Package 'DiscreteFWER'

May 7, 2026

Type Package

Title FWER-Based Multiple Testing Procedures with Adaptation for Discrete Tests

Version 1.0.0

Date 2024-11-26

Description Implementations of several multiple testing procedures that control the family-wise error rate (FWER) designed specifically for discrete tests. Included are discrete adaptations of the Bonferroni, Holm, Hochberg and Šidák procedures as described in the papers Döhler (2010) ``Validation of credit default probabilities using multiple-testing procedures'' <doi:10.21314/JRMV.2010.062> and Zhu & Guo (2019) ``Family-Wise Error Rate Controlling Procedures for Discrete Data'' <doi:10.1080/19466315.2019.1654912>. The main procedures of this package take as input the results of a test procedure from package 'DiscreteTests' or a set of observed p-values and their discrete support under their nulls. A shortcut function to apply discrete procedures directly to data is also provided.

License GPL (>= 2)

Language en-GB

Encoding UTF-8

Depends R (>= 4.0)

Imports Rcpp (>= 1.0.13), checkmate, DiscreteFDR (>= 2.0.0)

LinkingTo Rcpp, RcppArmadillo

Suggests DiscreteDatasets, DiscreteTests (>= 0.2.1)

URL https://github.com/DISOhda/DiscreteFWER

BugReports https://github.com/DISOhda/DiscreteFWER/issues

RoxygenNote 7.3.2

NeedsCompilation yes

Author Sebastian Döhler [aut, ctb] (ORCID: <https://orcid.org/0000-0002-0321-6355>), Florian Junge [aut, ctb, cre] (ORCID: <https://orcid.org/0009-0001-6856-6938>)

Maintainer Florian Junge <diso.fbmn@h-da.de>

Repository CRAN

Date/Publication 2024-11-27 13:00:05 UTC

Contents

`DiscreteFWER-package` *FWER-Based Multiple Testing Procedures with Adaptation for Discrete Tests*

Description

This package implements adaptions for discrete tests of the Bonferroni, Holm, Hochberg and Šidák procedures for control of the family-wise error rate (FWER).

Details

The main function `discrete_FWER()` makes all four procedures available to the user. `DBonferroni()`, `DHolm()`, `DHochberg()` and `DSidak()` are wrapper functions that enable the user to access them directly. Their main parameters are either a `DiscreteTestResults` object from package DiscreteTests or a vector of raw observed p-values and a list whose elements are the discrete supports of the CDFs of the p-values.

The function `direct_discrete_FWER()`is a wrapper for `DiscreteFDR::generate.pvalues()` and `discrete_FWER()`, which applies discrete procedures directly to data.

Author(s)

Maintainer: Florian Junge <diso.fbmn@h-da.de> (ORCID) [contributor]

Authors:

- Sebastian Döhler <sebastian.doehler@h-da.de> (ORCID) [contributor]

References

Döhler, S. (2010). Validation of credit default probabilities using multiple-testing procedures. *Journal of Risk Model Validation*, *4*(4), 59-92. doi:10.21314/JRMV.2010.062

Zhu, Y., & Guo, W. (2019). Family-Wise Error Rate Controlling Procedures for Discrete Data. *Statistics in Biopharmaceutical Research*, *12*(1), 117-128. doi:10.1080/19466315.2019.1654912

See Also

Useful links:

- https://github.com/DISOhda/DiscreteFWER
- Report bugs at https://github.com/DISOhda/DiscreteFWER/issues

---

`DBonferroni` *Discrete Bonferroni Procedure*

Description

`DBonferroni()` is a wrapper function of `discrete_FWER()` for computing the discrete Bonferroni procedure for tests with an arbitrary dependency structure. It simply passes its arguments to `discrete_FWER()` with fixed `independence = FALSE` and `single_step = TRUE`.

Usage

```
DBonferroni(test_results, ...)

## Default S3 method:
DBonferroni(
  test_results,
  pCDFlist,
  alpha = 0.05,
  critical_values = FALSE,
  select_threshold = 1,
  pCDFlist_indices = NULL,
  ...
)

## S3 method for class 'DiscreteTestResults'
DBonferroni(
  test_results,
  alpha = 0.05,
  critical_values = FALSE,
  select_threshold = 1,
  ...
)
```

Arguments

`test_results`	either a numeric vector with p-values or an R6 object of class `DiscreteTestResults` from package DiscreteTests for which a discrete FWER procedure is to be performed.
`...`	further arguments to be passed to or from other methods. They are ignored here.
`pCDFlist`	list of the supports of the CDFs of the p-values; each list item must be a numeric vector, which is sorted in increasing order and whose last element equals 1.
`alpha`	single real number strictly between 0 and 1 indicating the target FWER level.
`critical_values`	single boolean specifying whether critical constants are to be computed.
`select_threshold`	single real number strictly between 0 and 1 indicating the largest raw p-value to be considered, i.e. only p-values below this threshold are considered and the procedures are adjusted in order to take this selection effect into account; if `select_threshold = 1` (the default), all raw p-values are selected.
`pCDFlist_indices`	list of numeric vectors containing the test indices that indicate to which raw p-value(s) each support in `pCDFlist` belongs; if `NULL` (the default) the lengths of `test_results` and `pCDFlist` **must** be equal.

Details

Computing critical constants (`critical_values = TRUE`) requires considerably more execution time, especially if the number of unique supports is large. We recommend that users should only have them calculated when they need them, e.g. for illustrating the rejection set in a plot or other theoretical reasons. Setting (`critical_values = FALSE`) is sufficient for obtaining rejection decisions and adjusted p-values.

Value

A `DiscreteFWER` S3 class object whose elements are:

`Rejected`	rejected raw p-values.
`Indices`	indices of rejected hypotheses.
`Num_rejected`	number of rejections.
`Adjusted`	adjusted p-values.
`Critical_constants`	critical values (only exists if computations where performed with `critical_values = TRUE`).
`Data`	list with input data.
`Data$Method`	character string describing the performed algorithm, e.g. 'Discrete Bonferroni procedure'.
`Data$Raw_pvalues`	observed p-values.
`Data$pCDFlist`	list of the p-value supports.

`Data$FWER_level`
: FWER level `alpha`.

`Data$Independence`
: boolean indicating whether the p-values were considered as independent.

`Data$Single_step`
: boolean indicating whether a single-step or step-down procedure was performed.

`Data$Data_name`
: the respective variable names of the input data.

`Select`
: list with data related to p-value selection; only exists if `select_threshold` < 1.

`Select$Threshold`
: p-value selection threshold (`select_threshold`).

`Select$Effective_Thresholds`
: results of each p-value CDF evaluated at the selection threshold.

`Select$Pvalues`
: selected p-values that are $\leq$ selection threshold.

`Select$Indices`
: indices of p-values $\leq$ selection threshold.

`Select$Scaled`
: scaled selected p-values.

`Select$Number`
: number of selected p-values $\leq$ selection threshold.

References

Döhler, S. (2010). Validation of credit default probabilities using multiple-testing procedures. *Journal of Risk Model Validation*, *4*(4), 59-92. doi:10.21314/JRMV.2010.062

Zhu, Y., & Guo, W. (2019). Family-Wise Error Rate Controlling Procedures for Discrete Data. *Statistics in Biopharmaceutical Research*, *12*(1), 117-128. doi:10.1080/19466315.2019.1654912

See Also

`discrete_FWER()`, `DHolm()`, `DSidak()`, `DHochberg()`

Examples

```
X1 <- c(4, 2, 2, 14, 6, 9, 4, 0, 1)
X2 <- c(0, 0, 1, 3, 2, 1, 2, 2, 2)
N1 <- rep(148, 9)
N2 <- rep(132, 9)
Y1 <- N1 - X1
Y2 <- N2 - X2
df <- data.frame(X1, Y1, X2, Y2)
df

# Computation of p-values and their supports with Fisher's exact test
library(DiscreteTests)  # for Fisher's exact test
test_results <- fisher_test_pv(df)
raw_pvalues <- test_results$get_pvalues()
pCDFlist <- test_results$get_pvalue_supports()

# d-Bonferroni without critical values; using extracted p-values and supports
DBonferroni_fast <- DBonferroni(raw_pvalues, pCDFlist)
summary(DBonferroni_fast)
```

```
# d-Bonferroni with critical values; using test results object
DBonferroni_crit <- DBonferroni(test_results, critical_values = TRUE)
summary(DBonferroni_crit)
```

DHochberg *Discrete Hochberg Procedure*

Description

DHochberg() is a wrapper function of discrete_FWER() for computing the discrete Hochberg step-up procedure for independent or positively correlated discrete tests. It simply passes its arguments to discrete_FWER() with fixed independence = TRUE and single_step = FALSE.

Usage

```
DHochberg(test_results, ...)

## Default S3 method:
DHochberg(
  test_results,
  pCDFlist,
  alpha = 0.05,
  critical_values = FALSE,
  select_threshold = 1,
  pCDFlist_indices = NULL,
  ...
)

## S3 method for class 'DiscreteTestResults'
DHochberg(
  test_results,
  alpha = 0.05,
  critical_values = FALSE,
  select_threshold = 1,
  ...
)
```

Arguments

test_results	either a numeric vector with p-values or an R6 object of class DiscreteTestResults from package DiscreteTests for which a discrete FWER procedure is to be performed.
...	further arguments to be passed to or from other methods. They are ignored here.
pCDFlist	list of the supports of the CDFs of the p-values; each list item must be a numeric vector, which is sorted in increasing order and whose last element equals 1.

`alpha` single real number strictly between 0 and 1 indicating the target FWER level.

`critical_values`
: single boolean specifying whether critical constants are to be computed.

`select_threshold`
: single real number strictly between 0 and 1 indicating the largest raw p-value to be considered, i.e. only p-values below this threshold are considered and the procedures are adjusted in order to take this selection effect into account; if `select_threshold = 1` (the default), all raw p-values are selected.

`pCDFlist_indices`
: list of numeric vectors containing the test indices that indicate to which raw p-value(s) each support in `pCDFlist` belongs; if `NULL` (the default) the lengths of `test_results` and `pCDFlist` **must** be equal.

Details

Computing critical constants (`critical_values = TRUE`) requires considerably more execution time, especially if the number of unique supports is large. We recommend that users should only have them calculated when they need them, e.g. for illustrating the rejection set in a plot or other theoretical reasons. Setting (`critical_values = FALSE`) is sufficient for obtaining rejection decisions and adjusted p-values.

Value

A `DiscreteFWER` S3 class object whose elements are:

`Rejected` rejected raw p-values.

`Indices` indices of rejected hypotheses.

`Num_rejected` number of rejections.

`Adjusted` adjusted p-values.

`Critical_constants`
: critical values (only exists if computations where performed with `critical_values = TRUE`).

`Data` list with input data.

`Data$Method` character string describing the performed algorithm, e.g. 'Discrete Bonferroni procedure'.

`Data$Raw_pvalues`
: observed p-values.

`Data$pCDFlist` list of the p-value supports.

`Data$FWER_level`
: FWER level `alpha`.

`Data$Independence`
: boolean indicating whether the p-values were considered as independent.

`Data$Single_step`
: boolean indicating whether a single-step or step-down procedure was performed.

`Data$Data_name` the respective variable names of the input data.

`Select`	list with data related to p-value selection; only exists if `select_threshold` < 1.
`Select$Threshold`	p-value selection threshold (`select_threshold`).
`Select$Effective_Thresholds`	results of each p-value CDF evaluated at the selection threshold.
`Select$Pvalues`	selected p-values that are $\leq$ selection threshold.
`Select$Indices`	indices of p-values $\leq$ selection threshold.
`Select$Scaled`	scaled selected p-values.
`Select$Number`	number of selected p-values $\leq$ selection threshold.

References

Zhu, Y., & Guo, W. (2019). Family-Wise Error Rate Controlling Procedures for Discrete Data. *Statistics in Biopharmaceutical Research*, *12*(1), 117-128. doi:10.1080/19466315.2019.1654912

See Also

`discrete_FWER()`, `DSidak()`, `DBonferroni()`, `DHolm`

Examples

```
X1 <- c(4, 2, 2, 14, 6, 9, 4, 0, 1)
X2 <- c(0, 0, 1, 3, 2, 1, 2, 2, 2)
N1 <- rep(148, 9)
N2 <- rep(132, 9)
Y1 <- N1 - X1
Y2 <- N2 - X2
df <- data.frame(X1, Y1, X2, Y2)
df

# Computation of p-values and their supports with Fisher's exact test
library(DiscreteTests)  # for Fisher's exact test
test_results <- fisher_test_pv(df)
raw_pvalues <- test_results$get_pvalues()
pCDFlist <- test_results$get_pvalue_supports()

# d-Hochberg without critical values; using test results object
DHoch_fast <- DHochberg(test_results)
summary(DHoch_fast)

# d-Hochberg with critical values; using extracted p-values and supports
DHoch_crit <- DHochberg(raw_pvalues, pCDFlist, critical_values = TRUE)
summary(DHoch_crit)
```

DHolm *Discrete Holm Procedure*

Description

DHolm() is a wrapper function of discrete_FWER() for computing the discrete Holm step-down procedure for tests with an arbitrary dependency structure. It simply passes its arguments to discrete_FWER() with fixed independence = FALSE and single_step = FALSE.

Usage

```
DHolm(test_results, ...)

## Default S3 method:
DHolm(
  test_results,
  pCDFlist,
  alpha = 0.05,
  critical_values = FALSE,
  select_threshold = 1,
  pCDFlist_indices = NULL,
  ...
)

## S3 method for class 'DiscreteTestResults'
DHolm(
  test_results,
  alpha = 0.05,
  critical_values = FALSE,
  select_threshold = 1,
  ...
)
```

Arguments

test_results	either a numeric vector with p-values or an R6 object of class DiscreteTestResults from package DiscreteTests for which a discrete FWER procedure is to be performed.
...	further arguments to be passed to or from other methods. They are ignored here.
pCDFlist	list of the supports of the CDFs of the p-values; each list item must be a numeric vector, which is sorted in increasing order and whose last element equals 1.
alpha	single real number strictly between 0 and 1 indicating the target FWER level.
critical_values	single boolean specifying whether critical constants are to be computed.

`select_threshold`
: single real number strictly between 0 and 1 indicating the largest raw p-value to be considered, i.e. only p-values below this threshold are considered and the procedures are adjusted in order to take this selection effect into account; if `select_threshold = 1` (the default), all raw p-values are selected.

`pCDFlist_indices`
: list of numeric vectors containing the test indices that indicate to which raw p-value(s) each support in `pCDFlist` belongs; if `NULL` (the default) the lengths of `test_results` and `pCDFlist` **must** be equal.

Details

Computing critical constants (`critical_values = TRUE`) requires considerably more execution time, especially if the number of unique supports is large. We recommend that users should only have them calculated when they need them, e.g. for illustrating the rejection set in a plot or other theoretical reasons. Setting (`critical_values = FALSE`) is sufficient for obtaining rejection decisions and adjusted p-values.

Value

A `DiscreteFWER` S3 class object whose elements are:

`Rejected`
: rejected raw p-values.

`Indices`
: indices of rejected hypotheses.

`Num_rejected`
: number of rejections.

`Adjusted`
: adjusted p-values.

`Critical_constants`
: critical values (only exists if computations where performed with `critical_values = TRUE`).

`Data`
: list with input data.

`Data$Method`
: character string describing the performed algorithm, e.g. 'Discrete Bonferroni procedure'.

`Data$Raw_pvalues`
: observed p-values.

`Data$pCDFlist`
: list of the p-value supports.

`Data$FWER_level`
: FWER level `alpha`.

`Data$Independence`
: boolean indicating whether the p-values were considered as independent.

`Data$Single_step`
: boolean indicating whether a single-step or step-down procedure was performed.

`Data$Data_name`
: the respective variable names of the input data.

`Select`
: list with data related to p-value selection; only exists if `select_threshold < 1`.

`Select$Threshold`
: p-value selection threshold (`select_threshold`).

`Select$Effective_Thresholds`
: results of each p-value CDF evaluated at the selection threshold.

`Select$Pvalues`
: selected p-values that are $\leq$ selection threshold.

`Select$Indices`
: indices of p-values $\leq$ selection threshold.

`Select$Scaled`
: scaled selected p-values.

`Select$Number`
: number of selected p-values $\leq$ selection threshold.

References

Döhler, S. (2010). Validation of credit default probabilities using multiple-testing procedures. *Journal of Risk Model Validation*, *4*(4), 59-92. doi:10.21314/JRMV.2010.062

Zhu, Y., & Guo, W. (2019). Family-Wise Error Rate Controlling Procedures for Discrete Data. *Statistics in Biopharmaceutical Research*, *12*(1), 117-128. doi:10.1080/19466315.2019.1654912

See Also

`discrete_FWER()`, `DBonferroni()`, `DSidak()`, `DHochberg()`

Examples

```
X1 <- c(4, 2, 2, 14, 6, 9, 4, 0, 1)
X2 <- c(0, 0, 1, 3, 2, 1, 2, 2, 2)
N1 <- rep(148, 9)
N2 <- rep(132, 9)
Y1 <- N1 - X1
Y2 <- N2 - X2
df <- data.frame(X1, Y1, X2, Y2)
df

# Computation of p-values and their supports with Fisher's exact test
library(DiscreteTests)  # for Fisher's exact test
test_results <- fisher_test_pv(df)
raw_pvalues <- test_results$get_pvalues()
pCDFlist <- test_results$get_pvalue_supports()

# d-Holm without critical values; using extracted p-values and supports
DHolm_fast <- DHolm(raw_pvalues, pCDFlist)
summary(DHolm_fast)

# d-Holm with critical values; using test results object
DHolm_crit <- DHolm(test_results, critical_values = TRUE)
summary(DHolm_crit)
```

`direct_discrete_FWER` *Direct Application of Multiple Testing Procedures to Dataset*

Description

Apply one of the various FWER adaptation procedures, with or without computing the critical constants, to a data set of 2x2 contingency tables using statistical test functions from package `DiscreteTests`. If necessary, functions for pre-processing can be passed as well.

Usage

```
direct_discrete_FWER(
  dat,
  test_fun,
  test_args = NULL,
  alpha = 0.05,
  independence = FALSE,
  single_step = TRUE,
  critical_values = FALSE,
  select_threshold = 1,
  preprocess_fun = NULL,
  preprocess_args = NULL
)
```

Arguments

`dat`	input data; must be suitable for the first parameter of the provided `preprocess_fun` function or, if `preprocess_fun` is `NULL`, for the first parameter of the `test_fun` function.
`test_fun`	function **from package** `DiscreteTests`, i.e. one whose name ends with `*_test_pv` and which performs hypothesis tests and provides an object with p-values and their support sets; can be specified by a single character string (which is automatically checked for being a suitable function **from that package** and may be abbreviated) or a single function object.
`test_args`	optional named list with arguments for `test_fun`; the names of the list fields must match the test function's parameter names. The first parameter of the test function (i.e. the data) MUST NOT be included!
`alpha`	single real number strictly between 0 and 1 indicating the target FWER level.
`independence`	single boolean specifying whether the p-values are statistically independent or not.
`single_step`	single boolean specifying whether to perform a single-step (`TRUE`) or step-down (`FALSE`; the default) procedure.
`critical_values`	single boolean specifying whether critical constants are to be computed.

`select_threshold`
: single real number strictly between 0 and 1 indicating the largest raw p-value to be considered, i.e. only p-values below this threshold are considered and the procedures are adjusted in order to take this selection effect into account; if `select_threshold = 1` (the default), all raw p-values are selected.

`preprocess_fun`
: optional function for pre-processing the input `data`; its result must be suitable for the first parameter of the `test_fun` function.

`preprocess_args`
: optional named list with arguments for `preprocess_fun`; the names of the list fields must match the pre-processing function's parameter names. The first parameter of the test function (i.e. the data) MUST NOT be included!

Value

A `DiscreteFWER` S3 class object whose elements are:

`Rejected`
: rejected raw p-values.

`Indices`
: indices of rejected hypotheses.

`Num_rejected`
: number of rejections.

`Adjusted`
: adjusted p-values.

`Critical_constants`
: critical values (only exists if computations where performed with `critical_values = TRUE`).

`Data`
: list with input data.

`Data$Method`
: character string describing the performed algorithm, e.g. 'Discrete Bonferroni procedure'.

`Data$Raw_pvalues`
: observed p-values.

`Data$pCDFlist`
: list of the p-value supports.

`Data$FWER_level`
: FWER level `alpha`.

`Data$Independence`
: boolean indicating whether the p-values were considered as independent.

`Data$Single_step`
: boolean indicating whether a single-step or step-down procedure was performed.

`Data$Data_name`
: the respective variable names of the input data.

`Select`
: list with data related to p-value selection; only exists if `select_threshold < 1`.

`Select$Threshold`
: p-value selection threshold (`select_threshold`).

`Select$Effective_Thresholds`
: results of each p-value CDF evaluated at the selection threshold.

`Select$Pvalues`
: selected p-values that are $\leq$ selection threshold.

`Select$Indices`
: indices of p-values $\leq$ selection threshold.

`Select$Scaled`
: scaled selected p-values.

`Select$Number`
: number of selected p-values $\leq$ selection threshold.

Examples

```
X1 <- c(4, 2, 2, 14, 6, 9, 4, 0, 1)
X2 <- c(0, 0, 1, 3, 2, 1, 2, 2, 2)
N1 <- rep(148, 9)
N2 <- rep(132, 9)
Y1 <- N1 - X1
Y2 <- N2 - X2
df <- data.frame(X1, Y1, X2, Y2)
df

# Computation of p-values and their supports with Fisher's exact test
library(DiscreteTests)  # for Fisher's exact test
test_results <- fisher_test_pv(df)
raw_pvalues <- test_results$get_pvalues()
pCDFlist <- test_results$get_pvalue_supports()

DBonf <- direct_discrete_FWER(df, "fisher")
summary(DBonf)

DHolm <- direct_discrete_FWER(df, "fisher_test_pv", single_step = FALSE)
summary(DHolm)

DBonf_bin <- direct_discrete_FWER(X1 + X2, "binom_test_pv",
                                  list(n = N1 + N2, p = 0.05))
summary(DBonf_bin)

DHolm_bin <- direct_discrete_FWER(X1 + X2, "binom",
                                  list(n = N1 + N2, p = 0.05),
                                  single_step = TRUE)
summary(DHolm_bin)
```

`discrete_FWER` *Discrete Family-wise Error Rate (FWER) Adaptation Procedures*

Description

Apply a discrete FWER adaptation procedure, with or without computing the critical values, to a set of p-values and their discrete support.

Usage

```
discrete_FWER(test_results, ...)

## Default S3 method:
discrete_FWER(
  test_results,
  pCDFlist,
  alpha = 0.05,
```

```
  independence = FALSE,
  single_step = FALSE,
  critical_values = FALSE,
  select_threshold = 1,
  pCDFlist_indices = NULL,
  ...
)

## S3 method for class 'DiscreteTestResults'
discrete_FWER(
  test_results,
  alpha = 0.05,
  independence = FALSE,
  single_step = FALSE,
  critical_values = FALSE,
  select_threshold = 1,
  ...
)
```

Arguments

`test_results`	either a numeric vector with p-values or an R6 object of class `DiscreteTestResults` from package DiscreteTests for which a discrete FWER procedure is to be performed.
`...`	further arguments to be passed to or from other methods. They are ignored here.
`pCDFlist`	list of the supports of the CDFs of the p-values; each list item must be a numeric vector, which is sorted in increasing order and whose last element equals 1.
`alpha`	single real number strictly between 0 and 1 indicating the target FWER level.
`independence`	single boolean specifying whether the p-values are statistically independent or not.
`single_step`	single boolean specifying whether to perform a single-step (`TRUE`) or step-down (`FALSE`; the default) procedure.
`critical_values`	single boolean specifying whether critical constants are to be computed.
`select_threshold`	single real number strictly between 0 and 1 indicating the largest raw p-value to be considered, i.e. only p-values below this threshold are considered and the procedures are adjusted in order to take this selection effect into account; if `select_threshold = 1` (the default), all raw p-values are selected.
`pCDFlist_indices`	list of numeric vectors containing the test indices that indicate to which raw p-value(s) each support in `pCDFlist` belongs; if `NULL` (the default) the lengths of `test_results` and `pCDFlist` **must** be equal.

Details

Computing critical constants (`critical_values = TRUE`) requires considerably more execution time, especially if the number of unique supports is large. We recommend that users should only have

them calculated when they need them, e.g. for illustrating the rejection set in a plot or other theoretical reasons. Setting (`critical_values = FALSE`) is sufficient for obtaining rejection decisions and adjusted p-values.

Depending on the choices of `independence` and `single_step`, one of the following procedures, is applied:

	single-step	stepwise
independent	Šidák	Hochberg (step-up)
not independent	Bonferroni	Holm (step-down)

Each procedure is available by its own shortcut function:

	single-step	stepwise
independent	`DSidak()`	`DHochberg()`
not independent	`DBonferroni()`	`DHolm()`

Value

A `DiscreteFWER` S3 class object whose elements are:

`Rejected` rejected raw p-values.

`Indices` indices of rejected hypotheses.

`Num_rejected` number of rejections.

`Adjusted` adjusted p-values.

`Critical_constants` critical values (only exists if computations where performed with `critical_values = TRUE`).

`Data` list with input data.

`Data$Method` character string describing the performed algorithm, e.g. 'Discrete Bonferroni procedure'.

`Data$Raw_pvalues` observed p-values.

`Data$pCDFlist` list of the p-value supports.

`Data$FWER_level` FWER level `alpha`.

`Data$Independence` boolean indicating whether the p-values were considered as independent.

`Data$Single_step` boolean indicating whether a single-step or step-down procedure was performed.

`Data$Data_name` the respective variable names of the input data.

`Select` list with data related to p-value selection; only exists if `select_threshold < 1`.

`Select$Threshold` p-value selection threshold (`select_threshold`).

`Select$Effective_Thresholds`
: results of each p-value CDF evaluated at the selection threshold.

`Select$Pvalues`
: selected p-values that are $\leq$ selection threshold.

`Select$Indices`
: indices of p-values $\leq$ selection threshold.

`Select$Scaled`
: scaled selected p-values.

`Select$Number`
: number of selected p-values $\leq$ selection threshold.

References

Döhler, S. (2010). Validation of credit default probabilities using multiple-testing procedures. *Journal of Risk Model Validation*, *4*(4), 59-92. doi:10.21314/JRMV.2010.062

Zhu, Y., & Guo, W. (2019). Family-Wise Error Rate Controlling Procedures for Discrete Data. *Statistics in Biopharmaceutical Research*, *12*(1), 117-128. doi:10.1080/19466315.2019.1654912

See Also

DiscreteFWER, DBonferroni(), DHolm(), DSidak(), DHochberg()

Examples

```
X1 <- c(4, 2, 2, 14, 6, 9, 4, 0, 1)
X2 <- c(0, 0, 1, 3, 2, 1, 2, 2, 2)
N1 <- rep(148, 9)
N2 <- rep(132, 9)
Y1 <- N1 - X1
Y2 <- N2 - X2
df <- data.frame(X1, Y1, X2, Y2)
df

# Computation of p-values and their supports with Fisher's exact test
library(DiscreteTests)  # for Fisher's exact test
test_results <- fisher_test_pv(df)
raw_pvalues <- test_results$get_pvalues()
pCDFlist <- test_results$get_pvalue_supports()

# d-Holm without critical values; using test results object
DFWER_dep_sd_fast <- discrete_FWER(test_results)
summary(DFWER_dep_sd_fast)

# d-Holm with critical values; using extracted p-values and supports
DFWER_dep_sd_crit <- discrete_FWER(raw_pvalues, pCDFlist,
                                   critical_values = TRUE)
summary(DFWER_dep_sd_crit)

# d-Bonferroni without critical values; using test results object
DFWER_dep_fast <- discrete_FWER(test_results, single_step = TRUE)
summary(DFWER_dep_fast)

# d-Bonferroni with critical values; using extracted p-values and supports
DFWER_dep_crit <- discrete_FWER(raw_pvalues, pCDFlist, single_step = TRUE,
```

```
                                critical_values = TRUE)
summary(DFWER_dep_crit)

# d-Hochberg without critical values; using test results object
DFWER_ind_su_fast <- discrete_FWER(test_results, independence = TRUE)
summary(DFWER_ind_su_fast)

# d-Hochberg with critical values; using extracted p-values and supports
DFWER_ind_su_crit <- discrete_FWER(raw_pvalues, pCDFlist,
                                   independence = TRUE,
                                   critical_values = TRUE)
summary(DFWER_ind_su_crit)

# d-Šidák without critical values; using extracted p-values and supports
DFWER_ind_fast <- discrete_FWER(raw_pvalues, pCDFlist,
                                independence = TRUE,
                                single_step = TRUE)
summary(DFWER_ind_fast)

# d-Šidák with critical values; using test results object
DFWER_ind_crit <- discrete_FWER(test_results, independence = TRUE,
                                single_step = TRUE,
                                critical_values = TRUE)
summary(DFWER_ind_crit)
```

DSidak *Discrete Šidák Procedure for Independent Tests*

Description

DSidak() is a wrapper function of discrete_FWER() for computing the discrete Šidák procedure for independent discrete tests. It simply passes its arguments to discrete_FWER() with fixed independence = TRUE and single_step = TRUE.

Usage

```
DSidak(test_results, ...)

## Default S3 method:
DSidak(
  test_results,
  pCDFlist,
  alpha = 0.05,
  critical_values = FALSE,
  select_threshold = 1,
  pCDFlist_indices = NULL,
  ...
)
```

```
## S3 method for class 'DiscreteTestResults'
DSidak(
  test_results,
  alpha = 0.05,
  critical_values = FALSE,
  select_threshold = 1,
  ...
)
```

Arguments

`test_results`	either a numeric vector with p-values or an R6 object of class `DiscreteTestResults` from package DiscreteTests for which a discrete FWER procedure is to be performed.
`...`	further arguments to be passed to or from other methods. They are ignored here.
`pCDFlist`	list of the supports of the CDFs of the p-values; each list item must be a numeric vector, which is sorted in increasing order and whose last element equals 1.
`alpha`	single real number strictly between 0 and 1 indicating the target FWER level.
`critical_values`	single boolean specifying whether critical constants are to be computed.
`select_threshold`	single real number strictly between 0 and 1 indicating the largest raw p-value to be considered, i.e. only p-values below this threshold are considered and the procedures are adjusted in order to take this selection effect into account; if `select_threshold = 1` (the default), all raw p-values are selected.
`pCDFlist_indices`	list of numeric vectors containing the test indices that indicate to which raw p-value(s) each support in `pCDFlist` belongs; if `NULL` (the default) the lengths of `test_results` and `pCDFlist` **must** be equal.

Details

Computing critical constants (`critical_values = TRUE`) requires considerably more execution time, especially if the number of unique supports is large. We recommend that users should only have them calculated when they need them, e.g. for illustrating the rejection set in a plot or other theoretical reasons. Setting (`critical_values = FALSE`) is sufficient for obtaining rejection decisions and adjusted p-values.

Value

A `DiscreteFWER` S3 class object whose elements are:

`Rejected`	rejected raw p-values.
`Indices`	indices of rejected hypotheses.
`Num_rejected`	number of rejections.
`Adjusted`	adjusted p-values.

`Critical_constants`
: critical values (only exists if computations where performed with `critical_values = TRUE`).

`Data`
: list with input data.

`Data$Method`
: character string describing the performed algorithm, e.g. 'Discrete Bonferroni procedure'.

`Data$Raw_pvalues`
: observed p-values.

`Data$pCDFlist`
: list of the p-value supports.

`Data$FWER_level`
: FWER level `alpha`.

`Data$Independence`
: boolean indicating whether the p-values were considered as independent.

`Data$Single_step`
: boolean indicating whether a single-step or step-down procedure was performed.

`Data$Data_name`
: the respective variable names of the input data.

`Select`
: list with data related to p-value selection; only exists if `select_threshold < 1`.

`Select$Threshold`
: p-value selection threshold (`select_threshold`).

`Select$Effective_Thresholds`
: results of each p-value CDF evaluated at the selection threshold.

`Select$Pvalues`
: selected p-values that are $\leq$ selection threshold.

`Select$Indices`
: indices of p-values $\leq$ selection threshold.

`Select$Scaled`
: scaled selected p-values.

`Select$Number`
: number of selected p-values $\leq$ selection threshold.

References

Döhler, S. (2010). Validation of credit default probabilities using multiple-testing procedures. *Journal of Risk Model Validation*, *4*(4), 59-92. doi:10.21314/JRMV.2010.062

See Also

`discrete_FWER()`, `DHochberg()`, `DBonferroni()`, `DHolm()`

Examples

```
X1 <- c(4, 2, 2, 14, 6, 9, 4, 0, 1)
X2 <- c(0, 0, 1, 3, 2, 1, 2, 2, 2)
N1 <- rep(148, 9)
N2 <- rep(132, 9)
Y1 <- N1 - X1
Y2 <- N2 - X2
df <- data.frame(X1, Y1, X2, Y2)
df
```

```
# Computation of p-values and their supports with Fisher's exact test
library(DiscreteTests)  # for Fisher's exact test
test_results <- fisher_test_pv(df)
raw_pvalues <- test_results$get_pvalues()
pCDFlist <- test_results$get_pvalue_supports()

# d-Šidák without critical values; using extracted p-values and supports
DSidak_fast <- DSidak(raw_pvalues, pCDFlist)
summary(DSidak_fast)

# d-Šidák with critical values; using test results object
DSidak_crit <- DSidak(test_results, critical_values = TRUE)
summary(DSidak_crit)
```

`hist.DiscreteFWER` *Histogram of Raw P-Values*

Description

Computes a histogram of the raw p-values of a `DiscreteFWER` object.

Usage

```
## S3 method for class 'DiscreteFWER'
hist(x, breaks = "FD", mode = c("raw", "selected"), ...)
```

Arguments

`x`	an object of class `DiscreteFWER`.
`breaks`	as in `graphics::hist()`; here, the Friedman-Diaconis algorithm (`"FD"`) is used as default.
`mode`	single character string specifying for which p-values the histogram is to be generated; must either be `"raw"` or `"selected"`.
`...`	further arguments to `graphics::hist()` or `graphics::plot.histogram()`, respectively.

Details

If `x` does not contain results of a selection approach, a warning is issued and a histogram of the raw p-values is drawn.

Value

An object of class `histogram`.

Examples

```
X1 <- c(4, 2, 2, 14, 6, 9, 4, 0, 1)
X2 <- c(0, 0, 1, 3, 2, 1, 2, 2, 2)
N1 <- rep(148, 9)
N2 <- rep(132, 9)
Y1 <- N1 - X1
Y2 <- N2 - X2
df <- data.frame(X1, Y1, X2, Y2)
df

# Computation of p-values and their supports with Fisher's exact test
library(DiscreteTests)  # for Fisher's exact test
test_results <- fisher_test_pv(df)
raw_pvalues <- test_results$get_pvalues()
pCDFlist <- test_results$get_pvalue_supports()

# d-Holm with critical values; using test results object
DHolm_crit <- DHolm(test_results, critical_values = TRUE)
hist(DHolm_crit)
```

---

plot.DiscreteFWER *Plot Method for* DiscreteFWER *objects*

Description

Plots raw p-values of a DiscreteFWER object and highlights rejected and accepted p-values. If calculated, the critical values are plotted, too.

Usage

```
## S3 method for class 'DiscreteFWER'
plot(
  x,
  col = c(2, 4, 1),
  pch = c(20, 20, 17),
  lwd = rep(par()$lwd, 3),
  cex = rep(par()$cex, 3),
  type_crit = "b",
  legend = NULL,
  ...
)
```

Arguments

x	object of class DiscreteFWER.
col	numeric or character vector of length 3 indicating the colours of the 1. rejected p-values

2. accepted p-values
3. critical values (if present).

`pch` numeric or character vector of length 3 indicating the point characters of the

1. rejected p-values
2. accepted p-values
3. critical values (if present and `type_crit` is a plot type like `'p'`, `'b'` etc.).

`lwd` numeric vector of length 3 indicating the thickness of the points and lines; defaults to current `par()$lwd` setting.

`cex` numeric vector of length 3 indicating the size of point characters or lines of the

1. rejected p-values
2. accepted p-values
3. critical values (if present).

defaults to current `par()$cex` setting.

`type_crit` 1-character string giving the type of plot desired for the critical values (e.g.: `'p'`, `'l'` etc; see `plot()`).

`legend` if `NULL`, no legend is plotted; otherwise expecting a character string like `"topleft"` etc. or a numeric vector of two elements indicating (x, y) coordinates.

`...` further arguments to `plot.default()`.

Value

A plot is created, but no value is returned.

Examples

```
X1 <- c(4, 2, 2, 14, 6, 9, 4, 0, 1)
X2 <- c(0, 0, 1, 3, 2, 1, 2, 2, 2)
N1 <- rep(148, 9)
N2 <- rep(132, 9)
Y1 <- N1 - X1
Y2 <- N2 - X2
df <- data.frame(X1, Y1, X2, Y2)
df

# Computation of p-values and their supports with Fisher's exact test
library(DiscreteTests)  # for Fisher's exact test
test_results <- fisher_test_pv(df)
raw_pvalues <- test_results$get_pvalues()
pCDFlist <- test_results$get_pvalue_supports()

DBonf_fast <- DBonferroni(raw_pvalues, pCDFlist)
DBonf_crit <- DBonferroni(test_results, critical_values = TRUE)
DHolm_fast <- DHolm(test_results)
DHolm_crit <- DHolm(raw_pvalues, pCDFlist, critical_values = TRUE)

plot(DBonf_fast)
plot(DBonf_crit, xlim = c(1, 5), ylim = c(0, 0.4))
```

```
plot(DHolm_fast, col = c(2, 4), pch = c(2, 3), lwd = c(2, 2),
     legend = "topleft", xlim = c(1, 5), ylim = c(0, 0.4))
plot(DHolm_crit, col = c(2, 4, 1), pch = c(1, 1, 4), lwd = c(1, 1, 2),
     type_crit = 'o', legend = c(1, 0.4), lty = 1, xlim = c(1, 5),
     ylim = c(0, 0.4))
```

`print.DiscreteFWER` *Printing discrete FWER results*

Description

Prints the results of discrete FWER analysis, stored in a `DiscreteFWER` S3 class object.

Usage

```
## S3 method for class 'DiscreteFWER'
print(x, ...)
```

Arguments

`x`	object of class `DiscreteFWER`.
`...`	further arguments to be passed to or from other methods. They are ignored in this function.

Value

The respective input object is invisibly returned via `invisible(x)`.

Examples

```
X1 <- c(4, 2, 2, 14, 6, 9, 4, 0, 1)
X2 <- c(0, 0, 1, 3, 2, 1, 2, 2, 2)
N1 <- rep(148, 9)
N2 <- rep(132, 9)
Y1 <- N1 - X1
Y2 <- N2 - X2
df <- data.frame(X1, Y1, X2, Y2)
df

# Computation of p-values and their supports with Fisher's exact test
library(DiscreteTests)  # for Fisher's exact test
test_results <- fisher_test_pv(df)
raw_pvalues <- test_results$get_pvalues()
pCDFlist <- test_results$get_pvalue_supports()

# d-Holm with critical values; using test results object
DHolm_crit <- DHolm(test_results, critical.values = TRUE)
# print results
print(DHolm_crit)
```

summary.DiscreteFWER *Summarizing Discrete FWER Results*

Description

summary method for class DiscreteFWER.

Usage

```
## S3 method for class 'DiscreteFWER'
summary(object, ...)

## S3 method for class 'summary.DiscreteFWER'
print(x, max = NULL, ...)
```

Arguments

object	an object of class DiscreteFWER.
...	further arguments passed to or from other methods.
x	an object of class summary.DiscreteFWER.
max	numeric or NULL, specifying the maximal number of *rows* of the p-value table to be printed. By default, when NULL, getOption("max.print") is used.

Details

summary.DiscreteFWER objects contain all data of an DiscreteFWER object, but also include an additional table which includes the raw p-values, their indices, the respective critical values (if present), the adjusted p-values (if present) and a logical column to indicate rejection. The table is sorted in ascending order by the raw p-values.

print.summary.DiscreteFWER simply prints the same output as print.DiscreteFWER, but also prints the p-value table.

Value

summary.DiscreteFWER computes and returns a list that includes all the data of an input DiscreteFWER object, plus

Table	data.frame, sorted by the raw p-values, that contains the indices, the raw p-values themselves, their respective critical values (if present), their adjusted p-values (if present) and a logical column to indicate rejection.

Examples

```
X1 <- c(4, 2, 2, 14, 6, 9, 4, 0, 1)
X2 <- c(0, 0, 1, 3, 2, 1, 2, 2, 2)
N1 <- rep(148, 9)
N2 <- rep(132, 9)
Y1 <- N1 - X1
Y2 <- N2 - X2
df <- data.frame(X1, Y1, X2, Y2)
df

# Computation of p-values and their supports with Fisher's exact test
library(DiscreteTests)  # for Fisher's exact test
test_results <- fisher_test_pv(df)
raw_pvalues <- test_results$get_pvalues()
pCDFlist <- test_results$get_pvalue_supports()

# d-Holm procedure without critical values; using test results object
DFWER_dep_sd_fast <- discrete_FWER(test_results)
summary(DFWER_dep_sd_fast)

# d-Bonferroni procedure with critical values; using test results object
DFWER_dep_crit <- discrete_FWER(test_results, single_step = TRUE,
critical_values = TRUE)
summary(DFWER_dep_crit)
```

Index