function(design, n, ...) that additionally takes the arguments given in weight_params. If NULL, the original weights suggested by Fujikawa are used (based on the Jensen-Shannon divergence).
weight_params	A named list of input parameters (additional to design and n) for the function weight_fun.
...	Further arguments.

Value

A list containing the rejection probabilities, posterior means, mean squared errors and mean limits of HDI intervals for all baskets as well as the family-wise error rate and the experiment-wise power.

Examples

```
design <- setup_fujikawa(k = 3, p0 = 0.2)

# Equal sample sizes
get_details(design = design, n = 20, p1 = c(0.2, 0.5, 0.5),
  lambda = 0.95, epsilon = 2, tau = 0, iter = 100)

# Unequal sample sizes
get_details(design = design, n = c(15, 20, 25), p1 = c(0.2, 0.5, 0.5),
  lambda = 0.95, epsilon = 2, tau = 0, iter = 100)

# A custom weight function can be defined, e.g.
weight_noshare <- function(design, n, epsilon, tau, logbase){
  n_sum <- n + 1
  return(diag(n_sum))
}
get_details(design = design, n = 20, p1 = c(0.2, 0.5, 0.5), lambda = 0.95,
            epsilon = 2, tau = 0, iter = 1000, weight_fun = weight_noshare)
```

---

get_details.jsdglobal *Get Details of a Basket Trial Simulation with the Power Prior Design Based on Global JSD Weights*

Description

Get Details of a Basket Trial Simulation with the Power Prior Design Based on Global JSD Weights

Usage

```
## S3 method for class 'jsdglobal'
get_details(
  design,
  n,
  p1 = NULL,
  lambda,
  level = 0.95,
  eps_pair,
  tau = 0,
  eps_all,
  logbase = 2,
  iter = 1000,
  data = NULL,
  ...
)
```

Arguments

design	An object of class jsdgen.
n	The sample sizes of the baskets. A vector must be used for varying sample sizes.
p1	Probabilities used for the simulation. If NULL then all probabilities are set to p0.
lambda	The posterior probability threshold.
level	Level of the credibility intervals.
eps_pair	Tuning parameter that determines the amount of borrowing based on pairwise similarity.
tau	Tuning parameter that determines how similar the baskets have to be that information is shared.
eps_all	Tuning parameter that determines the amount of borrowing based on overall heterogeneity.
logbase	Tuning parameter. The base of the logarithm that is used to calculate the Jensen-Shannon divergence.
iter	The number of iterations in the simulation. Is ignored if data is specified.
data	A data matrix with k column with the number of responses for each basket. Has to be generated with get_data. If data is used, then iter is ignored.
...	Further arguments.

Value

A list containing the rejection probabilities, posterior means, mean squared errors and mean limits of HDI intervals for all baskets as well as the family-wise error rate.

Examples

```
design <- setup_jsdglobal(k = 3, p0 = 0.2)
get_details(design = design, n = 20, p1 = c(0.2, 0.5, 0.5), lambda = 0.95,
  eps_pair = 2, eps_all = 2, iter = 100)
```

get_details.mml *Get Details of a Basket Trial Simulation with the MML Design*

Description

Get Details of a Basket Trial Simulation with the MML Design

Usage

```
## S3 method for class 'mml'
get_details(
  design,
  n,
  p1 = NULL,
  lambda,
  level = 0.95,
  iter = 1000,
  data = NULL,
  ...
)
```

Arguments

design	An object of class cpp.
n	The sample sizes of the baskets. A vector must be used for varying sample sizes.
p1	Probabilities used for the simulation. If NULL then all probabilities are set to p0.
lambda	The posterior probability threshold.
level	Level of the credibility intervals.
iter	The number of iterations in the simulation. Is ignored if data is specified.
data	A data matrix with k column with the number of responses for each basket. Has to be generated with get_data. If data is used, then iter is ignored.
...	Further arguments.

Value

A list containing the rejection probabilities, posterior means, mean squared errors and mean limits of HDI intervals for all baskets as well as the family-wise error rate.

Examples

```
design <- setup_mml(k = 3, p0 = 0.2)
get_details(design = design, n = 20, p1 = c(0.2, 0.5, 0.5), lambda = 0.95,
  tune_a = 1, tune_b = 1, iter = 100)
```

`get_details.mmlglobal` *Get Details of a Basket Trial Simulation with the Global MML Design*

Description

Get Details of a Basket Trial Simulation with the Global MML Design

Usage

```
## S3 method for class 'mmlglobal'
get_details(
  design,
  n,
  p1 = NULL,
  lambda,
  level = 0.95,
  iter = 1000,
  data = NULL,
  ...
)
```

Arguments

`design`	An object of class `mmlglobal`.
`n`	The sample sizes of the baskets. A vector must be used for varying sample sizes.
`p1`	Probabilities used for the simulation. If `NULL` then all probabilities are set to p0.
`lambda`	The posterior probability threshold.
`level`	Level of the credibility intervals.
`iter`	The number of iterations in the simulation. Is ignored if `data` is specified.
`data`	A data matrix with k column with the number of responses for each basket. Has to be generated with `get_data`. If `data` is used, then `iter` is ignored.
`...`	Further arguments.

Value

A list containing the rejection probabilities, posterior means, mean squared errors and mean limits of HDI intervals for all baskets as well as the family-wise error rate.

Examples

```
design <- setup_mmlglobal(k = 3, p0 = 0.2)
get_details(design = design, n = 20, p1 = 0.5, lambda = 0.95, iter = 100)
```

get_evaluation *Evaluate a Basket Trial*

Description

Evaluate a Basket Trial

Usage

```
get_evaluation(design, ...)
```

Arguments

design	An object created with one of the setup functions.
...	Further arguments.

Value

A list containing the point estimates of the basket-specific response rates and, for some methods, the posterior probabilities that the estimated response rates are above a specified threshold p0.

Examples

```
# Example for a basket trial with Fujikawa's Design
design <- setup_fujikawa(k = 3, p0 = 0.2)

# Equal sample sizes
get_evaluation(design = design, n = 20, r = c(10, 15, 5),
  lambda = 0.95, epsilon = 2, tau = 0, iter = 100)

# Unequal sample sizes
get_evaluation(design = design, n = c(15, 20, 25),
   r = c(10, 15, 17), lambda = 0.95, epsilon = 2,
   tau = 0, iter = 100)
```

get_evaluation.app *Evaluate a Basket Trial with the Adaptive Power Prior Design for sequential clinical trials*

Description

Evaluate a Basket Trial with the Adaptive Power Prior Design for sequential clinical trials

Usage

```
## S3 method for class 'app'
get_evaluation(design, n, r, lambda, level = 0.95, ...)
```

Arguments

design	An object of class app.
n	The sample sizes of the baskets. A vector must be used for varying sample sizes.
r	Vector of responses.
lambda	The posterior probability threshold.
level	Level of the credibility intervals.
...	Further arguments.

Value

A list containing the point estimates of the basket-specific response rates and the posterior probabilities that the estimated response rates are above a specified threshold p0.

Examples

```
design <- setup_app(k = 3, p0 = 0.2)

# Equal sample sizes
get_evaluation(design = design, n = 20, r = c(10, 15, 5),
 lambda = 0.95, iter = 100)

# Unequal sample sizes
get_evaluation(design = design, n = c(15, 20, 25), r = c(10, 15, 17),
 lambda = 0.95, iter = 100)
```

---

get_evaluation.bhm *Evaluate a BHM Basket Trial*

Description

Evaluate a BHM Basket Trial

Usage

```
## S3 method for class 'bhm'
get_evaluation(
  design,
  n,
  r,
  lambda,
  level = 0.95,
  tau_scale,
  n_mcmc = 10000,
  ...
)
```

Arguments

design	An object of class bhm.
n	The sample sizes of the baskets. A vector must be used for varying sample sizes.
r	Vector of responses.
lambda	The posterior probability threshold.
level	Level of the credibility intervals.
tau_scale	Standard deviation of the half normal prior distribution for the variance of the thetas.
n_mcmc	Number of MCMC samples.
...	Further arguments.

Value

A list containing the point estimates of the basket-specific response rates.

Examples

```
design <- setup_bhm(k = 3, p0 = 0.2, p_target = 0.5)

get_evaluation(design = design, n = c(20, 20, 20), r = c(10, 15, 5),
  lambda = 0.95, tau_scale = 1, iter = 100)

# Unequal sample sizes
get_evaluation(design = design, n = c(15, 20, 25), r = c(10, 15, 17),
  lambda = 0.95, tau_scale = 1, iter = 100)
```

---

get_evaluation.cpp *Evaluate a Basket Trial with the Calibrated Power Prior Design*

Description

Evaluate a Basket Trial with the Calibrated Power Prior Design

Usage

```
## S3 method for class 'cpp'
get_evaluation(design, n, r, lambda, level = 0.95, tune_a, tune_b, ...)
```

Arguments

design	An object of class cpp.
n	The sample sizes of the baskets. A vector must be used for varying sample sizes.
r	Vector of responses.
lambda	The posterior probability threshold.

level	Level of the credibility intervals.
tune_a	First tuning parameter that determines the amount of borrowing based on pairwise similarity between baskets.
tune_b	Second tuning parameter that determines the amount of borrowing based on pairwise similarity between baskets.
...	Further arguments.

Value

A list containing the point estimates of the basket-specific response rates and the posterior probabilities that the estimated response rates are above a specified threshold p0.

Examples

```
design <- setup_cpp(k = 3, p0 = 0.2)

# Equal sample sizes
get_evaluation(design = design, n = 20, r = c(10, 15, 5),
  lambda = 0.95, tune_a = 1, tune_b = 1, iter = 100)

# Unequal sample sizes
get_evaluation(design = design, n = c(15, 20, 25), r = c(10, 15, 17),
  lambda = 0.95, tune_a = 1, tune_b = 1, iter = 100)
```

---

get_evaluation.cpplim *Evaluate a Basket Trial with the Limited Calibrated Power Prior Design*

Description

Evaluate a Basket Trial with the Limited Calibrated Power Prior Design

Usage

```
## S3 method for class 'cpplim'
get_evaluation(design, n, r, lambda, level = 0.95, tune_a, tune_b, ...)
```

Arguments

design	An object of class cpplim.
n	The sample sizes of the baskets. A vector must be used for varying sample sizes.
r	Vector of responses.
lambda	The posterior probability threshold.
level	Level of the credibility intervals.
tune_a	First tuning parameter that determines the amount of borrowing based on pairwise similarity between baskets.

tune_b	Second tuning parameter that determines the amount of borrowing based on pairwise similarity between baskets.
...	Further arguments.

Value

A list containing the point estimates of the basket-specific response rates and the posterior probabilities that the estimated response rates are above a specified threshold p0.

Examples

```
design <- setup_cpplim(k = 3, p0 = 0.2)

# Equal sample sizes
get_evaluation(design = design, n = 20, r = c(10, 15, 5),
  lambda = 0.95, tune_a = 1, tune_b = 1, iter = 100)

# Unequal sample sizes
get_evaluation(design = design, n = c(15, 20, 25), r = c(10, 15, 17),
  lambda = 0.95, tune_a = 1, tune_b = 1, iter = 100)
```

---

get_evaluation.exnex *Evaluate a Basket Trial with the EXNEX Design*

Description

Evaluate a Basket Trial with the EXNEX Design

Usage

```
## S3 method for class 'exnex'
get_evaluation(
  design,
  n,
  r,
  lambda,
  level = 0.95,
  tau_scale,
  w_j,
  n_mcmc = 10000,
  ...
)
```

Arguments

design	An object of class exnex.
n	The sample sizes of the baskets. A vector must be used for varying sample sizes.

r	Vector of responses.
lambda	The posterior probability threshold.
level	Level of the credibility intervals.
tau_scale	Standard deviation of the half normal prior exchangeability distribution for the variance of the thetas.
w_j	Fixed prior weight for the exchangeability part of the model.
n_mcmc	Number of MCMC samples.
...	Further arguments.

Value

A list containing the point estimates of the basket-specific response rates.

Examples

```
design <- setup_exnex(k = 3, p0 = 0.2)

# Equal sample sizes
get_evaluation(design = design, n = c(20, 20, 20), r = c(10, 15, 5),
  lambda = 0.95, tau_scale = 1, w_j = 0.5, iter = 100)

# Unequal sample sizes
get_evaluation(design = design, n = c(15, 20, 25), r = c(10, 15, 17),
  lambda = 0.95, tau_scale = 1, w_j = 0.5, iter = 100)
```

---

get_evaluation.fujikawa

Evaluate a Basket Trial with Fujikawa's Design

Description

Evaluate a Basket Trial with Fujikawa's Design

Usage

```
## S3 method for class 'fujikawa'
get_evaluation(
  design,
  n,
  r,
  lambda,
  level = 0.95,
  epsilon,
  tau,
  logbase = 2,
  ...
)
```

Arguments

design	An object of class fujikawa.
n	The sample sizes of the baskets. A vector must be used for varying sample sizes.
r	Vector of responses.
lambda	The posterior probability threshold.
level	Level of the credibility intervals.
epsilon	Tuning parameter that determines the amount of borrowing. See setup_fujikawa).
tau	Tuning parameter that determines how similar the baskets have to be that information is shared. See setup_fujikawa).
logbase	Tuning parameter. The base of the logarithm that is used to calculate the Jensen-Shannon divergence.
...	Further arguments.

Value

A list containing the point estimates of the basket-specific response rates and the posterior probabilities that the estimated response rates are above a specified threshold p0.

Examples

```
design <- setup_fujikawa(k = 3, p0 = 0.2)

# Equal sample sizes
get_evaluation(design = design, n = 20, r = c(10, 15, 5),
  lambda = 0.95, epsilon = 2, tau = 0, iter = 100)

# Unequal sample sizes
get_evaluation(design = design, n = c(15, 20, 25),
   r = c(10, 15, 17), lambda = 0.95, epsilon = 2,
   tau = 0, iter = 100)
```

---

get_results *Get Results for Simulation of Basket Trial Designs*

Description

Get Results for Simulation of Basket Trial Designs

Usage

```
get_results(design, ...)
```

Arguments

design	An object created with one of the setup functions.
...	Further arguments.

Value

A matrix of results with `iter` rows. A 0 means, that the null hypothesis that the response probability exceeds `p0` was not rejected, a 1 means, that the null hypothesis was rejected.

Examples

```
# Example for a basket trial with Fujikawa's Design
design <- setup_fujikawa(k = 3, p0 = 0.2)

# Equal sample sizes
get_results(design = design, n = 20, p1 = c(0.2, 0.5, 0.5),
  lambda = 0.95, epsilon = 2, tau = 0, iter = 100)

# Unequal sample sizes
get_results(design = design, n = c(15, 20, 25), p1 = c(0.2, 0.5, 0.5),
  lambda = 0.95, epsilon = 2, tau = 0, iter = 100)
```

---

get_results.app *Get Results for Simulation of a Basket Trial with Adaptive Power Prior Design*

Description

Get Results for Simulation of a Basket Trial with Adaptive Power Prior Design

Usage

```
## S3 method for class 'app'
get_results(design, n, p1 = NULL, lambda, iter = 1000, data = NULL, ...)
```

Arguments

`design`	An object of class app.
`n`	The sample sizes of the baskets. A vector must be used for varying sample sizes.
`p1`	Probabilities used for the simulation. If `NULL` then all probabilities are set to p0.
`lambda`	The posterior probability threshold.
`iter`	The number of iterations in the simulation. Is ignored if `data` is specified.
`data`	A data matrix with k column with the number of responses for each basket. Has to be generated with `get_data`. If `data` is used, then `iter` is ignored.
`...`	Further arguments.

Value

A matrix of results with `iter` rows. A 0 means, that the null hypothesis that the response probability exceeds `p0` was not rejected, a 1 means, that the null hypothesis was rejected.

Examples

```
design <- setup_app(k = 3, p0 = 0.2)

# Equal sample sizes
get_results(design = design, n = 20, p1 = c(0.2, 0.5, 0.5),
  lambda = 0.95, iter = 100)

# Unequal sample sizes
get_results(design = design, n = c(15, 20, 25), p1 = c(0.2, 0.5, 0.5),
  lambda = 0.95, iter = 100)
```

---

get_results.bhm *Get Results for Simulation of a Basket Trial with the BHM Design*

Description

Get Results for Simulation of a Basket Trial with the BHM Design

Usage

```
## S3 method for class 'bhm'
get_results(
  design,
  n,
  p1 = NULL,
  lambda,
  tau_scale,
  iter = 1000,
  n_mcmc = 10000,
  data = NULL,
  ...
)
```

Arguments

design	An object of class bhm.
n	The sample sizes of the baskets. A vector must be used for varying sample sizes.
p1	Probabilities used for the simulation. If NULL then all probabilities are set to p0.
lambda	The posterior probability threshold.
tau_scale	Standard deviation of the half normal prior distribution for the variance of the thetas.
iter	The number of iterations in the simulation. Is ignored if data is specified.
n_mcmc	Number of MCMC samples.
data	An object of class scenario_list as returned by the function bhmbasket::simulateScenarios.
...	Further arguments.

Value

A matrix of results with `iter` rows. A 0 means, that the null hypothesis that the response probability exceeds `p0` was not rejected, a 1 means, that the null hypothesis was rejected.

Examples

```
design <- setup_bhm(k = 3, p0 = 0.2, p_target = 0.5)

# Equal sample sizes
get_results(design, n = 20, p1 = c(0.2, 0.5, 0.5),
  lambda = 0.95, tau_scale = 1, iter = 100)

# Unequal sample sizes
get_results(design, n = c(15, 20, 25), p1 = c(0.2, 0.5, 0.5),
  lambda = 0.95, tau_scale = 1, iter = 100)
```

---

get_results.cpp *Get Results for Simulation of a Basket Trial with a Calibrated Power Prior Design*

Description

Get Results for Simulation of a Basket Trial with a Calibrated Power Prior Design

Usage

```
## S3 method for class 'cpp'
get_results(
  design,
  n,
  p1 = NULL,
  lambda,
  tune_a,
  tune_b,
  iter = 1000,
  data = NULL,
  ...
)
```

Arguments

`design`	An object of class cpp.
`n`	The sample sizes of the baskets. A vector must be used for varying sample sizes.
`p1`	Probabilities used for the simulation. If `NULL` then all probabilities are set to p0.
`lambda`	The posterior probability threshold.

tune_a	First tuning parameter that determines the amount of borrowing based on pairwise similarity between baskets.
tune_b	Second tuning parameter that determines the amount of borrowing based on pairwise similarity between baskets.
iter	The number of iterations in the simulation. Is ignored if data is specified.
data	A data matrix with k column with the number of responses for each basket. Has to be generated with get_data. If data is used, then iter is ignored.
...	Further arguments.

Value

A matrix of results with iter rows. A 0 means, that the null hypothesis that the response probability exceeds p0 was not rejected, a 1 means, that the null hypothesis was rejected.

Examples

```
design <- setup_cpp(k = 3, p0 = 0.2)

# Equal sample sizes
get_results(design = design, n = 20, p1 = c(0.2, 0.5, 0.5),
  lambda = 0.95, tune_a = 1, tune_b = 1, iter = 100)

# Unequal sample sizes
get_results(design = design, n = c(15, 20, 25), p1 = c(0.2, 0.5, 0.5),
  lambda = 0.95, tune_a = 1, tune_b = 1, iter = 100)
```

---

get_results.cppglobal *Get Results for Simulation of a Basket Trial with a Global Calibrated Power Prior Design*

Description

Get Results for Simulation of a Basket Trial with a Global Calibrated Power Prior Design

Usage

```
## S3 method for class 'cppglobal'
get_results(
  design,
  n,
  p1 = NULL,
  lambda,
  tune_a,
  tune_b,
  epsilon,
  iter = 1000,
  data = NULL,
  ...
)
```

Arguments

design	An object of class cppgen.
n	The sample sizes of the baskets. A vector must be used for varying sample sizes.
p1	Probabilities used for the simulation. If NULL then all probabilities are set to p0.
lambda	The posterior probability threshold.
tune_a	First tuning parameter that determines the amount of borrowing based on pairwise similarity between baskets.
tune_b	Second tuning parameter that determines the amount of borrowing based on pairwise similarity between baskets.
epsilon	Tuning parameter that determines the amount of borrowing based on overall heterogeneity.
iter	The number of iterations in the simulation. Is ignored if data is specified.
data	A data matrix with k column with the number of responses for each basket. Has to be generated with get_data. If data is used, then iter is ignored.
...	Further arguments.

Value

A matrix of results with iter rows. A 0 means, that the null hypothesis that the response probability exceeds p0 was not rejected, a 1 means, that the null hypothesis was rejected.

Examples

```
design <- setup_cppglobal(k = 3, p0 = 0.2)
get_results(design = design, n = 20, p1 = c(0.2, 0.5, 0.5), lambda = 0.95,
  tune_a = 1, tune_b = 1, epsilon = 2, iter = 100)
```

---

get_results.cpplim *Get Results for Simulation of a Basket Trial with a Limited Calibrated Power Prior Design*

Description

Get Results for Simulation of a Basket Trial with a Limited Calibrated Power Prior Design

Usage

```
## S3 method for class 'cpplim'
get_results(
  design,
  n,
  p1 = NULL,
  lambda,
  tune_a,
```

```
  tune_b,
  iter = 1000,
  data = NULL,
  ...
)
```

Arguments

design	An object of class cpplim.
n	The sample sizes of the baskets. A vector must be used for varying sample sizes.
p1	Probabilities used for the simulation. If NULL then all probabilities are set to p0.
lambda	The posterior probability threshold.
tune_a	First tuning parameter that determines the amount of borrowing based on pairwise similarity between baskets.
tune_b	Second tuning parameter that determines the amount of borrowing based on pairwise similarity between baskets.
iter	The number of iterations in the simulation. Is ignored if data is specified.
data	A data matrix with k column with the number of responses for each basket. Has to be generated with get_data. If data is used, then iter is ignored.
...	Further arguments.

Value

A matrix of results with iter rows. A 0 means, that the null hypothesis that the response probability exceeds p0 was not rejected, a 1 means, that the null hypothesis was rejected.

Examples

```
design <- setup_cpplim(k = 3, p0 = 0.2)

# Equal sample sizes
get_results(design = design, n = 20, p1 = c(0.2, 0.5, 0.5),
  lambda = 0.95, tune_a = 1, tune_b = 1, iter = 100)

# Unequal sample sizes
get_results(design = design, n = c(15, 20, 25), p1 = c(0.2, 0.5, 0.5),
  lambda = 0.95, tune_a = 1, tune_b = 1, iter = 100)
```

---

get_results.exnex *Get Results for Simulation of a Basket Trial with the EXNEX Design*

Description

Get Results for Simulation of a Basket Trial with the EXNEX Design

Usage

```
## S3 method for class 'exnex'
get_results(
  design,
  n,
  p1 = NULL,
  lambda,
  tau_scale,
  w_j,
  iter = 1000,
  n_mcmc = 10000,
  data = NULL,
  ...
)
```

Arguments

design	An object of class exnex.
n	The sample sizes of the baskets. A vector must be used for varying sample sizes.
p1	Probabilities used for the simulation. If NULL then all probabilities are set to p0.
lambda	The posterior probability threshold.
tau_scale	Standard deviation of the half normal prior exchangeability distribution for the variance of the thetas.
w_j	Fixed prior weight for the exchangeability part of the model.
iter	The number of iterations in the simulation. Is ignored if data is specified.
n_mcmc	Number of MCMC samples.
data	An object of class scenario_list as returned by the function bhmbasket::simulateScenarios.
...	Further arguments.

Value

A matrix of results with iter rows. A 0 means, that the null hypothesis that the response probability exceeds p0 was not rejected, a 1 means, that the null hypothesis was rejected.

Examples

```
design <- setup_exnex(k = 3, p0 = 0.2)

# Equal sample sizes
get_results(design, n = 20, p1 = c(0.2, 0.5, 0.5),
  lambda = 0.95, tau_scale = 1, w_j = 0.5, iter = 100)

# Unequal sample sizes
get_results(design, n = c(15, 20, 25), p1 = c(0.2, 0.5, 0.5),
  lambda = 0.95, tau_scale = 1, w_j = 0.5, iter = 100)
```

`get_results.fujikawa` *Get Results for Simulation of a Basket Trial with Fujikawa's Design*

Description

Get Results for Simulation of a Basket Trial with Fujikawa's Design

Usage

```
## S3 method for class 'fujikawa'
get_results(
  design,
  n,
  p1 = NULL,
  lambda,
  epsilon,
  tau,
  logbase = 2,
  iter = 1000,
  data = NULL,
  ...
)
```

Arguments

`design`	An object of class `fujikawa`.
`n`	The sample sizes of the baskets. A vector must be used for varying sample sizes.
`p1`	Probabilities used for the simulation. If `NULL` then all probabilities are set to p0.
`lambda`	The posterior probability threshold.
`epsilon`	Tuning parameter that determines the amount of borrowing. See `setup_fujikawa`).
`tau`	Tuning parameter that determines how similar the baskets have to be that information is shared. See `setup_fujikawa`).
`logbase`	Tuning parameter. The base of the logarithm that is used to calculate the Jensen-Shannon divergence.
`iter`	The number of iterations in the simulation. Is ignored if `data` is specified.
`data`	A data matrix with k column with the number of responses for each basket. Has to be generated with `get_data`. If `data` is used, then `iter` is ignored.
`...`	Further arguments.

Value

A matrix of results with `iter` rows. A 0 means, that the null hypothesis that the response probability exceeds `p0` was not rejected, a 1 means, that the null hypothesis was rejected.

Examples

```
design <- setup_fujikawa(k = 3, p0 = 0.2)

# Equal sample sizes
get_results(design = design, n = 20, p1 = c(0.2, 0.5, 0.5),
  lambda = 0.95, epsilon = 2, tau = 0, iter = 100)

# Unequal sample sizes
get_results(design = design, n = c(15, 20, 25), p1 = c(0.2, 0.5, 0.5),
  lambda = 0.95, epsilon = 2, tau = 0, iter = 100)
```

---

get_results.jsdglobal *Get Results for Simulation of a Basket Trial with the Power Prior Design Based on Global JSD Weights*

Description

Get Results for Simulation of a Basket Trial with the Power Prior Design Based on Global JSD Weights

Usage

```
## S3 method for class 'jsdglobal'
get_results(
  design,
  n,
  p1 = NULL,
  lambda,
  eps_pair,
  tau = 0,
  eps_all,
  logbase = 2,
  iter = 1000,
  data = NULL,
  ...
)
```

Arguments

design	An object of class jsdgen.
n	The sample sizes of the baskets. A vector must be used for varying sample sizes.
p1	Probabilities used for the simulation. If NULL then all probabilities are set to p0.
lambda	The posterior probability threshold.
eps_pair	Tuning parameter that determines the amount of borrowing based on pairwise similarity.

tau	Tuning parameter that determines how similar the baskets have to be that information is shared.
eps_all	Tuning parameter that determines the amount of borrowing based on overall heterogeneity.
logbase	Tuning parameter. The base of the logarithm that is used to calculate the Jensen-Shannon divergence.
iter	The number of iterations in the simulation. Is ignored if data is specified.
data	A data matrix with k column with the number of responses for each basket. Has to be generated with get_data. If data is used, then iter is ignored.
...	Further arguments.

Value

A matrix of results with iter rows. A 0 means, that the null hypothesis that the response probability exceeds p0 was not rejected, a 1 means, that the null hypothesis was rejected.

Examples

```
design <- setup_jsdglobal(k = 3, p0 = 0.2)
get_results(design = design, n = 20, p1 = c(0.2, 0.5, 0.5), lambda = 0.95,
  eps_pair = 2, eps_all = 2, iter = 100)
```

---

get_results.mml *Get Results for Simulation of a Basket Trial with the MML Design*

Description

Get Results for Simulation of a Basket Trial with the MML Design

Usage

```
## S3 method for class 'mml'
get_results(design, n, p1 = NULL, lambda, iter = 1000, data = NULL, ...)
```

Arguments

design	An object of class mml.
n	The sample sizes of the baskets. A vector must be used for varying sample sizes.
p1	Probabilities used for the simulation. If NULL then all probabilities are set to p0.
lambda	The posterior probability threshold.
iter	The number of iterations in the simulation. Is ignored if data is specified.
data	A data matrix with k column with the number of responses for each basket. Has to be generated with get_data. If data is used, then iter is ignored.
...	Further arguments.

Value

A matrix of results with `iter` rows. A 0 means, that the null hypothesis that the response probability exceeds `p0` was not rejected, a 1 means, that the null hypothesis was rejected.

Examples

```
design <- setup_mml(k = 3, p0 = 0.2)
get_results(design, n = 20, p1 = c(0.2, 0.5, 0.5), lambda = 0.95,
  iter = 100)
```

---

get_results.mmlglobal *Get Results for Simulation of a Basket Trial with the Global MML Design*

Description

Get Results for Simulation of a Basket Trial with the Global MML Design

Usage

```
## S3 method for class 'mmlglobal'
get_results(design, n, p1 = NULL, lambda, iter = 1000, data = NULL, ...)
```

Arguments

`design`	An object of class `mmlglobal`.
`n`	The sample sizes of the baskets. A vector must be used for varying sample sizes.
`p1`	Probabilities used for the simulation. If `NULL` then all probabilities are set to p0.
`lambda`	The posterior probability threshold.
`iter`	The number of iterations in the simulation. Is ignored if `data` is specified.
`data`	A data matrix with k column with the number of responses for each basket. Has to be generated with `get_data`. If `data` is used, then `iter` is ignored.
`...`	Further arguments.

Value

A matrix of results with `iter` rows. A 0 means, that the null hypothesis that the response probability exceeds `p0` was not rejected, a 1 means, that the null hypothesis was rejected.

Examples

```
design <- setup_mmlglobal(k = 3, p0 = 0.2)
get_results(design, n = 20, p1 = c(0.2, 0.5, 0.5), lambda = 0.95,
  iter = 100)
```

get_scenarios *Create a Scenario Matrix*

Description

Creates a default scenario matrix.

Usage

```
get_scenarios(design, p1)
```

Arguments

design	An object created with one of the setup functions.
p1	Probability under the alternative hypothesis.

Details

get_scenarios creates a default scenario matrix that can be used for opt_design. The function creates k + 1 scenarios, from a global null to a global alternative scenario.

Value

A matrix with k rows and k + 1 columns.

Examples

```
design <- setup_fujikawa(k = 3, p0 = 0.2)
get_scenarios(design = design, p1 = 0.5)
```

opt_design *Optimize a Basket Trial Design*

Description

Optimize a Basket Trial Design

Usage

```
opt_design(
  design,
  n,
  alpha,
  design_params = list(),
  scenarios,
  prec_digits,
```

```
  iter = 1000,
  data = NULL,
  ...
)
```

Arguments

design	An object created with one of the setup functions.
n	The sample sizes of the baskets. A vector must be used for varying sample sizes.
alpha	The one-sided significance level.
design_params	A list of params that is specific to the class of design.
scenarios	A matrix of scenarios.
prec_digits	Number of decimal places that are considered when adjusting lambda.
iter	The number of iterations in the simulation. Is ignored if data is specified.
data	A list of data matrices generated with get_data. The list elements have to correspond to the columsn of scenarios.
...	Further arguments.

Value

A matrix with the expected number of correct decisions.

Examples

```
design <- setup_fujikawa(k = 3, p0 = 0.2)
scenarios <- get_scenarios(design, p1 = 0.5)

## Equal sample sizes
# Without simulated data

opt_design(design, n = 20, alpha = 0.05, design_params =
           list(epsilon = c(1, 2), tau = c(0, 0.5)), scenarios = scenarios,
           prec_digits = 3)


# With simulated data
scenario_list <- as.list(data.frame(scenarios))
data_list <- lapply(scenario_list,
  function(x) get_data(k = 3, n = 20, p = x, iter = 1000))

opt_design(design, n = 20, alpha = 0.05, design_params =
           list(epsilon = c(1, 2), tau = c(0, 0.5)), scenarios = scenarios,
           prec_digits = 3, data = data_list)


## Unequal sample sizes
# Without simulated data
opt_design(design, n = c(15, 20, 25), alpha = 0.05, design_params =
           list(epsilon = c(1, 2), tau = c(0, 0.5)), scenarios = scenarios,
```

```
            prec_digits = 3)

# With simulated data
scenario_list <- as.list(data.frame(scenarios))
data_list <- lapply(scenario_list,
                    function(x) get_data(k = 3, n = c(15, 20, 25),
                                         p = x, iter = 1000))
opt_design(design, n = c(15, 20, 25), alpha = 0.05, design_params =
           list(epsilon = c(1, 2), tau = c(0, 0.5)), scenarios = scenarios,
           prec_digits = 3, data = data_list)
```

---

setup_app *Set Up Adaptive Power Prior Design Object*

Description

Set Up Adaptive Power Prior Design Object

Usage

```
setup_app(k, p0, shape1 = 1, shape2 = 1)
```

Arguments

k	The number of baskets.
p0	A common probability under the null hypothesis.
shape1	First common shape parameter of the beta prior.
shape2	Second common shape parameter of the beta prior.

Details

The class app implements the adaptive power prior design for sequential clinical trials proposed by Ollier et al. (2020).

Value

An S3 object of class app

References

Ollier, A., Morita, S., Ursino, M., & Zohar, S. (2020). An adaptive power prior for sequential clinical trials - Application to bridging studies. Statistical methods in medical research, 29(8), 2282–2294.

Examples

```
design_app <- setup_app(k = 3, p0 = 0.2)
```

setup_bhm *Set Up BHM Design Object*

Description

Set Up BHM Design Object

Usage

```
setup_bhm(k, p0, p_target, mu_mean = NULL, mu_sd = 100)
```

Arguments

k	The number of baskets.
p0	A common probability under the null hypothesis.
p_target	The response rate of interest. See details.
mu_mean	Mean of the normal prior distribution for the mean of the thetas. See details.
mu_sd	Standard deviation of the normal prior distribution for the mean of the thetas.

Details

The class bhm implements the Bayesian Hierarchical Model proposed by Berry et al. (2013). Methods for this class are mostly wrappers for functions from the package bhmbasket.

In the BHM the thetas of all baskets are modeled, where theta_i = logit(p_i) - logit(p_target). These thetas are assumed to come from a normal distribution with mean mu_mean and standard deviation mu_sd. If mu_mean = NULL then mu_mean is determined as logit(p0) - logit(p_target), hence the mean of the normal distribution corresponds to the null hypothesis.

Value

An S3 object of class bhm

References

Berry, S. M., Broglio, K. R., Groshen, S., & Berry, D. A. (2013). Bayesian hierarchical modeling of patient subpopulations: efficient designs of phase II oncology clinical trials. Clinical Trials, 10(5), 720-734.

Examples

```
design_bhm <- setup_bhm(k = 3, p0 = 0.2, p_target = 0.5)
```

setup_binomial *Set Up Frequentist Binomial Design Object*

Description

Set Up Frequentist Binomial Design Object

Usage

```
setup_binomial(k, p0, pool = FALSE)
```

Arguments

k	The number of baskets.
p0	A common probability under the null hypothesis.
pool	A logical, should data from all baskets be pooled? Default is FALSE. Currently FALSE is the only working option, reserved for future implementation.

Details

The class binomial implements a basket trial design, in which each null hypothesis is tested using the frequentist binomial test without multiplicity correction. All baskets are either tested separately (the default) or pooled (not implemented yet).

Value

An S3 object of class binomial

Examples

```
design_binomial <- setup_binomial(k = 3, p0 = 0.2)
```

setup_cpp *Set Up Calibrated Power Prior Design Object*

Description

Set Up Calibrated Power Prior Design Object

Usage

```
setup_cpp(k, p0, shape1 = 1, shape2 = 1)
```

Arguments

k	The number of baskets.
p0	A common probability under the null hypothesis.
shape1	First common shape parameter of the beta prior.
shape2	Second common shape parameter of the beta prior.

Details

The class cpp implements a version of the power prior design, in which the amount of information that is shared between baskets is determined by the Kolmogorov-Smirnov test statistic between baskets (which is equivalent to the absolut difference in response rates).

Value

An S3 object of class cpp

References

Baumann, L., Sauer, L. D., & Kieser, M. (2025). A Basket Trial Design Based on Power Priors. Statistics in Biopharmaceutical Research, 17(3), 446–456. https://doi.org/10.1080/19466315.2024.2402275

Examples

```
design_cpp <- setup_cpp(k = 3, p0 = 0.2)
```

---

setup_cppglobal *Set Up Global Calibrated Power Prior Design Object*

Description

Set Up Global Calibrated Power Prior Design Object

Usage

```
setup_cppglobal(k, p0, shape1 = 1, shape2 = 1)
```

Arguments

k	The number of baskets.
p0	A common probability under the null hypothesis.
shape1	First common shape parameter of the beta prior.
shape2	Second common shape parameter of the beta prior.

Details

The class cppglobal implements a version of the power prior design, in which the amount of information that is shared between baskets is determined by the Kolmogorov-Smirnov test statistic between basekts and a function based on response rate differences that quantifies the overall heterogeneity.

Value

An S3 object of class cppglobal

References

Baumann, L., Sauer, L. D., & Kieser, M. (2025). A Basket Trial Design Based on Power Priors. Statistics in Biopharmaceutical Research, 17(3), 446–456. https://doi.org/10.1080/19466315.2024.2402275

Examples

```
design_cppglobal <- setup_cppglobal(k = 3, p0 = 0.2)
```

---

setup_cpplim *Set Up Limited Calibrated Power Prior Design Object*

Description

Set Up Limited Calibrated Power Prior Design Object

Usage

```
setup_cpplim(k, p0, shape1 = 1, shape2 = 1)
```

Arguments

k	The number of baskets.
p0	A common probability under the null hypothesis.
shape1	First common shape parameter of the beta prior.
shape2	Second common shape parameter of the beta prior.

Details

The class cpplim implements a combined version of the adaptive power prior (app) and the calibrated power prior (cpp), where the parameter limiting the amount of information to be borrowed in the adaptive power prior design is included in the calibrated power prior design.

Value

An S3 object of class cpplim

References

Ollier, A., Morita, S., Ursino, M., & Zohar, S. (2020). An adaptive power prior for sequential clinical trials - Application to bridging studies. Statistical methods in medical research, 29(8), 2282–2294.

Baumann, L., Sauer, L. D., & Kieser, M. (2025). A Basket Trial Design Based on Power Priors. Statistics in Biopharmaceutical Research, 17(3), 446–456. https://doi.org/10.1080/19466315.2024.2402275

Examples

```
design_cpplim <- setup_cpplim(k = 3, p0 = 0.2)
```

`setup_exnex` *Set Up EXNEX Design Object*

Description

Set Up EXNEX Design Object

Usage

```
setup_exnex(
  k,
  p0,
  basket_mean = NULL,
  basket_sd = 100,
  mu_mean = NULL,
  mu_sd = 100
)
```

Arguments

`k`	The number of baskets.
`p0`	A common probability under the null hypothesis.
`basket_mean`	Mean of the normal prior distribution of the individual thetas (NEX part). See details.
`basket_sd`	Standard deviation of the normal prior distribution of the individual thetas (NEX part).
`mu_mean`	Mean of the normal prior exchangeability distribution for the mean of the thetas (EX part). See details.
`mu_sd`	Standard deviation of the normal prior exchangeability distribution for the mean of the thetas (EX part).

Details

The class exnex implements the EXNEX model proposed by Neuenschwander et al. (2016). Methods for this class are mostly wrappers for functions from the package bhmbasket.

In the EXNEX model the thetas of all baskets are modeled as a mixture of individual models and a Bayesian Hierarchical Model with a fixed mixture weight w. If mu_mean and basket_mean are NULL then they are set to logit(p0). Note that Neuenschwander et al. (2016) use different prior means and standard deviations. The default values here are used for better comparison with the BHM model (see setup_bhm).

Value

An S3 object of class exnex

References

Neuenschwander, B., Wandel, S., Roychoudhury, S., & Bailey, S. (2016). Robust exchangeability designs for early phase clinical trials with multiple strata. Pharmaceutical statistics, 15(2), 123-134.

Examples

```
design_exnex <- setup_exnex(k = 3, p0 = 0.2)
```

---

setup_fujikawa *Set Up Fujikawa Design Object*

Description

Set Up Fujikawa Design Object

Usage

```
setup_fujikawa(k, p0, shape1 = 1, shape2 = 1)
```

Arguments

k	The number of baskets.
p0	A common probability under the null hypothesis.
shape1	First common shape parameter of the beta prior.
shape2	Second common shape parameter of the beta prior.

Details

The class fujikawa implements a design by Fujikawa et al. (2020) in which information is shared based on the pairwise similarity between baskets which is quantified using the Jensen-Shannon divergence between the individual posterior distributions between baskets.

Value

An S3 object of class `fujikawa`

References

Fujikawa, K., Teramukai, S., Yokota, I., & Daimon, T. (2020). A Bayesian basket trial design that borrows information across strata based on the similarity between the posterior distributions of the response probability. Biometrical Journal, 62(2), 330-338.

Examples

```
design_fujikawa <- setup_fujikawa(k = 3, p0 = 0.2)
```

---

setup_jsdglobal *Set Up Global JSD Design Object*

Description

Set Up Global JSD Design Object

Usage

```
setup_jsdglobal(k, p0, shape1 = 1, shape2 = 1)
```

Arguments

`k`	The number of baskets.
`p0`	A common probability under the null hypothesis.
`shape1`	First common shape parameter of the beta prior.
`shape2`	Second common shape parameter of the beta prior.

Details

The class `jsdglobal` implements a version of the power prior design, in which information is shared based on pairwise similarity and overall heterogeneity between baskets. Both pairwise similarity and overall heterogeneity are assessed using the Jensen-Shannon divergence.

Value

An S3 object of class `jsdglobal`

References

Baumann, L., Sauer, L. D., & Kieser, M. (2025). A Basket Trial Design Based on Power Priors. Statistics in Biopharmaceutical Research, 17(3), 446–456. https://doi.org/10.1080/19466315.2024.2402275

Examples

```
design_jsdglobal <- setup_jsdglobal(k = 3, p0 = 0.2)
```

`setup_mml` *Set Up mml Design Object*

Description

Creates an object of class `mml`.

Usage

```
setup_mml(k, p0, shape1 = 1, shape2 = 1)
```

Arguments

`k`	The number of baskets.
`p0`	A common probability under the null hypothesis.
`shape1`	First common shape parameter of the beta prior.
`shape2`	Second common shape parameter of the beta prior.

Details

The class `mml` implements a modified version of the empirical Bayes method by Gravestock & Held (2017) which was proposed for borrowing strength from an external study. In their approach, the sharing weight is found as the maximum of the marginal likelihood of the weight, given the external data set. This leads, however, to non-symmetric weights when applied to sharing in basket trials, i.e. Basket i would not share the information from Basket j as the other way round. Therefore, a symmetrised version is used, where the mean of the two weights resulting from sharing in both directions is used.

Value

An S3 object of class `mml`

References

Gravestock, I., & Held, L. (2017). Adaptive power priors with empirical Bayes for clinical trials. Pharmaceutical statistics, 16(5), 349-360.

Examples

```
design_mml <- setup_mml(k = 3, p0 = 0.2)
```

`setup_mmlglobal` *Set Up mmlglobal Design Object*

Description

Creates an object of class `mmlglobal`.

Usage

```
setup_mmlglobal(k, p0, shape1 = 1, shape2 = 1)
```

Arguments

`k`	The number of baskets.
`p0`	A common probability under the null hypothesis.
`shape1`	First common shape parameter of the beta prior.
`shape2`	Second common shape parameter of the beta prior.

Details

The class `mmlglobal` implements an empirical Bayes method by Gravestock & Held (2019) which was proposed for borrowing strength from multiple external studies.

Value

An S3 object of class `mmlglobal`

References

Gravestock, I., & Held, L. (2019). Power priors based on multiple historical studies for binary outcomes. Biometrical Journal, 61(5), 1201-1218.

Baumann, L., Sauer, L. D., & Kieser, M. (2025). A Basket Trial Design Based on Power Priors. Statistics in Biopharmaceutical Research, 17(3), 446–456. https://doi.org/10.1080/19466315.2024.2402275

Examples

```
design_mmlglobal <- setup_mmlglobal(k = 3, p0 = 0.2)
```

toer *Calculate the Type 1 Error Rate for a Basket Trial Design*

Description

Calculate the Type 1 Error Rate for a Basket Trial Design

Usage

```
toer(
  design,
  n,
  p1 = NULL,
  lambda,
  design_params = list(),
  iter = 1000,
  data = NULL,
  ...
)
```

Arguments

design	An object created with one of the setup functions.
n	The sample sizes of the baskets. A vector must be used for varying sample sizes.
p1	Probabilities under the alternative hypothesis. If NULL then the type 1 error rate under the global null hypothesis is calculated.
lambda	The posterior probability threshold.
design_params	A list of params that is specific to the class of design.
iter	The number of iterations in the simulation. Is ignored if data is specified.
data	A data matrix with k column with the number of responses for each basket. Has to be generated with get_data. If data is used, then iter is ignored.
...	Further arguments.

Value

A numeric value.

Examples

```
# Example for a basket trial with Fujikawa's Design
design <- setup_fujikawa(k = 3, p0 = 0.2)

# Equal sample sizes
toer(design = design, n = 20, p1 = c(0.2, 0.5, 0.5),
  lambda = 0.95, design_params = list(epsilon = 2, tau = 0),
  iter = 1000)
```

```
# Unequal sample sizes
toer(design = design, n = c(15, 20, 25), p1 = c(0.2, 0.5, 0.5),
  lambda = 0.95, design_params = list(epsilon = 2, tau = 0),
  iter = 1000)
```

Index

adjust_lambda, 3
adjust_lambda.bhm, 4
adjust_lambda.default, 5
adjust_lambda.exnex, 7

ecd, 8

geom_borrow, 9
geom_borrow.fujikawa, 10
geom_posterior, 11
geom_posterior.fujikawa, 12
geom_prior, 13
geom_prior.fujikawa, 13
get_data, 14
get_details, 15
get_details.app, 16
get_details.bhm, 17
get_details.binomial, 18
get_details.cpp, 19
get_details.cppglobal, 20
get_details.cpplim, 21
get_details.exnex, 22
get_details.fujikawa, 24
get_details.jsdglobal, 25
get_details.mml, 27
get_details.mmlglobal, 28
get_evaluation, 29
get_evaluation.app, 29
get_evaluation.bhm, 30
get_evaluation.cpp, 31
get_evaluation.cpplim, 32
get_evaluation.exnex, 33
get_evaluation.fujikawa, 34
get_results, 35
get_results.app, 36
get_results.bhm, 37
get_results.cpp, 38
get_results.cppglobal, 39
get_results.cpplim, 40
get_results.exnex, 41
get_results.fujikawa, 43
get_results.jsdglobal, 44
get_results.mml, 45
get_results.mmlglobal, 46
get_scenarios, 47

opt_design, 47, *47*

setup_app, 49
setup_bhm, 50, *55*
setup_binomial, 51
setup_cpp, 51
setup_cppglobal, 52
setup_cpplim, 53
setup_exnex, 54
setup_fujikawa, *10*, *24*, *35*, *43*, 55
setup_jsdglobal, 56
setup_mml, 57
setup_mmlglobal, 58

toer, *3*, 59

uniroot, *3*