Package: baskwrap (via r-universe)

May 20, 2026

Title Wrapper Package for Several Basket Trial R Packages

Version 1.0.3

Description A simple interface to switch between two methods for calculating basket trial characteristics, numerical integration (``exact'') and Monte Carlo simulation (``simulated'') for the basket trial design by Fujikawa et al. 2020 <doi:10.1002/bimj.201800404>. The exact implementation is from the 'baskexact' package, see Baumann (2024) <doi:10.1016/j.softx.2024.101793>. The simulated implementation is from the 'basksim' package, which was developed for Baumann et al. (2024) <doi:10.1080/19466315.2024.2402275>. The package's syntax is compatible with the 'basksim' syntax and easily extendable.

License MIT + file LICENSE

Encoding UTF-8

Roxygen list(markdown = TRUE)

RoxygenNote 7.3.3

Imports basksim (>= 0.1.0), baskexact

URL https://github.com/LukasDSauer/baskwrap

BugReports https://github.com/LukasDSauer/baskwrap/issues

Suggests testthat (>= 3.0.0), here, reticulate, ggplot2

Config/testthat/edition 3

Config/pak/sysreqs libgmp3-dev jags

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

Date/Publication 2026-03-20 09:28:14 UTC

RemoteUrl https://github.com/lukasdsauer/baskwrap

RemoteRef HEAD

RemoteSha 60835efed163dce0395f4fd00dfcda8aa35b6c8b

Contents

`basket_test` *Test for the results of a basket trial*

Description

Generic function for calculating the posterior probabilities of a basket trial design.

Usage

```
basket_test(design, ...)
```

Arguments

`design`	An object created with one of the `setup_` functions from the `basksim` package or the `baskwrap` package.
`...`	Further arguments.

Value

A numeric vector of posterior probabilities for all strata.

Examples

```
design_x <- setup_fujikawa_x(k = 3, p0 = 0.2, backend = "exact")
basket_test(design = design_x, n = 20, r = c(2, 7, 19), lambda = 0.95,
            epsilon = 2, tau = 0,
            logbase = exp(1))
```

`basket_test.fujikawa_x`

Test for the results of a basket trial

Description

This function is a wrapper of `baskexact::basket_test()`. It returns the posterior probabilities of a basket trial including borrowing.

Usage

```
## S3 method for class 'fujikawa_x'
basket_test(
  design,
  n,
  r,
  lambda,
  epsilon,
  tau,
  logbase = 2,
  weight_fun = weights_jsd,
  weight_params = list(epsilon = epsilon, tau = tau, logbase = logbase),
  ...
)
```

Arguments

`design`	An object of class `fujikawa_x`.
`n`	The sample size per basket.
`r`	A numeric vector of the number of responses in each stratum.
`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.
`weight_fun`	Which functions should be used to calculated the pairwise weights? Default is `weights_jsd`.
`weight_params`	A list of tuning parameters specific to `weight_fun`. By default, it takes the function arguments `epsilon`, `tau` and `logbase`.
`...`	Further arguments.

Value

A numeric vector of posterior probabilities for all strata.

Examples

```
design_x <- setup_fujikawa_x(k = 3, p0 = 0.2, backend = "exact")
basket_test(design = design_x, n = 20, r = c(2, 7, 19), lambda = 0.95,
            epsilon = 2, tau = 0,
            logbase = exp(1))
```

`check_mon_within` *Check Within- And Between-Trial Monotonicity Of A Basket Trial Design*

Description

Generic function for checking monotonicity conditions of a basket trial design. Currently only implemented for designs of class `fujikawa_x`. In that case, the functions are wrappers of `baskexact::check_mon_within()` and `baskexact::check_mon_between()`.

Usage

```
check_mon_within(design, ...)

check_mon_between(design, ...)

## S3 method for class 'fujikawa_x'
check_mon_within(
  design,
  n,
  lambda,
  weight_fun,
  weight_params = list(),
  globalweight_fun = NULL,
  globalweight_params = list(),
  details = TRUE,
  ...
)

## S3 method for class 'fujikawa_x'
check_mon_between(
  design,
  n,
  lambda,
  weight_fun,
  weight_params = list(),
  details = TRUE,
```

```
  globalweight_fun = NULL,
  globalweight_params = list(),
  ...
)
```

Arguments

design	An object created with one of the setup_ functions from the basksim package or the baskwrap package.
...	Further arguments.
n	The sample size per basket.
lambda	The posterior probability threshold.
weight_fun	Which function should be used to calculate the pairwise weights.
weight_params	A list of tuning parameters specific to weight_fun.
globalweight_fun	Which function should be used to calculate the global weights.
globalweight_params	A list of tuning parameters specific to globalweight_fun.
details	Whether the cases where the monotonicity condition is violated should be returned, in case there are any.

Details

Details on the within- and between-trial monotonicity conditions can be found in Baumann et al. 2022.

Value

If details = FALSE then only a logical value is returned. If details = TRUE then if there are any cases where the within-trial monotonicity condition is violated, a list of these cases and their results are returned. If at least one tuning parameter is a vector, then an array that shows for which combination of parameters the within-trial monotonicity condition holds. In this case, the argument details is ignored.

References

Baumann, L., Krisam, J., & Kieser, M. (2022). Monotonicity conditions for avoiding counterintuitive decisions in basket trials. Biometrical Journal, 64(5), 934-947.

Examples

```
design4 <- setup_fujikawa_x(k = 4, shape1 = 1, shape2 = 1, p0 = 0.2)
check_mon_within(design = design4, n = 15, lambda = 0.99,
                 weight_fun = baskexact::weights_fujikawa,
                 weight_params = list(epsilon = 0.5, tau = 0),
                 details = TRUE)
design3 <- setup_fujikawa_x(k = 3, shape1 = 1, shape2 = 1, p0 = 0.2)
check_mon_between(design = design3, n = 24, lambda = 0.99,
```

```
weight_fun = baskexact::weights_fujikawa,
weight_params = list(epsilon = c(0.5, 1),
                     tau = c(0, 0.2, 0.3)),
globalweight_fun = baskexact::globalweights_fix,
globalweight_params = list(w = c(0.5, 0.7)))
```

---

convert_to_fujikawa_x *Class conversions*

Description

These convenience functions can convert objects from the baskexact, basksim and baskwrap into one another.

Usage

```
convert_to_fujikawa_x(design)

convert_to_baskexact(design)

convert_to_basksim(design)
```

Arguments

design	An R object.

Details

convert_to_fujikawa_x() can currently convert objects of class "OneStageBasket" from the baskexact package to objects of class fujikawa_x. The functions convert_to_baskexact() and convert_to_basksim() can convert fujikawa_x objects to baskexact and basksim objects, respectively.

Value

An object of class fujikawa_x.

Examples

```
design <- baskexact::setupOneStageBasket(k = 3, p0 = 0.2)
design_fujx <- convert_to_fujikawa_x(design)
design_bsim <- convert_to_basksim(design_fujx)
# Below should be identical to initial design
design_bx <- convert_to_baskexact(design_fujx)
```

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

Description

Generic function for calculating the expected number of correct decisions of a basket trial design. It defaults to the function `basksim::ecd`.

Usage

```
ecd(design, ...)

## Default S3 method:
ecd(design, ...)
```

Arguments

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

Value

A numeric value.

Examples

```
design <- setup_fujikawa_x(k = 3, p0 = 0.2)
ecd(design = design, n = 20, p1 = c(0.2, 0.5, 0.5), lambda = 0.95,
     design_params = list(epsilon = 2, tau = 0), iter = 100)
```

ecd.fujikawa_x — *Calculate the Expected Number of Correct Decisions for Fujikawa et al.'s Basket Trial Design*

Description

This wrapper functions returns the expected number of correct decisions (ECD) for Fujikawa et al.'s basket trial design. The ECD is calculated using backends from two different R packages:

- If `design$backend == "sim"`, the ECD is calculated using `basksim::ecd`.
- If `design$backend == "exact"`, the ECD are calculated using `baskexact::ecd`.

Usage

```
## S3 method for class 'fujikawa_x'
ecd(
  design,
  n,
  p1,
  lambda,
  epsilon,
  tau,
  logbase = 2,
  design_params = list(epsilon = epsilon, tau = tau, logbase = logbase),
  iter = 1000,
  data = NULL,
  weight_fun = weights_jsd,
  weight_params = design_params,
  globalweight_fun = NULL,
  globalweight_params = list(),
  ...
)
```

Arguments

design	An object of class fujikawa_x.
n	The sample size per basket.
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.
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.
weight_fun	Which functions should be used to calculated the pairwise weights? Default is weights_jsd.
weight_params	A list of tuning parameters specific to weight_fun. By default, it takes the function arguments epsilon, tau and logbase.
globalweight_fun	Which functions should be used to calculated the global weights? Currently, this is only supported for the exact backend.
globalweight_params	A list of tuning parameters specific to globalweight_fun.
...	Further arguments.

Value

A numeric value.

Examples

```
design <- setup_fujikawa_x(k = 3, p0 = 0.2)
ecd(design = design, n = 20, p1 = c(0.2, 0.5, 0.5), lambda = 0.95,
     design_params = list(epsilon = 2, tau = 0), iter = 100)
```

`estim` *Calculate the Posterior Mean and Mean Squared Error for a Basket Trial Design*

Description

Generic function for calculating the posterior mean and mean squared error of a basket trial design. It defaults to the function `estim.default` which does not rely on any `baskwrap`-specific function.

Usage

```
estim(design, ...)

## Default S3 method:
estim(design, ...)

## S3 method for class 'fujikawa_x'
estim(
  design,
  n,
  p1,
  lambda = NULL,
  epsilon,
  tau,
  logbase = 2,
  iter = 1000,
  weight_fun = weights_jsd,
  weight_params = list(epsilon = epsilon, tau = tau, logbase = logbase),
  globalweight_fun = NULL,
  globalweight_params = list(),
  ...
)
```

Arguments

`design`	An object created with one of the `setup_` functions from the `basksim` package or the `baskwrap` package.
`...`	Further arguments.

n	The sample size per basket.
p1	Probabilities under the alternative hypothesis. If length(p1) == 1, then this is a common probability for all baskets.
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.
weight_fun	Which functions should be used to calculated the pairwise weights? Default is weights_jsd.
weight_params	A list of tuning parameters specific to weight_fun. By default, it takes the function arguments epsilon, tau and logbase.
globalweight_fun	Which functions should be used to calculated the global weights? Currently, this is only supported for the exact backend.
globalweight_params	A list of tuning parameters specific to globalweight_fun.

Details

estim.default is in fact just a wrapper of basksim::get_details() that select posterior mean and mean squared error.

Value

A list containing means of the posterior distribution and the mean squared errors for all baskets.

Examples

```
# Example for a basket trial with Fujikawa's Design
design <- setup_fujikawa_x(k = 3, p0 = 0.2)
estim(design = design, n = 20, p1 = c(0.2, 0.5, 0.5), lambda = 0.95,
     epsilon = 2, tau = 0, iter = 100)
```

---

get_details.fujikawa_x

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

---

Description

This wrapper functions returns details for basket trial design.

Usage

```
## S3 method for class 'fujikawa_x'
get_details(
  design,
  ...,
  n,
  p1 = NULL,
  lambda,
  level = 0.95,
  epsilon,
  tau,
  logbase = 2,
  iter = 1000,
  data = NULL,
  weight_fun = weights_jsd,
  weight_params = list(epsilon = epsilon, tau = tau, logbase = logbase),
  globalweight_fun = NULL,
  globalweight_params = list(),
  which_details = "all",
  verbose = TRUE
)
```

Arguments

design	An object of class fujikawa_x.
...	Further arguments.
n	The sample size per basket.
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.
weight_fun	Which functions should be used to calculated the pairwise weights? Default is weights_jsd.
weight_params	A list of tuning parameters specific to weight_fun. By default, it takes the function arguments epsilon, tau and logbase.
globalweight_fun	Which functions should be used to calculated the global weights? Currently, this is only supported for the exact backend.

`globalweight_params`
: A list of tuning parameters specific to `globalweight_fun`.

`which_details`
: A character vector specifying which details should be calculated. This is used only for `backend = "exact"`, where the number of details is relevant for runtime. Default is `"all"`, see details for explanation.

`verbose`
: A logical, should message be shown if EWP or FWER is 0. Default is `TRUE`.

Details

It calculates the details using backends from two different R packages:

- If `design$backend == "sim"`, the details are calculated using `basksim::get_details.fujikawa`.
- If `design$backend == "exact"`, the details are calculated using `baskexact::toer`, `baskexact::pow` and `baskexact::estim`. Note that the standard weight function `weights_jsd` calculates the weights anew for each of the three function calls. This may compromise performance and can be fixed by manually calculating the weights beforehand.

For the `baskexact` backend, the number of details is a relevant factor for the function's runtime. Hence, one can select precisely which details should be calculated:

- If `which_details == "all"`, everything will be calculated.
- If `"FWER" %in% which_details`, then FWER will be calculated.
- If `"EWP" %in% which_details`, then EWP will be calculated.
- If `"Rejection_Probabilities" %in% which_details`, then per-basket rejection probabilities will be calculated.
- If `"ECD" %in% which_details`, then ECD will be calculated.
- If `"Mean" %in% which_details`, then mean response rate and its MSE will be calculated. The mean is the expected posterior mean conditional under the assumption that `p1` is true, and the MSE is the expected squared deviation of the posterior mean from this true value.

Value

A list containing the rejection probabilites, 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_x(k = 3, p0 = 0.2)
get_details(design = design, n = 20, p1 = c(0.2, 0.5, 0.5), lambda = 0.95,
            epsilon = 2, tau = 0, iter = 100)
design_x <- setup_fujikawa_x(k = 3, p0 = 0.2, backend = "exact")
get_details(design = design_x, n = 20, p1 = c(0.2, 0.5, 0.5), lambda = 0.95,
            epsilon = 2, tau = 0, weight_fun = baskexact::weights_fujikawa,
            logbase = exp(1))
# If you call get_details() with backend = "exact" multiple without
# changing design and n, it can make sense to save the weights and supply
# them separately using a custom function. This can save run time.
weight_mat_vanilla <- weights_jsd_vanilla(design_x, n = 20,
                                          logbase = exp(1))
```

```
weights_from_save <- function(epsilon,
                              tau,
                              ...) {
  return(weights_fujikawa_tuned(weight_mat_vanilla,
                                epsilon = epsilon,
                                tau = tau))
}
get_details(design = design_x,
            n = 20,
            p1 = c(0.2, 0.5, 0.5),
            lambda = 0.95,
            epsilon = 2, tau = 0,
            weight_fun = weights_from_save,
            logbase = NULL)
```

`get_scenarios` *Generate a matrix of default outcome scenarios*

Description

This functions generates a matrix of k+1 default outcome scenarios, ranging from no responsive strata (global null scenarios) to all responsive strata (global alternative scenario). It is a wrapper of the function `basksim::get_scenarios`.

Usage

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

Arguments

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

Details

No backend switching is implemented in this function because `baskexact::get_scenarios()` does precisely the same thing as `basksim::get_scenarios()`.

Value

A matrix with k rows and k + 1 columns.

Examples

```
# Example for a basket trial with Fujikawa's Design
design <- setup_fujikawa_x(k = 3, p0 = 0.2)
get_scenarios(design, p1 = 0.5)
```

is_baskexact_design *Check whether an R object is a* baskexact *design.*

Description

Check whether an R object is a baskexact design.

Usage

```
is_baskexact_design(design, baskexact_class)
```

Arguments

design	An R object.
baskexact_class	A character string to specify the respective baskexact class, e.g. "OneStageBasket" or "TwoStageBasket".

Value

A logical.

Examples

```
design <- baskexact::setupOneStageBasket(k = 3, p0 = 0.2)
is_baskexact_design(design, "OneStageBasket")
```

opt_design *Optimize a Basket Trial Design*

Description

Functions for optimizing the tuning parameters of a basket trial design. This defaults to the function basksim::opt_design.

Usage

```
opt_design(design, ...)

## Default S3 method:
opt_design(design, ...)
```

Arguments

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

Value

A matrix with the expected number of correct decisions.

Examples

```
design <- setup_fujikawa_x(k = 3, p0 = 0.2)
# In an actual application, usually increase to at least iter = 1000.
opt_design(design = design,
           n = 20, alpha = 0.05,
           design_params = list(epsilon = c(1, 2), tau = c(0, 0.5)),
           scenarios = get_scenarios(design, 0.5),
           prec_digits = 3,
           iter = 100)
```

---

opt_design.fujikawa_x *Optimize Fujikawa et al.'s Basket Trial Design*

Description

This wrapper function returns the optimal tuning parameters for Fujikawa et al.'s basket trial design. The design is optimized using backends from two different R packages:

- If `design$backend == "sim"`, the opt_design is calculated using `basksim::opt_design`.
- If `design$backend == "exact"`, the opt_design are calculated using `baskexact::opt_design`.

Usage

```
## S3 method for class 'fujikawa_x'
opt_design(
  design,
  n,
  alpha,
  design_params = list(),
  scenarios,
  prec_digits,
  iter = 1000,
  data = NULL,
  weight_fun = weights_jsd,
  weight_params = design_params,
  globalweight_fun = NULL,
  globalweight_params = list(),
  ...
)
```

Arguments

design	An object of class fujikawa_x.
n	The sample size per basket.
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 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.
weight_fun	Which functions should be used to calculated the pairwise weights? Default is weights_jsd.
weight_params	A list of tuning parameters specific to weight_fun. By default, it takes the function arguments epsilon, tau and logbase.
globalweight_fun	Which functions should be used to calculated the global weights? Currently, this is only supported for the exact backend.
globalweight_params	A list of tuning parameters specific to globalweight_fun.
...	Further arguments.

Value

A matrix with the expected number of correct decisions.

Examples

```
design <- setup_fujikawa_x(k = 3, p0 = 0.2)
# In an actual application, usually increase to at least iter = 1000.
opt_design(design = design,
           n = 20, alpha = 0.05,
           design_params = list(epsilon = c(1, 2), tau = c(0, 0.5)),
           scenarios = get_scenarios(design, 0.5),
           prec_digits = 3,
           iter = 100)
```

plot_weights *Plot Weights of a Basket Trial Design*

Description

Generic function for plotting the weights of a basket trial design. Currently only implemented for designs of class `fujikawa_x`.

Usage

```
plot_weights(design, ...)
```

Arguments

`design`	An object created with one of the `setup_` functions from the `basksim` package or the `baskwrap` package.
`...`	Further arguments.

Value

A `ggplot` object, a plot of the weights.

Examples

```
design <- setup_fujikawa_x(k = 3, p0 = 0.2, backend = "exact")
# Default weight function is weights_jsd, which is identical
# to the Jensen-Shannon weights
plot_weights(design = design, n = 20, r1 = 11,
             weight_params = list(tau = 0, epsilon = c(0.25, 0.5, 1, 2),
                                  logbase = 2))
# Explicitly compare Jensen-Shannon and Hellinger weights
plot_weights(design = design, n = 20, r1 = 11,
             weight_fun = baskexact::weights_fujikawa,
             weight_params = list(tau = 0, epsilon = c(0.25, 0.5, 1, 2),
                                  logbase = 2))
plot_weights(design = design, n = 20, r1 = 11, weight_fun = weights_hld,
             weight_params = list(tau = 0, epsilon = c(0.25, 0.5, 1, 2),
                                  logbase = 2))
```

plot_weights.fujikawa_x *Plot Weight Functions of Fujikawa et al.'s Basket Trial Design*

Description

This function is a wrapper of `baskexact::plot_weights()`. It visualizes the weight functions defined for Fujikawa et al.'s design.

Usage

```
## S3 method for class 'fujikawa_x'
plot_weights(
  design,
  n,
  r1,
  weight_fun = weights_jsd,
  weight_params = list(),
  ...
)
```

Arguments

`design`	An object created with one of the `setup_` functions from the `basksim` package or the `baskwrap` package.
`n`	The sample size per basket.
`r1`	Number of responses in one basket
`weight_fun`	Which function should be used to calculate the pairwise weights.
`weight_params`	A list of tuning parameters specific to `weight_fun`.
`...`	Further arguments.

Value

A `ggplot` object, showing the range of responses in the "other" basket on the x-axis and the corresponding weight on the y-axis.

Examples

```
design <- setup_fujikawa_x(k = 3, p0 = 0.2, backend = "exact")
# Default weight function is weights_jsd, which is identical
# to the Jensen-Shannon weights
plot_weights(design = design, n = 20, r1 = 11,
             weight_params = list(tau = 0, epsilon = c(0.25, 0.5, 1, 2),
                                  logbase = 2))
# Explicitly compare Jensen-Shannon and Hellinger weights
plot_weights(design = design, n = 20, r1 = 11,
```

```
              weight_fun = baskexact::weights_fujikawa,
              weight_params = list(tau = 0, epsilon = c(0.25, 0.5, 1, 2),
                                   logbase = 2))
plot_weights(design = design, n = 20, r1 = 11, weight_fun = weights_hld,
              weight_params = list(tau = 0, epsilon = c(0.25, 0.5, 1, 2),
                                   logbase = 2))
```

---

pow *Calculate the Power for a Basket Trial Design*

Description

Generic function for calculating the power of a basket trial design. It defaults to the function `pow.default` which works similarly to `basksim::toer` and does not rely on any `baskwrap`-specific function.

Usage

```
pow(design, ...)

## Default S3 method:
pow(
  design,
  n,
  p1 = NULL,
  lambda,
  design_params = list(),
  iter = 1000,
  data = NULL,
  ...
)
```

Arguments

`design`	An object created with one of the `setup_` functions from the `basksim` package or the `baskwrap` package.
`...`	Further arguments.
`n`	The sample size per basket.
`p1`	Probabilities under the alternative hypothesis. If `length(p1) == 1`, then this is a common probability for all baskets.
`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.

Value

A numeric value.

Examples

```
design <- setup_fujikawa_x(k = 3, p0 = 0.2)
pow(design = design, n = 20, p1 = c(0.2, 0.5, 0.5), lambda = 0.95,
     design_params = list(epsilon = 2, tau = 0), iter = 100)
```

`pow.fujikawa_x` *Calculate the Power for a Fujikawa et al.'s Basket Trial Design*

Description

This wrapper functions returns the power for Fujikawa et al.'s basket trial design. The power is calculated using backends from two different R packages:

- If `design$backend == "sim"`, the power is calculated using `basksim::pow`.
- If `design$backend == "exact"`, the power is calculated using `baskexact::pow`.

Usage

```
## S3 method for class 'fujikawa_x'
pow(
  design,
  n,
  p1 = NULL,
  lambda,
  epsilon,
  tau,
  logbase = 2,
  design_params = list(epsilon = epsilon, tau = tau, logbase = logbase),
  iter = 1000,
  data = NULL,
  weight_fun = weights_jsd,
  weight_params = design_params,
  globalweight_fun = NULL,
  globalweight_params = list(),
  results = c("ewp", "group"),
  ...
)
```

Arguments

`design`	An object of class `fujikawa_x`.
`n`	The sample size per basket.
`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.
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.
weight_fun	Which functions should be used to calculated the pairwise weights? Default is weights_jsd.
weight_params	A list of tuning parameters specific to weight_fun. By default, it takes the function arguments epsilon, tau and logbase.
globalweight_fun	Which functions should be used to calculated the global weights? Currently, this is only supported for the exact backend.
globalweight_params	A list of tuning parameters specific to globalweight_fun.
results	Whether only the experimentwise power (option ewp) or also the rejection probabilities per group (option group) should be returned.
...	Further arguments.

Value

A numeric value.

Examples

```
design <- setup_fujikawa_x(k = 3, p0 = 0.2)
pow(design = design, n = 20, p1 = c(0.2, 0.5, 0.5), lambda = 0.95,
     design_params = list(epsilon = 2, tau = 0), iter = 100)
```

---

set_backend *Set the backend of a Fujikawa design*

Description

For a basket trial design of class fujikawa_x, set the backend.

Usage

```
set_backend(design, backend)
```

Arguments

design	An object of class fujikawa_x, i.e. a Fujikawa basket trial design.
backend	A string, either "sim" or "exact" for specifying the use of the basksim package or the baskexact package for calculation of basket trial details.

Value

An object of class fujikawa_x.

Examples

```
design <- setup_fujikawa_x(k = 3, p0 = 0.2)
design <- set_backend(design, backend = "exact")
```

---

set_design_exact *Set baskexact design object*

Description

For a given object of class fujikawa_x, this function sets the attribute fujikawa_x$design_exact to baskexact::OneStageBasket object with the same attributes as the given design. The attribute will then be used in internal calls to baskexact.

Usage

```
set_design_exact(design)
```

Arguments

design	An object of class fujikawa_x, i.e. a Fujikawa basket trial design.

Value

An object of class fujikawa_x.

setup_fujikawa_x *Set up a Fujikawa design object with flexible backend*

Description

The fujikawa_x S3 class is similar to the basksim::fujikawa class, but it has an additional parameter backend, that allows users to decide which backend should be used for calculation of details (i.e. rejection rates, power, type-I error rate etc.): the basksim package or the baskexact package.

Usage

```
setup_fujikawa_x(k, p0, shape1 = 1, shape2 = 1, backend = "sim")
```

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.
backend	A string, either "sim" or "exact" for specifying the use of the basksim package or the baskexact package for calculation of basket trial details.

Value

An object of class fujikawa_x.

Examples

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

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

Description

Generic function for calculating the type-1 error rate of a basket trial design. It defaults to the function basksim::toer.

Usage

```
toer(design, ...)

## Default S3 method:
toer(design, ...)
```

Arguments

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

Value

A numeric value.

Examples

```
design <- setup_fujikawa_x(k = 3, p0 = 0.2)
toer(design = design, n = 20, p1 = c(0.2, 0.5, 0.5), lambda = 0.95,
     design_params = list(epsilon = 2, tau = 0), iter = 100)
```

---

toer.fujikawa_x *Calculate the Type 1 Error Rate for a Fujikawa et al.'s Basket Trial Design*

Description

This wrapper functions returns the type-1 error rate (TOER) for Fujikawa et al.'s basket trial design. The TOER is calculated using backends from two different R packages:

- If design$backend == "sim", the TOER is calculated using basksim::toer.
- If design$backend == "exact", the TOER are calculated using baskexact::toer.

Usage

```
## S3 method for class 'fujikawa_x'
toer(
  design,
  n,
  p1 = NULL,
  lambda,
  epsilon = epsilon,
  tau = tau,
  logbase = logbase,
  design_params = list(epsilon = epsilon, tau = tau, logbase = logbase),
  iter = 1000,
  data = NULL,
  weight_fun = weights_jsd,
  weight_params = design_params,
  globalweight_fun = NULL,
  globalweight_params = list(),
  results = c("fwer", "group"),
  ...
)
```

Arguments

design	An object of class fujikawa_x.
n	The sample size per basket.
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.
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.
weight_fun	Which functions should be used to calculated the pairwise weights? Default is weights_jsd.
weight_params	A list of tuning parameters specific to weight_fun. By default, it takes the function arguments epsilon, tau and logbase.
globalweight_fun	Which functions should be used to calculated the global weights? Currently, this is only supported for the exact backend.
globalweight_params	A list of tuning parameters specific to globalweight_fun.
results	Whether only the family wise error rate (option fwer) or also the rejection probabilities per group (option group) should be returned.
...	Further arguments.

Value

A numeric value.

Examples

```
design <- setup_fujikawa_x(k = 3, p0 = 0.2)
toer(design = design, n = 20, p1 = c(0.2, 0.5, 0.5), lambda = 0.95,
     design_params = list(epsilon = 2, tau = 0), iter = 100)
```

weights_jsd *Further weight functions*

Description

A couple of weight functions additional to the ones implemented in baskexact are supplied. The weight functions are based on the Jensen-Shannon divergence (JSD) and the Hellinger distance (HLD). The function weights_jsd is a wrapper of baskexact::weights_fujikawa. It can be used with both designs of class fujikawa_x (from the baskwrap package) and designs of class OneStageBasket (from the baskexact package). The function weights_jsd_vanilla is a convenience wrapper that calls this with epsilon = 1 and tau = 0 without pruning. Hence, this function returns precisely Fujikawa et al.'s weights without any tuning. The function weights_fujikawa_tuned tunes an existing weight matrix using the parameters epsilon and tau in accordance with Fujikawa et al.'s tuning rules. The function weights_hld and the "convenience wrapper" weights_hld_vanilla are a variant of Fujikawa's weights where the similarity is calculated using 1 minus Hellinger distance instead of 1 minus Jensen-Shannon divergence (see Details).

Usage

```
weights_jsd(design, n, logbase, epsilon, tau, lambda = NULL, ...)

weights_jsd_vanilla(design, n, logbase, ...)

weights_fujikawa_tuned(weight_mat, epsilon = 1.25, tau = 0.5, ...)

weights_hld_vanilla(design, n, ...)

weights_hld(design, n, epsilon, tau, ...)
```

Arguments

design	An object of class fujikawa_x or of class OneStageBasket from the baskexact package.
n	The sample size per basket.
logbase	Tuning parameter. The base of the logarithm that is used to calculate the Jensen-Shannon divergence.
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).
lambda	The posterior probability threshold, currently only used for designs with "exact" backend where pruning is activated. See documentation of baskexact::weights_fujikawa for more information.
...	Further arguments.
weight_mat	An untuned matrix including the weights of all possible pairwise outcomes.

Details

For posterior beta distributions as in Fujikawa's design, the Hellinger distance can be calculated "analytically", e.g. for posterior parameters (a_1, b_1) and (a_2, b_2), we have

$$HLD(\text{Beta}(a_1, b_1), \text{Beta}(a_2, b_2)) = 1 - \frac{B(\frac{a_1+a_2}{2}, \frac{b_1+b_2}{2})}{\sqrt{B(a_1, b_1)B(a_2, b_2)}},$$

where $B(\cdot, \cdot)$ is the beta function (Sasha 2012). The similarity between strata is calculated as $1 - HLD(\cdot, \cdot)$.

Value

A matrix including the weights of all possible pairwise outcomes.

References

Sasha. Answer to "Hellinger distance between Beta distributions"; 2012. Available from: https://math.stackexchange.com/a/1

Examples

```
design <- setup_fujikawa_x(k = 3, p0 = 0.2, backend = "exact")
weight_mat <- weights_jsd_vanilla(design, n = 20, logbase = 2)
weight_mat_tuned <- weights_fujikawa_tuned(weight_mat, epsilon = 1.25,
                                           tau = 0.5)
# In theory, this weights_function is also compatible with baskexact.
baskexact::toer(design$design_exact, n = 20,
                lambda = 0.95, weight_fun = weights_jsd,
                weight_params = list(epsilon = 2,
                                     tau = 0,
                                     logbase = 2))
# Use different function in get_details
get_details(design = design, n = 20, p1 = c(0.2, 0.5, 0.5), lambda = 0.95,
            epsilon = 2, tau = 0, weight_fun = weights_jsd,
            logbase = exp(1))
get_details(design = design, n = 20, p1 = c(0.2, 0.5, 0.5), lambda = 0.95,
            epsilon = 2, tau = 0, weight_fun = weights_hld,
            logbase = exp(1))
```

Index