Package 'baskexact'

September 15, 2021

Type Package

Title Exact Calculation of Basket Trial Operating Characteristics

Version 0.1.0

Description Calculates the exact operating characteristics of a single-stage basket trial with the design of Fujikawa, K., Teramukai, S., Yokota, I., & Daimon, T. (2020). <doi:10.1002/bimj.201800404>.

License GPL-3

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

Encoding UTF-8

RoxygenNote 7.1.1

LinkingTo Rcpp, RcppArmadillo

Imports Rcpp, arrangements, methods

Suggests testthat (>= 3.0.0), covr

Config/testthat/edition 3

Collate 'RcppExports.R' 'class.R' 'generics.R' 'adjust_lambda.R' 'analysis.R' 'basket_test.R' 'baskexact-package.R' 'borrowing.R' 'check.R' 'helper.R' 'monotonicity.R' 'pow.R' 'rejection_probabilities.R' 'toer.R' 'validate.R'

NeedsCompilation yes

Author Lukas Baumann [aut, cre] (<https://orcid.org/0000-0001-7931-7470>)

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

Repository CRAN

Date/Publication 2021-09-15 18:40:05 UTC

R topics documented:

adjust_lambda 2
basket_test 3
check_mon_between 5
check_mon_within 6

OneStageBasket-class 8
pow 9
setupOneStageBasket 11
toer 11

Index **14**

`adjust_lambda` *Adjust Lambda*

Description

Finds the value for `lambda` such that the family wise error rate is protected at level `alpha`.

Usage

```
adjust_lambda(
  design,
  alpha = 0.025,
  theta1 = NULL,
  n,
  epsilon,
  tau,
  logbase,
  prune,
  prec_digits,
  ...
)

## S4 method for signature 'OneStageBasket'
adjust_lambda(
  design,
  alpha = 0.025,
  theta1 = NULL,
  n,
  epsilon,
  tau,
  logbase,
  prune = FALSE,
  prec_digits,
  ...
)
```

Arguments

`design`	An object of class `Basket` created by `setupBasket`.
`alpha`	The one-sided signifance level.

`theta1`	Probabilities under the alternative hypothesis. If `length(theta1) == 1`, then this is a common probability for all baskets. If `is.null(theta1)` then the type 1 error rate under the global null hypothesis is computed.
`n`	The sample size per basket.
`epsilon`	A tuning parameter that determines the amount of borrowing. See details for more information.
`tau`	A tuning parameter that determines how similar the baskets have to be that borrowing occurs. See details for more information.
`logbase`	A tuning parameter that determines which logarithm base is used to compute the Jensen-Shannon divergence. See details for more information.
`prune`	Whether baskets with a number of responses below the critical pooled value should be pruned before the final analysis.
`prec_digits`	Number of decimal places that are considered when adjusting lambda
`...`	Further arguments.

Details

`adjust_alpha` finds the greatest value with `prec_digits` for `lambda` which controls the family wise error rate at level `alpha` (one-sided). A combination of the uniroot function followed by a grid search is used to finde the correct value for `lambda`.

This method is implemented for the class `OneStageBasket`.

Value

The greatest value with `prec_digits` decimal places for `lambda` which controls the family wise error rate at level `alpha` (one-sided) and the exact family wise error rate for this value of `lambda`.

Methods (by class)

- `OneStageBasket`: Adjust lambda for a single-stage design.

Examples

```
design <- setupOneStageBasket(k = 3, shape1 = 1, shape2 = 1, theta0 = 0.2)
adjust_lambda(design = design, alpha = 0.025, n = 15, epsilon = 1, tau = 0,
  logbase = 2, prune = FALSE, prec_digits = 4)
```

`basket_test` *Test for the Results of a Basket Trial*

Description

`basket_test` evaluates the results of a basket trial and calculates the posterior distributions with and without borrowing.

Usage

```
basket_test(design, n, r, lambda, epsilon, tau, logbase = 2, prune, ...)

## S4 method for signature 'OneStageBasket'
basket_test(design, n, r, lambda, epsilon, tau, logbase = 2, prune, ...)
```

Arguments

`design`	An object of class `Basket` created by `setupBasket`.
`n`	The sample size per basket.
`r`	The vector of observed responses.
`lambda`	The posterior probability threshold. See details for more information.
`epsilon`	A tuning parameter that determines the amount of borrowing. See details for more information.
`tau`	A tuning parameter that determines how similar the baskets have to be that borrowing occurs. See details for more information.
`logbase`	A tuning parameter that determines which logarithm base is used to compute the Jensen-Shannon divergence. See details for more information.
`prune`	Whether baskets with a number of responses below the critical pooled value should be pruned before the final analysis.
`...`	Further arguments.

Value

A list, including matrices of the weights that are used for borrowing, posterior distribution parameters for all baskets without and with borrowing, as well as the posterior probabilities for all baskets without and with borrowing.

Methods (by class)

- `OneStageBasket`: Testing for a single-stage basket design.

Examples

```
design <- setupOneStageBasket(k = 3, shape1 = 1, shape2 = 1, theta0 = 0.2)
basket_test(design = design, n = 24, r = c(5, 9, 10), lambda = 0.99,
  epsilon = 1, tau = 0, logbase = 2, prune = FALSE)
```

`check_mon_between` *Check Between-Trial Monotonicity*

Description

Checks whether the between-trial monotonicity condition holds.

Usage

```
check_mon_between(
  design,
  n,
  lambda,
  epsilon,
  tau,
  logbase = 2,
  prune,
  details,
  ...
)

## S4 method for signature 'OneStageBasket'
check_mon_between(
  design,
  n,
  lambda,
  epsilon,
  tau,
  logbase = 2,
  prune,
  details,
  ...
)
```

Arguments

`design`	An object of class `Basket` created by `setupBasket`.
`n`	The sample size per basket.
`lambda`	The posterior probability threshold. See details for more information.
`epsilon`	A tuning parameter that determines the amount of borrowing. See details for more information.
`tau`	A tuning parameter that determines how similar the baskets have to be that borrowing occurs. See details for more information.
`logbase`	A tuning parameter that determines which logarithm base is used to compute the Jensen-Shannon divergence. See details for more information.

`prune`	Whether baskets with a number of responses below the critical pooled value should be pruned before the final analysis.
`details`	Whether the cases where the monotonicity condition is violated should be returned, in case there are any.
`...`	Further arguments.

Details

`check_mon_between` checks whether the between-trial monotonicity condition holds. For a single-stage design with equal prior distributions and equal sample sizes for each basket this condition states that there are no cases where at least one null hypothesis is rejected when when there is a case with an equal or higher number of responses in each basket for which no null hypothesis is rejected.

If `prune = TRUE` then the baskets with an observed number of baskets smaller than the pooled critical value are not borrowed from. The pooled critical value is the smallest integer c for which all null hypotheses can be rejected if the number of responses is exactly c for all baskets.

This method is implemented for the class `OneStageBasket`.

Value

If `details = FALSE` then only a logical value is returned. If `details = TRUE` then if there are any cases where the between-trial monotonicity condition is violated, a list of theses cases and their results are returned.

Methods (by class)

- `OneStageBasket`: Between-trial monotonicity condition for a single-stage design.

Examples

```
design <- setupOneStageBasket(k = 4, shape1 = 1, shape2 = 1, theta0 = 0.2)
check_mon_between(design = design, n = 24, lambda = 0.99, epsilon = 3,
  tau = 0, prune = FALSE, details = TRUE)
```

`check_mon_within` *Check Within-Trial Monotonicity*

Description

Checks whether the within-trial monotonicity condition holds.

Usage

```
check_mon_within(
  design,
  n,
  lambda,
  epsilon,
  tau,
  logbase = 2,
  prune,
  details,
  ...
)

## S4 method for signature 'OneStageBasket'
check_mon_within(
  design,
  n,
  lambda,
  epsilon,
  tau,
  logbase = 2,
  prune,
  details,
  ...
)
```

Arguments

`design`	An object of class `Basket` created by `setupBasket`.
`n`	The sample size per basket.
`lambda`	The posterior probability threshold. See details for more information.
`epsilon`	A tuning parameter that determines the amount of borrowing. See details for more information.
`tau`	A tuning parameter that determines how similar the baskets have to be that borrowing occurs. See details for more information.
`logbase`	A tuning parameter that determines which logarithm base is used to compute the Jensen-Shannon divergence. See details for more information.
`prune`	Whether baskets with a number of responses below the critical pooled value should be pruned before the final analysis.
`details`	Whether the cases where the monotonicity condition is violated should be returned, in case there are any.
`...`	Further arguments.

Details

`check_mon_within` checks whether the within-trial monotonicity condition holds. For a single-stage design with equal prior distributions and equal sample sizes for each basket this condition states that there are no cases where the null hypothesis of a basket is rejected when there is at least one other basket with more observed responses for which the null hypothesis cannot be rejected.

If `prune = TRUE` then the baskets with an observed number of baskets smaller than the pooled critical value are not borrowed from. The pooled critical value is the smallest integer c for which all null hypotheses can be rejected if the number of responses is exactly c for all baskets.

This method is implemented for the class `OneStageBasket`.

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.

Methods (by class)

- `OneStageBasket`: Within-trial monotonicity condition for a single-stage design.

Examples

```
design <- setupOneStageBasket(k = 4, shape1 = 1, shape2 = 1, theta0 = 0.2)
check_mon_within(design = design, n = 24, lambda = 0.99, epsilon = 0.5,
  tau = 0, prune = FALSE, details = TRUE)
```

`OneStageBasket-class` *Class OneStageBasket*

Description

OneStageBasket is an S4 class. An object of this class contains the most important design features of a single-stage basket trial.

Details

This class implements a single-stage basket trial based on the design proposed by Fujikawa et al. In this design, at first separate posterior distributions are calculated for each basket based on a beta-binomial model. Information is then borrowed between baskets by calculating weights that reflect the similarity between the basket and computing posterior distributions for each basket where the parameters of the beta posterior are calculated as a weighted sum of the individual posterior distributions. The weight between two baskets i and j is found as (1 - JSD(i, j))^epsilon where JSD(i, j) is the Jensen-Shannon divergence between basket i and j. A small value of epsilon results in stronger borrowing also across baskets with heterogenous results. If epsilon is large then information is only borrowed between baskets with similar results. If a weight is smaller than tau it is set to 0, which results in no borrowing. If for a basket the posterior probability that θ > theta0 is greater than lambda, then the null hypothesis is rejected.

Currently only common prior distributions and a common null hypothesis are supported.

Slots

`k` The number of baskets.

`shape1` First common shape parameter of the beta prior.

`shape2` Second common shape parameter of the beta prior.

`theta0` A common probability under the null hypothesis.

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.

`pow` *Power*

Description

Computes the exact power for a basket trial.

Usage

```
pow(
  design,
  theta1,
  n,
  lambda,
  epsilon,
  tau,
  logbase = 2,
  prune = FALSE,
  results = c("ewp", "group"),
  ...
)

## S4 method for signature 'OneStageBasket'
pow(
  design,
  theta1,
  n,
  lambda,
  epsilon,
  tau,
  logbase = 2,
  prune = FALSE,
  results = c("ewp", "group"),
  ...
)
```

Arguments

`design`	An object of class `Basket` created by `setupBasket`.
`theta1`	Probabilities under the alternative hypothesis. If `length(theta1) == 1`, then this is a common probability for all baskets.
`n`	The sample size per basket.
`lambda`	The posterior probability threshold. See details for more information.
`epsilon`	A tuning parameter that determines the amount of borrowing. See details for more information.
`tau`	A tuning parameter that determines how similar the baskets have to be that borrowing occurs. See details for more information.
`logbase`	A tuning parameter that determines which logarithm base is used to compute the Jensen-Shannon divergence. See details for more information.
`prune`	Whether baskets with a number of responses below the critical pooled value should be pruned before the final analysis.
`results`	Whether only the experimentwise power (option `ewp`) or also the rejection probabilities per group (option `group`) should be returned.
`...`	Further arguments.

Details

`pow` computes the exact experimentwise power and the exact rejection probabilities per group. The experimentwise power is the probability to reject at least one null hypothesis for a basket with theta1 > theta0. The rejection probabilities correspond to the type 1 error rate for baskets with theta1 = theta 0 and to the power for baskets with theta1 > theta 0.

If `prune = TRUE` then the baskets with an observed number of baskets smaller than the pooled critical value are not borrowed from. The pooled critical value is the smallest integer c for which all null hypotheses can be rejected if the number of responses is exactly c for all baskets.

This method is implemented for the class `OneStageBasket`.

Value

If `results = "ewp"` then the experimentwise power is returned as a numeric value. If `results = "group"` then a list with the rejection probabilities per group and the experimentwise power is returned. For baskets with theta1 = theta0 the rejection probabilities corresponds to the type 1 error rate, for baskets with theta1 > theta0 the rejection probabilities corresponds to the power.

Methods (by class)

- `OneStageBasket`: Power for a single-stage basket design.

Examples

```
design <- setupOneStageBasket(k = 3, theta0 = 0.2)
pow(design, theta1 = c(0.2, 0.5, 0.5), n = 15, lambda = 0.99, epsilon = 2,
  tau = 0)
```

`setupOneStageBasket` *Setup OneStageBasket*

Description

Creates an object of class `OneStageBasket`.

Usage

```
setupOneStageBasket(k, shape1 = 1, shape2 = 1, theta0)
```

Arguments

`k`	The number of baskets.
`shape1`	First common shape parameter of the beta prior.
`shape2`	Second common shape parameter of the beta prior.
`theta0`	A common probability under the null hypothesis.

Details

A `OneStageBasket` object contains the most important design features of a basket trial. Currently only common prior distributions and a common null hypothesis are supported.

Value

An S4 object of class `OneStageBasket`.

Examples

```
design <- setupOneStageBasket(k = 3, theta0 = 0.2)
```

`toer` *Type 1 Error Rate*

Description

Computes the exact family wise type 1 error rate of a basket trial .

Usage

```
toer(
  design,
  theta1 = NULL,
  n,
  lambda,
  epsilon,
  tau,
  logbase = 2,
  prune = FALSE,
  results = c("fwer", "group"),
  ...
)

## S4 method for signature 'OneStageBasket'
toer(
  design,
  theta1 = NULL,
  n,
  lambda,
  epsilon,
  tau,
  logbase = 2,
  prune = FALSE,
  results = c("fwer", "group"),
  ...
)
```

Arguments

`design`	An object of class `Basket` created by `setupBasket`.
`theta1`	Probabilities under the alternative hypothesis. If `length(theta1) == 1`, then this is a common probability for all baskets. If `is.null(theta1)` then the type 1 error rate under the global null hypothesis is computed.
`n`	The sample size per basket.
`lambda`	The posterior probability threshold. See details for more information.
`epsilon`	A tuning parameter that determines the amount of borrowing. See details for more information.
`tau`	A tuning parameter that determines how similar the baskets have to be that borrowing occurs. See details for more information.
`logbase`	A tuning parameter that determines which logarithm base is used to compute the Jensen-Shannon divergence. See details for more information.
`prune`	Whether baskets with a number of responses below the critical pooled value should be pruned before the final analysis.
`results`	Whether only the family wise error rate (option `fwer`) or also the rejection probabilities per group (option `group`) should be returned.
`...`	Further arguments.

Details

`toer` computes the exact family wise type 1 error rate and the exact rejection probabilities per group. The family wise type 1 error rate is the probability to reject at least one null hypothesis for a basket with theta1 = theta0. If all theta1 > theta0 then the family wise type 1 error rate under the global null hypothesis is computed. The rejection probabilities correspond to the type 1 error rate for baskets with theta1 = theta 0 and to the power for baskets with theta1 > theta 0.

If `prune = TRUE` then the baskets with an observed number of baskets smaller than the pooled critical value are not borrowed from. The pooled critical value is the smallest integer c for which all null hypotheses can be rejected if the number of responses is exactly c for all baskets.

This method is implemented for the class `OneStageBasket`.

Value

If `results = "fwer"` then the family wise type 1 error rate is returned as a numeric value. If `results = "group"` then a list with the rejection probabilities per group and the family wise type 1 error rate is returned. If all theta1 > theta0 then the family wise type 1 error rate is calculated under the global null hypothesis. For baskets with theta1 = theta0 the rejection probabilities corresponds to the type 1 error rate, for baskets with theta1 > theta0 the rejection probabilities corresponds to the power.

Methods (by class)

- `OneStageBasket`: Type 1 error rate for a single-stage basket design.

Examples

```
design <- setupOneStageBasket(k = 3, theta0 = 0.2)
toer(design, n = 15, lambda = 0.99, epsilon = 2, tau = 0)
```

Index

adjust_lambda, 2
adjust_lambda,OneStageBasket-method
    *(*adjust_lambda*)*, 2

basket_test, 3
basket_test,OneStageBasket-method
    *(*basket_test*)*, 3

check_mon_between, 5
check_mon_between,OneStageBasket-method
    *(*check_mon_between*)*, 5
check_mon_within, 6
check_mon_within,OneStageBasket-method
    *(*check_mon_within*)*, 6

OneStageBasket, *3*, *6*, *8*, *10, 11*, *13*
OneStageBasket *(*OneStageBasket-class*)*, 8
OneStageBasket-class, 8

pow, 9
pow,OneStageBasket-method *(*pow*)*, 9

setupOneStageBasket, 11

toer, 11
toer,OneStageBasket-method *(*toer*)*, 11