Package 'FDX'

October 19, 2026

Type Package

Title False Discovery Exceedance Controlling Multiple Testing Procedures

Version 2.0.4

Date 2026-10-19

Description Multiple testing procedures for heterogeneous and discrete tests as described in Döhler and Roquain (2020) <doi:10.1214/20-EJS1771>. The main algorithms of the paper are available as continuous, discrete and weighted versions. They 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 obtain such p-values and supports is also provided, along with wrappers allowing to apply discrete procedures directly to data.

License GPL-3

Language en-US

Encoding UTF-8

Depends R (>= 3.00)

Imports Rcpp (>= 1.0.12), PoissonBinomial (>= 1.2.0), pracma, DiscreteFDR (>= 2.0.0), checkmate, lifecycle, methods

Suggests DiscreteTests (>= 0.2.1)

LinkingTo Rcpp, RcppArmadillo, PoissonBinomial

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

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

Config/roxygen2/version 8.1.1

NeedsCompilation yes

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

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

Repository CRAN

Date/Publication 2026-10-19 13:00:02 UTC

Contents

FDX-package . . . 2
continuous.GR . . . 4
continuous.LR . . . 6
DGR . . . 9
direct.discrete . . . 12
discrete.GR . . . 15
discrete.LR . . . 18
DLR . . . 22
DPB . . . 25
DPB.DiscreteTestResults . . . 28
fast.Discrete . . . 31
hist.FDX . . . 34
NDGR . . . 36
NDLR . . . 39
NDPB . . . 42
plot.FDX . . . 45
print.FDX . . . 47
rejection.path . . . 48
summary.FDX . . . 49
weighted.GR . . . 51
weighted.LR . . . 54
weighted.PB . . . 57

Index **60**

`FDX-package` *False Discovery Exceedance (FDX) Control for Heterogeneous and Discrete Tests*

Description

This package implements the [HLR], [HGR] and [HPB] procedures for both heterogeneous and discrete tests (see Reference).

Details

The functions are reorganized from the reference paper in the following way. `discrete.LR()` (for Discrete Lehmann-Romano) implements [DLR], `discrete.GR()` (for Discrete Guo-Romano) implements [DGR] and `discrete.PB()` (for Discrete Poisson-Binomial) implements [DPB]. `DLR()` and `NDLR()` are wrappers for `discrete.LR()` to access [DLR] and its non-adaptive version directly. Likewise, `DGR()`, `NDGR()`, `DPB()` and `NDPB()` are wrappers for `discrete.GR()` and `discrete.PB()`,

respectively. Their main parameters are a vector of raw observed p-values and a list of the same length, whose elements are the discrete supports of the CDFs of the p-values.

In the same fashion, `weighted.LR()` (for Weighted Lehmann-Romano), `weighted.GR()` (for Weighted Guo-Romano) and `weighted.PB()` (for Weighted Poisson-Binomial) implement [wLR], [wGR] and [wGR], respectively. They also possess wrapper functions, namely `wLR.AM()`, `wGR.AM()` and `wPB.AM()` for arithmetic weighting, and `wLR.GM()`, `wPB.GM()` and `wPB.GM()` for geometric weighting.

The functions `fast.Discrete.LR()`, `fast.Discrete.GR()` and `fast.Discrete.PB()` are wrappers for `DiscreteFDR::fisher.pvalues.support()` and `discrete.LR()`, `discrete.GR()` and `discrete.PB()`, respectively, which allow to apply discrete procedures directly to a data set of contingency tables.

Author(s)

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

Authors:

- Florian Junge <diso.fbmn@h-da.de> (ORCID)
- Sebastian Döhler <sebastian.doehler@h-da.de> (ORCID)

Other contributors:

- Etienne Roquain [contributor]

References

Döhler, S. & Roquain, E. (2020). Controlling False Discovery Exceedance for Heterogeneous Tests. *Electronic Journal of Statistics*, *14*(2), pp. 4244-4272. doi:10.1214/20EJS1771

Lehmann, E. L. & Romano, J. P. (2005). Generalizations of the familywise error rate. *The Annals of Statistics*, *33*(3), pp. 1138-1154. doi:10.1214/009053605000000084

Guo, W. & Romano, J. P. (2007). A generalized Sidak-Holm procedure and control of generalized error rates under independence. *Statistical Applications in Genetics and Molecular Biology*, *6*(1), Art. 3, 35 pp. (electronic). doi:10.2202/15446115.1247

See Also

Useful links:

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

`continuous.GR` *Continuous Guo-Romano procedure*

Description

Apply the usual continuous [GR] procedure, with or without computing the critical values, to a set of p-values. A non-adaptive version is available as well.

Usage

```
continuous.GR(
  test.results,
  alpha = 0.05,
  zeta = 0.5,
  adaptive = TRUE,
  critical.values = FALSE,
  select.threshold = 1
)

GR(
  test.results,
  alpha = 0.05,
  zeta = 0.5,
  critical.values = FALSE,
  select.threshold = 1
)

NGR(
  test.results,
  alpha = 0.05,
  zeta = 0.5,
  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 FDR procedure is to be performed.
`alpha`	single real number strictly between 0 and 1 specifying the target FDP.
`zeta`	single real number strictly between 0 and 1 specifying the target probability of not exceeding the desired FDP. If `zeta = NULL` (the default), then `zeta` is chosen equal to `alpha`.
`adaptive`	single boolean indicating whether to conduct an adaptive procedure or not.

`critical.values`
: single boolean indicating 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 `threshold = 1` (the default), all raw p-values are selected.

Details

`GR` and `NGR` are wrapper functions for `continuous.GR`. The first one simply passes all its arguments to `continuous.GR` with `adaptive = TRUE` and `NGR` does the same with `adaptive = FALSE`.

Value

A `FDX` S3 class object whose elements are:

`Rejected` rejected raw p-values.

`Indices` indices of rejected p-values.

`Num.rejected` number of rejections.

`Adjusted` adjusted p-values.

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

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

`Select$Threshold`
: p-value selection 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.

`Data` list with input data.

`Data$Method` character string describing the used algorithm, e.g. 'Discrete Lehmann-Romano procedure (step-up)'.

`Data$Raw.pvalues`
: all observed raw p-values.

`Data$FDP.threshold`
: FDP threshold `alpha`.

`Data$Exceedance.probability`
: probability `zeta` of FDP exceeding `alpha`; thus, FDP is being controlled at level `alpha` with confidence 1 - `zeta`.

`Data$Adaptive` boolean indicating whether an adaptive procedure was conducted or not.

`Data$Data.name` the respective variable name(s) of the input data.

References

Guo, W. & Romano, J. P. (2007). A generalized Sidak-Holm procedure and control of generalized error rates under independence. *Statistical Applications in Genetics and Molecular Biology*, *6*(1), Art. 3, 35 pp. (electronic). doi:10.2202/15446115.1247

See Also

kernel, FDX-package, continuous.LR(), discrete.LR(), discrete.GR(), discrete.PB(), weighted.LR(), weighted.GR(), weighted.PB()

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

# Construction of the 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()

# GR without critical values; using extracted p-values
GR.fast <- GR(raw.pvalues)
summary(GR.fast)

# LR with critical values; using test results object
GR.crit <- GR(test.results, critical.values = TRUE)
summary(GR.crit)

# Non-adaptive GR without critical values; using test results object
NGR.fast <- NGR(test.results)
summary(NGR.fast)

# Non-adaptive GR with critical values; using extracted p-values
NGR.crit <- NGR(raw.pvalues, critical.values = TRUE)
summary(NGR.crit)
```

---

continuous.LR *Continuous Lehmann-Romano procedure*

Description

Apply the usual (continuous) [LR] procedure, with or without computing the critical values, to a set of p-values. A non-adaptive version is available as well.

Usage

```
continuous.LR(
  test.results,
  alpha = 0.05,
  zeta = 0.5,
  adaptive = TRUE,
  critical.values = FALSE,
  select.threshold = 1
)

LR(
  test.results,
  alpha = 0.05,
  zeta = 0.5,
  critical.values = FALSE,
  select.threshold = 1
)

NLR(
  test.results,
  alpha = 0.05,
  zeta = 0.5,
  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 FDR procedure is to be performed.
`alpha`	single real number strictly between 0 and 1 specifying the target FDP.
`zeta`	single real number strictly between 0 and 1 specifying the target probability of not exceeding the desired FDP. If `zeta = NULL` (the default), then `zeta` is chosen equal to `alpha`.
`adaptive`	single boolean indicating whether to conduct an adaptive procedure or not.
`critical.values`	single boolean indicating 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 `threshold = 1` (the default), all raw p-values are selected.

Details

LR and NLR are wrapper functions for continuous.LR. The first one simply passes all its arguments to continuous.LR with adaptive = TRUE and NLR does the same with adaptive = FALSE.

Value

A FDX S3 class object whose elements are:

Rejected	rejected raw p-values.
Indices	indices of rejected p-values.
Num.rejected	number of rejections.
Adjusted	adjusted p-values.
Critical.values	critical values (only exists if computations where performed with critical.values = TRUE).
Select	list with data related to p-value selection; only exists if select.threshold < 1.
Select$Threshold	p-value selection threshold.
Select$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.
Data	list with input data.
Data$Method	character string describing the used algorithm, e.g. 'Discrete Lehmann-Romano procedure (step-up)'.
Data$Raw.pvalues	all observed raw p-values.
Data$FDP.threshold	FDP threshold alpha.
Data$Exceedance.probability	probability zeta of FDP exceeding alpha; thus, FDP is being controlled at level alpha with confidence 1 - zeta.
Data$Adaptive	boolean indicating whether an adaptive procedure was conducted or not.
Data$Data.name	the respective variable name(s) of the input data.

References

Lehmann, E. L. & Romano, J. P. (2005). Generalizations of the familywise error rate. *The Annals of Statistics*, *33*(3), pp. 1138-1154. doi:10.1214/009053605000000084

See Also

kernel(), FDX, continuous.GR(), discrete.LR(), discrete.GR(), discrete.PB(), weighted.LR(), weighted.GR(), weighted.PB()

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

# Construction of the 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()

# LR without critical values; using extracted p-values
LR.fast <- LR(raw.pvalues)
summary(LR.fast)

# LR with critical values; using test results object
LR.crit <- LR(test.results, critical.values = TRUE)
summary(LR.crit)

# Non-adaptive LR without critical values; using test results object
NLR.fast <- NLR(test.results)
summary(NLR.fast)

# Non-adaptive LR with critical values; using extracted p-values
NLR.crit <- NLR(raw.pvalues, critical.values = TRUE)
summary(NLR.crit)
```

DGR *Wrapper Functions for the Discrete Guo-Romano Procedure*

Description

DGR() is a wrapper function of discrete.GR() for computing [DGR]. It simply passes its arguments to discrete.GR() with fixed adaptive = TRUE.

Usage

```
DGR(test.results, ...)

## Default S3 method:
DGR(
  test.results,
  pCDFlist,
  alpha = 0.05,
  zeta = 0.5,
  critical.values = FALSE,
  select.threshold = 1,
  pCDFlist.indices = NULL,
  ...
)

## S3 method for class 'DiscreteTestResults'
DGR(
  test.results,
  alpha = 0.05,
  zeta = 0.5,
  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 FDR 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 specifying the target FDP.
`zeta`	single real number strictly between 0 and 1 specifying the target probability of not exceeding the desired FDP. If `zeta = NULL` (the default), then `zeta` is chosen equal to `alpha`.
`critical.values`	single boolean indicating 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 `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 each **unique** support in pCDFlist belongs; ignored if the lengths of test.results and pCDFlist are 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 area in a plot or other theoretical reasons.

Value

A FDX S3 class object whose elements are:

Rejected	rejected raw p-values.
Indices	indices of rejected p-values.
Num.rejected	number of rejections.
Adjusted	adjusted p-values.
Critical.values	critical values (only exists if computations where performed with critical.values = TRUE).
Select	list with data related to p-value selection; only exists if select.threshold < 1.
Select$Threshold	p-value selection threshold.
Select$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.
Data	list with input data.
Data$Method	character string describing the used algorithm, e.g. 'Discrete Lehmann-Romano procedure (step-up)'.
Data$Raw.pvalues	all observed raw p-values.
Data$pCDFlist	list of the p-value supports.
Data$FDP.threshold	FDP threshold alpha.
Data$Exceedance.probability	probability zeta of FDP exceeding alpha; thus, FDP is being controlled at level alpha with confidence 1 - zeta.
Data$Adaptive	boolean indicating whether an adaptive procedure was conducted or not.
Data$Data.name	the respective variable name(s) of the input data.

References

Döhler, S. & Roquain, E. (2020). Controlling False Discovery Exceedance for Heterogeneous Tests. *Electronic Journal of Statistics*, *14*(2), pp. 4244-4272. doi:10.1214/20EJS1771

See Also

discrete.GR(), NDGR(), discrete.LR(), DLR(), NDLR(), discrete.PB(), DPB(), NDPB()

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

# Construction of the 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()

# DGR without critical values; using extracted p-values and supports
DGR.fast <- DGR(raw.pvalues, pCDFlist)
summary(DGR.fast)

# DGR with critical values; using test results object
DGR.crit <- DGR(test.results, critical.values = TRUE)
summary(DGR.crit)
```

---

direct.discrete *Direct Application of Multiple Testing Procedures to Dataset*

Description

Apply the [DLR], [NDLR], [DGR], [NDGR], [PB] or [NPB] procedure, with or without computing the critical constants, to a data set of 2x2 contingency tables using a hypothesis test function from package DiscreteTests.

Usage

```
direct.discrete.LR(
  dat,
  test.fun,
  test.args = NULL,
```

```
  alpha = 0.05,
  zeta = 0.5,
  direction = "su",
  adaptive = FALSE,
  critical.values = FALSE,
  select.threshold = 1,
  preprocess.fun = NULL,
  preprocess.args = NULL
)

direct.discrete.GR(
  dat,
  test.fun,
  test.args = NULL,
  alpha = 0.05,
  zeta = 0.5,
  adaptive = FALSE,
  critical.values = FALSE,
  select.threshold = 1,
  preprocess.fun = NULL,
  preprocess.args = NULL
)

direct.discrete.PB(
  dat,
  test.fun,
  test.args = NULL,
  alpha = 0.05,
  zeta = 0.5,
  adaptive = FALSE,
  critical.values = FALSE,
  exact = TRUE,
  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 = 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 specifying the target FDP.

`zeta` single real number strictly between 0 and 1 specifying the target probability of not exceeding the desired FDP. If `zeta = NULL` (the default), then `zeta` is chosen equal to `alpha`.

`direction` single character string specifying whether to perform the step-up (`"su"`) or step-down (`"sd"`; the default) version of the Lehmann-Romano procedure.

`adaptive` single boolean indicating whether to conduct an adaptive procedure or not.

`critical.values`
single boolean indicating 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 `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!

`exact` single boolean indicating whether to compute the Poisson-Binomial distribution exactly or by normal approximation.

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

# Construction of the 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()

# DLR
DLR.sd <- direct.discrete.LR(df, "fisher")
summary(DLR.sd)

# Non-adaptive DLR (step-up variant; adjusted p-values do not exist here!)
NDLR.su <- direct.discrete.LR(df, "fisher", direction = "su", adaptive = FALSE)
```

```
summary(NDLR.su)

# DGR
DGR <- direct.discrete.GR(df, "fisher")
summary(DGR)

# Non-adaptive DGR
NDGR <- direct.discrete.GR(df, "fisher", adaptive = FALSE)
summary(NDGR)

# DPB (normal approximation)
PB.approx <- direct.discrete.PB(df, "fisher", exact = FALSE)
summary(DGR)

# Non-adaptive DPB
NPB.exact <- direct.discrete.GR(df, "fisher", adaptive = FALSE)
summary(NDGR)
```

`discrete.GR` *Discrete Guo-Romano procedure*

Description

Apply the [DGR] procedure, with or without computing the critical values, to a set of p-values and their discrete support. A non-adaptive version is available as well.

Usage

```
discrete.GR(test.results, ...)

## Default S3 method:
discrete.GR(
  test.results,
  pCDFlist,
  alpha = 0.05,
  zeta = 0.5,
  adaptive = TRUE,
  critical.values = FALSE,
  select.threshold = 1,
  pCDFlist.indices = NULL,
  ...
)

## S3 method for class 'DiscreteTestResults'
discrete.GR(
  test.results,
  alpha = 0.05,
```

```
  zeta = 0.5,
  adaptive = TRUE,
  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 FDR 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 specifying the target FDP.
`zeta`	single real number strictly between 0 and 1 specifying the target probability of not exceeding the desired FDP. If `zeta = NULL` (the default), then `zeta` is chosen equal to `alpha`.
`adaptive`	single boolean indicating whether to conduct an adaptive procedure or not.
`critical.values`	single boolean indicating 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 `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 each **unique** support in `pCDFlist` belongs; ignored if the lengths of `test.results` and `pCDFlist` are 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 area in a plot or other theoretical reasons.

Value

A `FDX` S3 class object whose elements are:

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

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

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

`Select$Threshold`
: p-value selection 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.

`Data`
: list with input data.

`Data$Method`
: character string describing the used algorithm, e.g. 'Discrete Lehmann-Romano procedure (step-up)'.

`Data$Raw.pvalues`
: all observed raw p-values.

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

`Data$FDP.threshold`
: FDP threshold `alpha`.

`Data$Exceedance.probability`
: probability `zeta` of FDP exceeding `alpha`; thus, FDP is being controlled at level `alpha` with confidence 1 - `zeta`.

`Data$Adaptive`
: boolean indicating whether an adaptive procedure was conducted or not.

`Data$Data.name`
: the respective variable name(s) of the input data.

References

Döhler, S. & Roquain, E. (2020). Controlling False Discovery Exceedance for Heterogeneous Tests. *Electronic Journal of Statistics*, *14*(2), pp. 4244-4272. doi:10.1214/20EJS1771

See Also

`FDX`, `discrete.LR()`, `discrete.PB()`, `continuous.LR()`, `continuous.GR()`, `weighted.LR()`, `weighted.GR()`, `weighted.PB()`

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
```

```
# Construction of the 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()

# DGR without critical values; using test results object
DGR.fast <- discrete.GR(test.results)
summary(DGR.fast)

# DGR with critical values; using extracted p-values and supports
DGR.crit <- discrete.GR(raw.pvalues, pCDFlist, critical.values = TRUE)
summary(DGR.crit)

# Non-Adaptive DGR without critical values; using extracted p-values and supports
NDGR.fast <- discrete.GR(raw.pvalues, pCDFlist, adaptive = FALSE)
summary(NDGR.fast)

# Non-Adaptive DGR without critical values; using test results object
NDGR.crit <- discrete.GR(test.results, adaptive = FALSE, critical.values = TRUE)
summary(NDGR.crit)
```

`discrete.LR` *Discrete Lehmann-Romano procedure*

Description

Apply the [DLR] procedure, with or without computing the critical values, to a set of p-values and their discrete support. Both step-down and step-up procedures can be computed and non-adaptive versions are available as well.

Usage

```
discrete.LR(test.results, ...)

## Default S3 method:
discrete.LR(
  test.results,
  pCDFlist,
  alpha = 0.05,
  zeta = 0.5,
  direction = "sd",
  adaptive = TRUE,
  critical.values = FALSE,
  select.threshold = 1,
  pCDFlist.indices = NULL,
  ...
```

```
)

## S3 method for class 'DiscreteTestResults'
discrete.LR(
  test.results,
  alpha = 0.05,
  zeta = 0.5,
  direction = "sd",
  adaptive = TRUE,
  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 FDR 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 specifying the target FDP.
`zeta`	single real number strictly between 0 and 1 specifying the target probability of not exceeding the desired FDP. If `zeta = NULL` (the default), then `zeta` is chosen equal to `alpha`.
`direction`	single character string specifying whether to perform the step-up (`"su"`) or step-down (`"sd"`; the default) version of the Lehmann-Romano procedure.
`adaptive`	single boolean indicating whether to conduct an adaptive procedure or not.
`critical.values`	single boolean indicating 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 `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 each **unique** support in `pCDFlist` belongs; ignored if the lengths of `test.results` and `pCDFlist` are 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 area in a plot or other theoretical reasons.

Value

A FDX S3 class object whose elements are:

Rejected	rejected raw p-values.
Indices	indices of rejected p-values.
Num.rejected	number of rejections.
Adjusted	adjusted p-values (only for step-down direction).
Critical.values	critical values (only exists if computations where performed with critical.values = TRUE).
Select	list with data related to p-value selection; only exists if select.threshold < 1.
Select$Threshold	p-value selection threshold.
Select$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.
Data	list with input data.
Data$Method	character string describing the used algorithm, e.g. 'Discrete Lehmann-Romano procedure (step-up)'.
Data$Raw.pvalues	all observed raw p-values.
Data$pCDFlist	list of the p-value supports.
Data$FDP.threshold	FDP threshold alpha.
Data$Exceedance.probability	probability zeta of FDP exceeding alpha; thus, FDP is being controlled at level alpha with confidence 1 - zeta.
Data$Adaptive	boolean indicating whether an adaptive procedure was conducted or not.
Data$Data.name	the respective variable name(s) of the input data.

References

Döhler, S. & Roquain, E. (2020). Controlling False Discovery Exceedance for Heterogeneous Tests. *Electronic Journal of Statistics*, *14*(2), pp. 4244-4272. doi:10.1214/20EJS1771

See Also

FDX, discrete.GR(), discrete.PB(), continuous.LR(), continuous.GR(), weighted.LR(), weighted.GR(), weighted.PB()

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

# Construction of the 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()

# DLR without critical values; using results object
DLR.sd.fast <- discrete.LR(test.results)
summary(DLR.sd.fast)

# DLR with critical values; using extracted p-values and supports
DLR.sd.crit <- discrete.LR(raw.pvalues, pCDFlist, critical.values = TRUE)
summary(DLR.sd.crit)

# DLR (step-up) without critical values; using extracted p-values and supports
DLR.su.fast <- discrete.LR(raw.pvalues, pCDFlist, direction = "su")
summary(DLR.su.fast)

# DLR (step-up) with critical values; using results object
DLR.su.crit <- discrete.LR(test.results, direction = "su",
                           critical.values = TRUE)
summary(DLR.su.crit)

# Non-adaptive DLR without critical values; using results object
NDLR.sd.fast <- discrete.LR(test.results, adaptive = FALSE)
summary(NDLR.sd.fast)

# Non-adaptive DLR with critical values; using extracted p-values and supports
NDLR.sd.crit <- discrete.LR(raw.pvalues, pCDFlist, adaptive = FALSE,
                            critical.values = TRUE)
summary(NDLR.sd.crit)

# Non-adaptive DLR (step-up) without critical values; using extracted p-values and supports
NDLR.su.fast <- discrete.LR(raw.pvalues, pCDFlist, direction = "su",
                            adaptive = FALSE)
summary(NDLR.su.fast)

# Non-adaptive DLR (step-up) with critical values; using results object
NDLR.su.crit <- discrete.LR(test.results, direction = "su",
                            adaptive = FALSE, critical.values = TRUE)
summary(NDLR.su.crit)
```

DLR *Wrapper Functions for the Discrete Guo-Romano Procedure*

Description

DLR() is a wrapper function of discrete.LR() for computing [DLR]. It simply passes its arguments to discrete.LR() with fixed adaptive = TRUE.

Usage

```
DLR(test.results, ...)

## Default S3 method:
DLR(
  test.results,
  pCDFlist,
  alpha = 0.05,
  zeta = 0.5,
  direction = "sd",
  critical.values = FALSE,
  select.threshold = 1,
  pCDFlist.indices = NULL,
  ...
)

## S3 method for class 'DiscreteTestResults'
DLR(
  test.results,
  alpha = 0.05,
  zeta = 0.5,
  direction = "sd",
  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 FDR 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 specifying the target FDP.

`zeta` single real number strictly between 0 and 1 specifying the target probability of not exceeding the desired FDP. If `zeta = NULL` (the default), then `zeta` is chosen equal to `alpha`.

`direction` single character string specifying whether to perform the step-up (`"su"`) or step-down (`"sd"`; the default) version of the Lehmann-Romano procedure.

`critical.values` single boolean indicating 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 `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 each **unique** support in `pCDFlist` belongs; ignored if the lengths of `test.results` and `pCDFlist` are 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 area in a plot or other theoretical reasons.

Value

A `FDX` S3 class object whose elements are:

`Rejected` rejected raw p-values.

`Indices` indices of rejected p-values.

`Num.rejected` number of rejections.

`Adjusted` adjusted p-values (only for step-down direction).

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

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

`Select$Threshold` p-value selection 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.

`Data` list with input data.

`Data$Method` character string describing the used algorithm, e.g. 'Discrete Lehmann-Romano procedure (step-up)'.

`Data$Raw.pvalues` all observed raw p-values.

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

`Data$FDP.threshold` FDP threshold `alpha`.

`Data$Exceedance.probability` probability `zeta` of FDP exceeding `alpha`; thus, FDP is being controlled at level `alpha` with confidence 1 - `zeta`.

`Data$Adaptive` boolean indicating whether an adaptive procedure was conducted or not.

`Data$Data.name` the respective variable name(s) of the input data.

References

Döhler, S. & Roquain, E. (2020). Controlling False Discovery Exceedance for Heterogeneous Tests. *Electronic Journal of Statistics*, *14*(2), pp. 4244-4272. doi:10.1214/20EJS1771

See Also

`discrete.LR()`, `NDLR()`, `discrete.GR()`, `DGR()`, `NDGR()`, `discrete.PB()`, `DPB()`, `NDPB()`

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

# Construction of the 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()

# DLR without critical values; using results object
DLR.sd.fast <- DLR(test.results)
summary(DLR.sd.fast)

# DLR with critical values; using extracted p-values and supports
DLR.sd.crit <- DLR(raw.pvalues, pCDFlist, critical.values = TRUE)
summary(DLR.sd.crit)

# DLR (step-up) without critical values; using extracted p-values and supports
DLR.su.fast <- DLR(raw.pvalues, pCDFlist, direction = "su")
summary(DLR.su.fast)
```

```
# DLR (step-up) with critical values; using test results object
DLR.su.crit <- DLR(test.results, direction = "su", critical.values = TRUE)
summary(DLR.su.crit)
```

DPB *Wrapper Functions for the Discrete Guo-Romano Procedure*

Description

DPB() is a wrapper function of discrete.PB() for computing [DPB]. It simply passes its arguments to discrete.PB() with fixed adaptive = TRUE.

Usage

```
DPB(test.results, ...)

## Default S3 method:
DPB(
  test.results,
  pCDFlist,
  alpha = 0.05,
  zeta = 0.5,
  critical.values = FALSE,
  exact = TRUE,
  select.threshold = 1,
  pCDFlist.indices = NULL,
  ...
)
```

Arguments

test.results	either a numeric vector with p-values or an R6 object of class DiscreteTestResults from package DiscreteTests for which a discrete FDR 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 specifying the target FDP.
zeta	single real number strictly between 0 and 1 specifying the target probability of not exceeding the desired FDP. If zeta = NULL (the default), then zeta is chosen equal to alpha.
critical.values	single boolean indicating whether critical constants are to be computed.

`exact` single boolean indicating whether to compute the Poisson-Binomial distribution exactly or by normal approximation.

`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 `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 each **unique** support in `pCDFlist` belongs; ignored if the lengths of `test.results` and `pCDFlist` are 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 area in a plot or other theoretical reasons.

Value

A `FDX` S3 class object whose elements are:

`Rejected` rejected raw p-values.

`Indices` indices of rejected p-values.

`Num.rejected` number of rejections.

`Adjusted` adjusted p-values.

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

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

`Select$Threshold`
: p-value selection 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.

`Data` list with input data.

`Data$Method` character string describing the used algorithm, e.g. 'Discrete Lehmann-Romano procedure (step-up)'.

`Data$Raw.pvalues`
: all observed raw p-values.

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

`Data$FDP.threshold`
: FDP threshold `alpha`.

`Data$Exceedance.probability`
: probability `zeta` of FDP exceeding `alpha`; thus, FDP is being controlled at level `alpha` with confidence 1 - `zeta`.

`Data$Adaptive`
: boolean indicating whether an adaptive procedure was conducted or not.

`Data$Data.name`
: the respective variable name(s) of the input data.

References

Döhler, S. & Roquain, E. (2020). Controlling False Discovery Exceedance for Heterogeneous Tests. *Electronic Journal of Statistics*, *14*(2), pp. 4244-4272. doi:10.1214/20EJS1771

See Also

`discrete.PB()`, `NDPB()`, `discrete.GR()`, `DGR()`, `NDGR()`, `discrete.LR()`, `DLR()`, `NDLR()`

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

# Construction of the 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()

# DPB (exact) without critical values; using results object
DPB.exact.fast <- DPB(test.results)
summary(DPB.exact.fast)

# DPB (exact) with critical values; using extracted p-values and supports
DPB.exact.crit <- DPB(raw.pvalues, pCDFlist, critical.values = TRUE)
summary(DPB.exact.crit)

# DPB (normal approximation) without critical values; using extracted p-values and supports
DPB.norm.fast <- DPB(raw.pvalues, pCDFlist, exact = FALSE)
summary(DPB.norm.fast)

# DPB (normal approximation) with critical values; using test results object
DPB.norm.crit <- DPB(test.results, critical.values = TRUE, exact = FALSE)
summary(DPB.norm.crit)
```

DPB.DiscreteTestResults

Discrete Poisson-Binomial procedure

Description

Apply the [DPB] procedure, with or without computing the critical values, to a set of p-values and their discrete support. A non-adaptive version is available as well. Additionally, the user can choose between exact computation of the Poisson-Binomial distribution or a refined normal approximation.

Usage

```
## S3 method for class 'DiscreteTestResults'
DPB(
  test.results,
  alpha = 0.05,
  zeta = 0.5,
  critical.values = FALSE,
  exact = TRUE,
  select.threshold = 1,
  ...
)

discrete.PB(test.results, ...)

## Default S3 method:
discrete.PB(
  test.results,
  pCDFlist,
  alpha = 0.05,
  zeta = 0.5,
  adaptive = TRUE,
  critical.values = FALSE,
  exact = TRUE,
  select.threshold = 1,
  pCDFlist.indices = NULL,
  ...
)

## S3 method for class 'DiscreteTestResults'
discrete.PB(
  test.results,
  alpha = 0.05,
  zeta = 0.5,
  adaptive = TRUE,
  critical.values = FALSE,
  exact = TRUE,
```

```
  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 FDR procedure is to be performed.
alpha	single real number strictly between 0 and 1 specifying the target FDP.
zeta	single real number strictly between 0 and 1 specifying the target probability of not exceeding the desired FDP. If zeta = NULL (the default), then zeta is chosen equal to alpha.
critical.values	single boolean indicating whether critical constants are to be computed.
exact	single boolean indicating whether to compute the Poisson-Binomial distribution exactly or by normal approximation.
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 threshold = 1 (the default), all raw p-values are selected.
...	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.
adaptive	single boolean indicating whether to conduct an adaptive procedure or not.
pCDFlist.indices	list of numeric vectors containing the test indices that indicate to which raw p-value each **unique** support in pCDFlist belongs; ignored if the lengths of test.results and pCDFlist are equal.

Details

DPB and NDPB are wrapper functions for discrete.PB. The first one simply passes all its arguments to discrete.PB with adaptive = TRUE and NDPB does the same with adaptive = FALSE.

Value

A FDX S3 class object whose elements are:

Rejected	rejected raw p-values.
Indices	indices of rejected p-values.
Num.rejected	number of rejections.
Adjusted	adjusted p-values.
Critical.values	critical values (only exists if computations where performed with critical.values = TRUE).

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

`Select$Threshold`
p-value selection 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.

`Data` list with input data.

`Data$Method` character string describing the used algorithm, e.g. 'Discrete Lehmann-Romano procedure (step-up)'.

`Data$Raw.pvalues`
all observed raw p-values.

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

`Data$FDP.threshold`
FDP threshold `alpha`.

`Data$Exceedance.probability`
probability `zeta` of FDP exceeding `alpha`; thus, FDP is being controlled at level `alpha` with confidence 1 - `zeta`.

`Data$Data.name` the respective variable name(s) of the input data.

References

Döhler, S. & Roquain, E. (2020). Controlling False Discovery Exceedance for Heterogeneous Tests. *Electronic Journal of Statistics*, *14*(2), pp. 4244-4272. doi:10.1214/20EJS1771

See Also

`kernel`, `FDX`, `continuous.LR()`, `continuous.GR()`, `discrete.LR()`, `discrete.GR()`, `weighted.LR()`, `weighted.GR()`, `weighted.PB()`

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

# Construction of the 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()

# DPB (exact) without critical values; using results object
DPB.exact.fast <- discrete.PB(test.results)
summary(DPB.exact.fast)

# DPB (exact) with critical values; using extracted p-values and supports
DPB.exact.crit <- discrete.PB(raw.pvalues, pCDFlist, critical.values = TRUE)
summary(DPB.exact.crit)

# DPB (normal approximation) without critical values; using extracted p-values and supports
DPB.norm.fast <- discrete.PB(raw.pvalues, pCDFlist, exact = FALSE)
summary(DPB.norm.fast)

# DPB (normal approximation) with critical values; using results object
DPB.norm.crit <- discrete.PB(test.results, critical.values = TRUE,
                             exact = FALSE)
summary(DPB.norm.crit)

# Non-adaptive DPB (exact) without critical values; using results object
NDPB.exact.fast <- discrete.PB(test.results, adaptive = FALSE)
summary(NDPB.exact.fast)

# Non-adaptive DPB (exact) with critical values; using extracted p-values and supports
NDPB.exact.crit <- discrete.PB(raw.pvalues, pCDFlist, adaptive = FALSE,
                               critical.values = TRUE)
summary(NDPB.exact.crit)

# Non-adaptive DPB (normal approx.) without critical values; using extracted p-values and supports
NDPB.norm.fast <- discrete.PB(raw.pvalues, pCDFlist, adaptive = FALSE,
                              exact = FALSE)
summary(NDPB.norm.fast)

# Non-adaptive DPB (normal approx.) with critical values; using results object
NDPB.norm.crit <- discrete.PB(test.results, adaptive = FALSE,
                              critical.values = TRUE, exact = FALSE)
summary(NDPB.norm.crit)
```

`fast.Discrete` *Fast application of discrete procedures*

Description

[Deprecated]

Applies the [DLR], [DGR] or [DPB] procedures, **without** computing the critical values, to a data set of 2 x 2 contingency tables using Fisher's exact test.

Note: These functions are deprecated and will be removed in a future version. Please use `direct.discrete.*()` with `test.fun = DiscreteTests::fisher.test.pv` and (optional) `preprocess.fun = DiscreteDatasets::reconstruc`

or preprocess.fun = DiscreteDatasets::reconstruct_four instead. Alternatively, use a pipeline like
data |>
DiscreteDatasets::reconstruct_*(<args>) |>
DiscreteTests::*.test.pv(<args>) |>
discrete.*(<args>).

Usage

```
fast.Discrete.LR(
  counts,
  alternative = "greater",
  input = "noassoc",
  alpha = 0.05,
  zeta = 0.5,
  direction = "sd",
  adaptive = TRUE
)

fast.Discrete.GR(
  counts,
  alternative = "greater",
  input = "noassoc",
  alpha = 0.05,
  zeta = 0.5,
  adaptive = TRUE
)

fast.Discrete.PB(
  counts,
  alternative = "greater",
  input = "noassoc",
  alpha = 0.05,
  zeta = 0.5,
  adaptive = TRUE,
  exact = FALSE
)
```

Arguments

counts	a data frame of 2 or 4 columns and any number of lines, each line representing a 2 x 2 contingency table to test. The number of columns and what they must contain depend on the value of the input argument, see Details of DiscreteFDR::fisher.pvalues.support
alternative	same argument as in fisher.test(). The three possible values are "greater" (default), "two.sided" or "less"; may be abbreviated.
input	the format of the input data frame, see Details of DiscreteFDR::fisher.pvalues.support(). The three possible values are "noassoc" (default), "marginal" or "HG2011"; may be abbreviated.

alpha	single real number strictly between 0 and 1 specifying the target FDP.
zeta	single real number strictly between 0 and 1 specifying the target probability of not exceeding the desired FDP. If zeta = NULL (the default), then zeta is chosen equal to alpha.
direction	single character string specifying whether to perform the step-up ("su") or step-down ("sd"; the default) version of the Lehmann-Romano procedure.
adaptive	single boolean indicating whether to conduct an adaptive procedure or not.
exact	single boolean indicating whether to compute the Poisson-Binomial distribution exactly or by normal approximation.

Value

A FDX S3 class object whose elements are:

Rejected	rejected raw p-values.
Indices	indices of rejected p-values.
Num.rejected	number of rejections.
Adjusted	adjusted p-values (only for step-down direction).
Critical.values	critical values (only exists if computations where performed with critical.values = TRUE).
Select	list with data related to p-value selection; only exists if select.threshold < 1.
Select$Threshold	p-value selection threshold.
Select$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.
Data	list with input data.
Data$Method	character string describing the used algorithm, e.g. 'Discrete Lehmann-Romano procedure (step-up)'.
Data$Raw.pvalues	all observed raw p-values.
Data$FDP.threshold	FDP threshold alpha.
Data$Exceedance.probability	probability zeta of FDP exceeding alpha; thus, FDP is being controlled at level alpha with confidence 1 - zeta.
Data$Adaptive	boolean indicating whether an adaptive procedure was conducted or not.
Data$Data.name	the respective variable name(s) of the input data.

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

# DLR
DLR.sd <- fast.Discrete.LR(counts = df, input = "noassoc")
summary(DLR.sd)

# DLR
DLR.su <- fast.Discrete.LR(counts = df, input = "noassoc", direction = "su")
summary(DLR.su)

# Non-adaptive DLR
NDLR.sd <- fast.Discrete.LR(counts = df, input = "noassoc", adaptive = FALSE)
summary(NDLR.sd)

# Non-adaptive DLR
NDLR.su <- fast.Discrete.LR(counts = df, input = "noassoc", direction = "su", adaptive = FALSE)
summary(NDLR.su)

# DGR
DGR <- fast.Discrete.GR(counts = df, input = "noassoc")
summary(DGR)

# Non-adaptive DGR
NDGR <- fast.Discrete.GR(counts = df, input = "noassoc", adaptive = FALSE)
summary(NDGR)

# DPB
DPB <- fast.Discrete.PB(counts = df, input = "noassoc")
summary(DPB)

# Non-adaptive DPB
NDPB <- fast.Discrete.PB(counts = df, input = "noassoc", adaptive = FALSE)
summary(NDPB)
```

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

Description

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

Usage

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

Arguments

x	object of class FDX.
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 be one of "raw", "selected" or "weighted".
...	further arguments to graphics::hist() or graphics::plot.histogram(), respectively.

Details

If x does not contain results of a weighting or 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

# Construction of the 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()

# DGR
DGR <- DGR(raw.pvalues, pCDFlist)
# histogram of raw p-values
hist(DGR)

# arithmetic-weighted GR (using 1 - raw.pvalues as weights)
wGR <- wGR.AM(raw.pvalues, 1 - raw.pvalues)
# histogram of raw p-values
hist(wGR)
# histogram of weighted p-values
```

```
hist(wGR, mode = "weighted")
```

NDGR *Wrapper Functions for the Non-Adaptive Discrete Guo-Romano Procedure*

Description

`NDGR()` is a wrapper function of `discrete.GR()` for computing non-adaptive [DGR]. It simply passes its arguments to `discrete.GR()` with fixed `adaptive = FALSE`.

Usage

```
NDGR(test.results, ...)

## Default S3 method:
NDGR(
  test.results,
  pCDFlist,
  alpha = 0.05,
  zeta = 0.5,
  critical.values = FALSE,
  select.threshold = 1,
  pCDFlist.indices = NULL,
  ...
)

## S3 method for class 'DiscreteTestResults'
NDGR(
  test.results,
  alpha = 0.05,
  zeta = 0.5,
  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 FDR 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 specifying the target FDP.

`zeta` single real number strictly between 0 and 1 specifying the target probability of not exceeding the desired FDP. If `zeta = NULL` (the default), then `zeta` is chosen equal to `alpha`.

`critical.values` single boolean indicating 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 `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 each **unique** support in `pCDFlist` belongs; ignored if the lengths of `test.results` and `pCDFlist` are 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 area in a plot or other theoretical reasons.

Value

A `FDX` S3 class object whose elements are:

`Rejected` rejected raw p-values.

`Indices` indices of rejected p-values.

`Num.rejected` number of rejections.

`Adjusted` adjusted p-values.

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

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

`Select$Threshold` p-value selection 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.

`Data` list with input data.

`Data$Method` character string describing the used algorithm, e.g. 'Discrete Lehmann-Romano procedure (step-up)'.

`Data$Raw.pvalues`
: all observed raw p-values.

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

`Data$FDP.threshold`
: FDP threshold `alpha`.

`Data$Exceedance.probability`
: probability `zeta` of FDP exceeding `alpha`; thus, FDP is being controlled at level `alpha` with confidence 1 - `zeta`.

`Data$Adaptive`
: boolean indicating whether an adaptive procedure was conducted or not.

`Data$Data.name`
: the respective variable name(s) of the input data.

References

Döhler, S. & Roquain, E. (2020). Controlling False Discovery Exceedance for Heterogeneous Tests. *Electronic Journal of Statistics*, *14*(2), pp. 4244-4272. doi:10.1214/20EJS1771

See Also

`discrete.GR()`, `DGR()`, `discrete.LR()`, `DLR()`, `NDLR()`, `discrete.PB()`, `DPB()`, `NDPB()`

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

# Construction of the 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()

# Non-adaptive DGR without critical values; using extracted p-values and supports
NDGR.fast <- NDGR(raw.pvalues, pCDFlist)
summary(NDGR.fast)

# Non-adaptive DGR with critical values; using test results object
NDGR.crit <- NDGR(test.results, critical.values = TRUE)
summary(NDGR.crit)
```

NDLR *Wrapper Functions for the Non-Adaptive Discrete Guo-Romano Procedure*

Description

NDLR() is a wrapper function of discrete.LR() for computing non-adaptive [DLR]. It simply passes its arguments to discrete.LR() with fixed adaptive = FALSE.

Usage

```
NDLR(test.results, ...)

## Default S3 method:
NDLR(
  test.results,
  pCDFlist,
  alpha = 0.05,
  zeta = 0.5,
  direction = "sd",
  critical.values = FALSE,
  select.threshold = 1,
  pCDFlist.indices = NULL,
  ...
)

## S3 method for class 'DiscreteTestResults'
NDLR(
  test.results,
  alpha = 0.05,
  zeta = 0.5,
  direction = "sd",
  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 FDR 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 specifying the target FDP.

`zeta` single real number strictly between 0 and 1 specifying the target probability of not exceeding the desired FDP. If `zeta = NULL` (the default), then `zeta` is chosen equal to `alpha`.

`direction` single character string specifying whether to perform the step-up (`"su"`) or step-down (`"sd"`; the default) version of the Lehmann-Romano procedure.

`critical.values` single boolean indicating 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 `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 each **unique** support in `pCDFlist` belongs; ignored if the lengths of `test.results` and `pCDFlist` are 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 area in a plot or other theoretical reasons.

Value

A `FDX` S3 class object whose elements are:

`Rejected` rejected raw p-values.

`Indices` indices of rejected p-values.

`Num.rejected` number of rejections.

`Adjusted` adjusted p-values (only for step-down direction).

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

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

`Select$Threshold` p-value selection 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.

`Data` list with input data.

`Data$Method` character string describing the used algorithm, e.g. 'Discrete Lehmann-Romano procedure (step-up)'.

`Data$Raw.pvalues` all observed raw p-values.

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

`Data$FDP.threshold` FDP threshold `alpha`.

`Data$Exceedance.probability` probability `zeta` of FDP exceeding `alpha`; thus, FDP is being controlled at level `alpha` with confidence 1 - `zeta`.

`Data$Adaptive` boolean indicating whether an adaptive procedure was conducted or not.

`Data$Data.name` the respective variable name(s) of the input data.

References

Döhler, S. & Roquain, E. (2020). Controlling False Discovery Exceedance for Heterogeneous Tests. *Electronic Journal of Statistics*, *14*(2), pp. 4244-4272. doi:10.1214/20EJS1771

See Also

`discrete.GR()`, `DGR()`, `discrete.LR()`, `DLR()`, `NDLR()`, `discrete.PB()`, `DPB()`, `NDPB()`

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

# Construction of the 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()

# Non-adaptive DLR without critical values; using results object
NDLR.sd.fast <- NDLR(test.results)
summary(NDLR.sd.fast)

# Non-adaptive DLR with critical values; using extracted p-values and supports
NDLR.sd.crit <- NDLR(raw.pvalues, pCDFlist, critical.values = TRUE)
summary(NDLR.sd.crit)

# Non-adaptive DLR (step-up) without critical values; using extracted p-values and supports
NDLR.su.fast <- NDLR(raw.pvalues, pCDFlist, direction = "su")
summary(NDLR.su.fast)
```

```
# Non-adaptive DLR (step-up) with critical values; using test results object
NDLR.su.crit <- NDLR(test.results, direction = "su", critical.values = TRUE)
summary(NDLR.su.crit)
```

NDPB *Wrapper Functions for the Non-Adaptive Discrete Guo-Romano Procedure*

Description

NDPB() is a wrapper function of discrete.PB() for computing non-adaptive [DPB]. It simply passes its arguments to discrete.PB() with fixed adaptive = FALSE.

Usage

```
NDPB(test.results, ...)

## Default S3 method:
NDPB(
  test.results,
  pCDFlist,
  alpha = 0.05,
  zeta = 0.5,
  critical.values = FALSE,
  exact = TRUE,
  select.threshold = 1,
  pCDFlist.indices = NULL,
  ...
)

## S3 method for class 'DiscreteTestResults'
NDPB(
  test.results,
  alpha = 0.05,
  zeta = 0.5,
  critical.values = FALSE,
  exact = TRUE,
  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 FDR 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 specifying the target FDP.

`zeta` single real number strictly between 0 and 1 specifying the target probability of not exceeding the desired FDP. If `zeta = NULL` (the default), then `zeta` is chosen equal to `alpha`.

`critical.values` single boolean indicating whether critical constants are to be computed.

`exact` single boolean indicating whether to compute the Poisson-Binomial distribution exactly or by normal approximation.

`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 `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 each **unique** support in `pCDFlist` belongs; ignored if the lengths of `test.results` and `pCDFlist` are 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 area in a plot or other theoretical reasons.

Value

A `FDX` S3 class object whose elements are:

`Rejected` rejected raw p-values.

`Indices` indices of rejected p-values.

`Num.rejected` number of rejections.

`Adjusted` adjusted p-values.

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

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

`Select$Threshold` p-value selection 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.
`Data`	list with input data.
`Data$Method`	character string describing the used algorithm, e.g. 'Discrete Lehmann-Romano procedure (step-up)'.
`Data$Raw.pvalues`	all observed raw p-values.
`Data$pCDFlist`	list of the p-value supports.
`Data$FDP.threshold`	FDP threshold `alpha`.
`Data$Exceedance.probability`	probability `zeta` of FDP exceeding `alpha`; thus, FDP is being controlled at level `alpha` with confidence 1 - `zeta`.
`Data$Adaptive`	boolean indicating whether an adaptive procedure was conducted or not.
`Data$Data.name`	the respective variable name(s) of the input data.

References

Döhler, S. & Roquain, E. (2020). Controlling False Discovery Exceedance for Heterogeneous Tests. *Electronic Journal of Statistics*, *14*(2), pp. 4244-4272. doi:10.1214/20EJS1771

See Also

`discrete.PB()`, `DPB()`, `discrete.GR()`, `DGR()`, `NDGR()`, `discrete.LR()`, `DLR()`, `NDLR()`

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

# Construction of the 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()

# Non-adaptive DPB (exact) without critical values; using results object
NDPB.exact.fast <- NDPB(test.results)
summary(NDPB.exact.fast)

# Non-adaptive DPB (exact) with critical values; using extracted p-values and supports
```

```
NDPB.exact.crit <- NDPB(raw.pvalues, pCDFlist, critical.values = TRUE)
summary(NDPB.exact.crit)

# Non-adaptive DPB (normal approx.) without critical values; using extracted p-values and supports
NDPB.norm.fast <- NDPB(raw.pvalues, pCDFlist, exact = FALSE)
summary(NDPB.norm.fast)

# Non-adaptive DPB (normal approx.) with critical values; using test results object
NDPB.norm.crit <- NDPB(test.results, critical.values = TRUE, exact = FALSE)
summary(NDPB.norm.crit)
```

`plot.FDX` *Plot Method for* `FDX` *objects*

Description

Plots raw p-values of a `FDX` object and highlights rejected and non-rejected p-values. If present, the critical values are plotted, too.

Usage

```
## S3 method for class 'FDX'
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`	an object of class `"FDX"`.
`col`	numeric or character vector of length 3 indicating the colors of the 1. rejected p-values 2. non-rejected 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. non-rejected 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()$lwd setting for all components.
cex	numeric vector of length 3 indicating the size of point characters or lines of the 1. rejected p-values 2. accepted p-values 3. critical values (if present). defaults to current par()$cex setting for all components.
type.crit	single character giving the type of plot desired for the critical values (e.g.: 'p', 'l' etc; see graphics::plot.default()).
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 graphics::plot.default().

Details

If x contains results of a weighted approach, the Y-axis of the plot is derived from the weighted p-values. Otherwise, it is constituted by the raw ones.

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

# Construction of the 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()

# DLR without critical values; using extracted p-values and supports
DLR.sd.fast <- DLR(raw.pvalues, pCDFlist)
# plot with default settings
plot(DLR.sd.fast)

# DLR (step-up) with critical values; using test results object
DLR.su.crit <- DLR(test.results, direction = "su", critical.values = TRUE)
# limited plot range
plot(DLR.su.crit, xlim = c(1, 5), ylim = c(0, 0.4))

# DPB without critical values; using test results object
DPB.fast <- DPB(test.results)
# limited plot range, custom colors, line widths and point symbols, top-left legend
plot(DPB.fast, col = c(2, 4), pch = c(2, 3), lwd = c(2, 2),
```

```
     legend = "topleft", xlim = c(1, 5), ylim = c(0, 0.4))

# DGR with critical values; using extracted p-values and supports
DGR.crit <- DGR(raw.pvalues, pCDFlist, critical.values = TRUE)
# additional customized plot parameters
plot(DGR.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), cex = c(3, 3, 2))
```

`print.FDX` *Printing FDX results*

Description

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

Usage

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

Arguments

`x`	object of class `FDX`.
`...`	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

# Construction of the 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()
```

```
# DPB with critical values; using test results object
DPB.crit <- DPB(test.results, critical.values = TRUE)
# print results
print(DPB.crit)
```

`rejection.path` *Rejection Path Plot (for* `FDX` *objects)*

Description

Displays the number of rejections of the raw p-values in a `FDX` object in dependence of the exceedance probability `zeta`.

Usage

```
rejection.path(
  x,
  xlim = NULL,
  ylim = NULL,
  main = NULL,
  xlab = expression(zeta),
  ylab = "Number of Rejections",
  verticals = FALSE,
  pch = 19,
  ref.show = FALSE,
  ref.col = "gray",
  ref.lty = 2,
  ref.lwd = 2,
  ...
)
```

Arguments

`x`	object of class `"FDX"`.
`xlim`	x axis limits of the plot. If `NULL` (default), the [0, 1] range is used.
`ylim`	the y limits of the plot. If `NULL` (default), the double of the median of the number of possible rejections is used as upper limit.
`main`	main title. If `NULL` (default), a description string is used.
`xlab`, `ylab`	labels for x and y axis.
`verticals`	logical; if `TRUE`, draw vertical lines at steps.
`pch`	jump point character.
`ref.show`	logical; if `TRUE` a vertical reference line is plotted, whose height is the number of rejections of the original Benjamini-Hochberg (BH) procedure.
`ref.col`	color of the reference line.

`ref.lty`, `ref.lwd`
: line type and thickness for the reference line.

`...`
: further arguments to `stats::plot.stepfun()`.

Value

Invisibly returns a `stepfun` object that computes the number of rejections in dependence on the exceedance probability `zeta`.

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

# Construction of the 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()

# DLR without critical values; using extracted p-values and supports
DLR <- DLR(raw.pvalues, pCDFlist)
# plot number of rejections dependent on the exceedance probability zeta
rejection.path(DLR, xlim = c(0, 1), ref.show = TRUE, ref.col = "green", ref.lty = 4)

# None-adaptive DLR without critical values; using test results object
NDLR <- NDLR(test.results)
# add plot for non-adaptive procedure (in red)
rejection.path(NDLR, col = "red", add = TRUE)
```

---

summary.FDX *Summarizing Discrete FDX Results*

Description

`summary` method for class `FDX`

Usage

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

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

Arguments

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

Details

summary.FDX objects include all data of an FDX class 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.FDX simply prints the same output as print.FDX, but also prints the p-value table.

Value

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

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

print.summary.FDX returns that object invisibly.

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

# Construction of the 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()

# DGR with critical values; using test results object
DGR.crit <- DGR(test.results, critical.values = TRUE)
# create summary
DGR.crit.summary <- summary(DGR.crit)
# print summary
print(DGR.crit.summary)
```

weighted.GR *Weighted Guo-Romano Procedure*

Description

Apply the weighted [wGR] procedure, with or without computing the critical values, to a set of p-values. Both arithmetic and geometric weighting are available.

Usage

```
weighted.GR(
  test.results,
  weights = NULL,
  alpha = 0.05,
  zeta = 0.5,
  weighting.method = c("AM", "GM"),
  critical.values = FALSE,
  select.threshold = 1
)

wGR.AM(
  test.results,
  weights,
  alpha = 0.05,
  zeta = 0.5,
  critical.values = FALSE,
  select.threshold = 1
)

wGR.GM(
  test.results,
  weights,
  alpha = 0.05,
  zeta = 0.5,
  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 FDR procedure is to be performed.
weights	numeric vector that contains the weights for the p-values.
alpha	single real number strictly between 0 and 1 specifying the target FDP.

`zeta` single real number strictly between 0 and 1 specifying the target probability of not exceeding the desired FDP. If `zeta = NULL` (the default), then `zeta` is chosen equal to `alpha`.

`weighting.method` single character string specifying whether to conduct arithmetic (`direction = "AM"`, the default) or geometric weighting (`direction = "GM"`) of p-values.

`critical.values` single boolean indicating 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 `threshold = 1` (the default), all raw p-values are selected.

Details

`wGR.AM` and `wGR.GM` are wrapper functions for `weighted.GR`. The first one simply passes all its arguments to `weighted.GR` with `weighting.method = "AM"` and `wGR.GM` does the same with `weighting.method = "GM"`.

Value

A `FDX` S3 class object whose elements are:

`Rejected` rejected raw p-values.

`Indices` indices of rejected p-values.

`Num.rejected` number of rejections.

`Weighted` weighted p-values.

`Adjusted` adjusted p-values.

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

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

`Select$Threshold` p-value selection 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.

`Data` list with input data.

`Data$Method` character string describing the used algorithm, e.g. 'Discrete Lehmann-Romano procedure (step-up)'.

`Data$Raw.pvalues`
: all observed raw p-values.

`Data$Weights`
: the weights for the raw p-values.

`Data$FDP.threshold`
: FDP threshold `alpha`.

`Data$Exceedance.probability`
: probability `zeta` of FDP exceeding `alpha`; thus, FDP is being controlled at level `alpha` with confidence 1 - `zeta`.

`Data$Weighting`
: character string describing the weighting method.

`Data$Data.name`
: the respective variable name(s) of the input data.

References

Döhler, S. & Roquain, E. (2020). Controlling False Discovery Exceedance for Heterogeneous Tests. *Electronic Journal of Statistics*, *14*(2), pp. 4244-4272. doi:10.1214/20EJS1771

See Also

`kernel`, `FDX`, `continuous.LR()`, `continuous.GR()`, `discrete.LR()`, `discrete.GR()`, `discrete.PB()`, `weighted.LR()`, `weighted.PB()`

Examples

```
# Construction of the p-values and their supports for weighted methods
raw.pvalues.weighted <- c(0.7389727, 0.1882310, 0.1302457, 0.9513677,
                          0.7592122, 0.0100559, 0.0000027, 0.1651034)
weights <- c(0.7947122, 1.2633867, 2.8097858, 2.2112801,
             2.3878654, 1.2389620, 2.3878654, 0.7947122)

# arithmetic-weighted Guo-Romano procedure without critical values
wGR.AM.fast <- wGR.AM(raw.pvalues.weighted, weights)
summary(wGR.AM.fast)

# arithmetic-weighted Guo-Romano procedure with critical values
wGR.AM.crit <- wGR.AM(raw.pvalues.weighted, weights, critical.values = TRUE)
summary(wGR.AM.crit)

# geometric-weighted Guo-Romano procedure without critical values
wGR.GM.fast <- wGR.GM(raw.pvalues.weighted, weights)
summary(wGR.GM.fast)

# geometric-weighted Guo-Romano procedure with critical values
wGR.GM.crit <- wGR.GM(raw.pvalues.weighted, weights, critical.values = TRUE)
summary(wGR.GM.crit)
```

`weighted.LR` *Weighted Lehmann-Romano Procedure*

Description

Apply the weighted [wLR] procedure, with or without computing the critical values, to a set of p-values. Both arithmetic and geometric weighting are available.

Usage

```
weighted.LR(
  test.results,
  weights = NULL,
  alpha = 0.05,
  zeta = 0.5,
  weighting.method = c("AM", "GM"),
  critical.values = FALSE,
  select.threshold = 1
)

wLR.AM(
  test.results,
  weights,
  alpha = 0.05,
  zeta = 0.5,
  critical.values = FALSE,
  select.threshold = 1
)

wLR.GM(
  test.results,
  weights,
  alpha = 0.05,
  zeta = 0.5,
  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 FDR procedure is to be performed.
`weights`	numeric vector that contains the weights for the p-values.
`alpha`	single real number strictly between 0 and 1 specifying the target FDP.

`zeta` single real number strictly between 0 and 1 specifying the target probability of not exceeding the desired FDP. If `zeta = NULL` (the default), then `zeta` is chosen equal to `alpha`.

`weighting.method` single character string specifying whether to conduct arithmetic (`direction = "AM"`, the default) or geometric weighting (`direction = "GM"`) of p-values.

`critical.values` single boolean indicating 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 `threshold = 1` (the default), all raw p-values are selected.

Details

`wLR.AM` and `wLR.GM` are wrapper functions for `weighted.LR`. The first one simply passes all its arguments to `weighted.LR` with `weighting.method = "AM"` and `wLR.GM` does the same with `weighting.method = "GM"`.

Value

A `FDX` S3 class object whose elements are:

`Rejected` rejected raw p-values.

`Indices` indices of rejected p-values.

`Num.rejected` number of rejections.

`Weighted` weighted p-values.

`Adjusted` adjusted p-values.

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

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

`Select$Threshold` p-value selection 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.

`Data` list with input data.

`Data$Method` character string describing the used algorithm, e.g. 'Discrete Lehmann-Romano procedure (step-up)'.

`Data$Raw.pvalues`
: all observed raw p-values.

`Data$Weights`
: the weights for the raw p-values.

`Data$FDP.threshold`
: FDP threshold `alpha`.

`Data$Exceedance.probability`
: probability `zeta` of FDP exceeding `alpha`; thus, FDP is being controlled at level `alpha` with confidence 1 - `zeta`.

`Data$Weighting`
: character string describing the weighting method.

`Data$Data.name`
: the respective variable name(s) of the input data.

References

Döhler, S. & Roquain, E. (2020). Controlling False Discovery Exceedance for Heterogeneous Tests. *Electronic Journal of Statistics*, *14*(2), pp. 4244-4272. doi:10.1214/20EJS1771

See Also

`kernel`, `FDX`, `continuous.LR()`, `continuous.GR()`, `discrete.LR()`, `discrete.GR()`, `discrete.PB()`, `weighted.GR()`, `weighted.PB()`

Examples

```
# Construction of the p-values and their supports for weighted methods
raw.pvalues.weighted <- c(0.7389727, 0.1882310, 0.1302457, 0.9513677,
                          0.7592122, 0.0100559, 0.0000027, 0.1651034)
weights <- c(0.7947122, 1.2633867, 2.8097858, 2.2112801,
             2.3878654, 1.2389620, 2.3878654, 0.7947122)

# arithmetic-weighted Lehmann-Romano procedure without critical values
wLR.AM.fast <- wLR.AM(raw.pvalues.weighted, weights)
summary(wLR.AM.fast)

# arithmetic-weighted Lehmann-Romano procedure with critical values
wLR.AM.crit <- wLR.AM(raw.pvalues.weighted, weights, critical.values = TRUE)
summary(wLR.AM.crit)

# geometric-weighted Lehmann-Romano procedure without critical values
wLR.GM.fast <- wLR.GM(raw.pvalues.weighted, weights)
summary(wLR.GM.fast)

# geometric-weighted Lehmann-Romano procedure with critical values
wLR.GM.crit <- wLR.GM(raw.pvalues.weighted, weights, critical.values = TRUE)
summary(wLR.GM.crit)
```

`weighted.PB` *Weighted Poisson-Binomial Procedure*

Description

Apply the weighted [wPB] procedure, with or without computing the critical values, to a set of p-values. Both arithmetic and geometric weighting are available. Additionally, the user can choose between exact computation of the Poisson-Binomial distribution or a refined normal approximation.

Usage

```
weighted.PB(
  test.results,
  weights = NULL,
  alpha = 0.05,
  zeta = 0.5,
  weighting.method = c("AM", "GM"),
  critical.values = FALSE,
  exact = TRUE,
  select.threshold = 1
)

wPB.AM(
  test.results,
  weights,
  alpha = 0.05,
  zeta = 0.5,
  critical.values = FALSE,
  exact = TRUE,
  select.threshold = 1
)

wPB.GM(
  test.results,
  weights,
  alpha = 0.05,
  zeta = 0.5,
  critical.values = FALSE,
  exact = TRUE,
  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 FDR procedure is to be performed.

`weights` numeric vector that contains the weights for the p-values.

`alpha` single real number strictly between 0 and 1 specifying the target FDP.

`zeta` single real number strictly between 0 and 1 specifying the target probability of not exceeding the desired FDP. If `zeta = NULL` (the default), then `zeta` is chosen equal to `alpha`.

`weighting.method` single character string specifying whether to conduct arithmetic (`direction = "AM"`, the default) or geometric weighting (`direction = "GM"`) of p-values.

`critical.values` single boolean indicating whether critical constants are to be computed.

`exact` single boolean indicating whether to compute the Poisson-Binomial distribution exactly or by normal approximation.

`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 `threshold = 1` (the default), all raw p-values are selected.

Details

`wPB.AM` and `wPB.GM` are wrapper functions for `weighted.PB`. The first one simply passes all its arguments to `weighted.PB` with `weighting.method = "AM"` and `wPB.GM` does the same with `weighting.method = "GM"`.

Value

A `FDX` S3 class object whose elements are:

`Rejected` rejected raw p-values.

`Indices` indices of rejected p-values.

`Num.rejected` number of rejections.

`Weighted` weighted p-values.

`Adjusted` adjusted p-values.

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

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

`Select$Threshold` p-value selection 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.
`Data`	list with input data.
`Data$Method`	character string describing the used algorithm, e.g. 'Discrete Lehmann-Romano procedure (step-up)'.
`Data$Raw.pvalues`	all observed raw p-values.
`Data$Weights`	the weights for the raw p-values.
`Data$FDP.threshold`	FDP threshold `alpha`.
`Data$Exceedance.probability`	probability `zeta` of FDP exceeding `alpha`; thus, FDP is being controlled at level `alpha` with confidence 1 - `zeta`.
`Data$Weighting`	character string describing the weighting method.
`Data$Data.name`	the respective variable name(s) of the input data.

References

Döhler, S. & Roquain, E. (2020). Controlling False Discovery Exceedance for Heterogeneous Tests. *Electronic Journal of Statistics*, *14*(2), pp. 4244-4272. doi:10.1214/20EJS1771

See Also

`kernel`, `FDX`, `continuous.LR()`, `continuous.GR()`, `discrete.LR()`, `discrete.GR()`, `discrete.PB()`, `weighted.LR()`, `weighted.GR()`

Examples

```
# Construction of the p-values and their supports for weighted methods
raw.pvalues.weighted <- c(0.7389727, 0.1882310, 0.1302457, 0.9513677,
                          0.7592122, 0.0100559, 0.0000027, 0.1651034)
weights <- c(0.7947122, 1.2633867, 2.8097858, 2.2112801,
             2.3878654, 1.2389620, 2.3878654, 0.7947122)

# arithmetic-weighted Poisson-binomial procedure without critical values
wPB.AM.fast <- wPB.AM(raw.pvalues.weighted, weights)
summary(wPB.AM.fast)

# arithmetic-weighted Poisson-binomial procedure with critical values
wPB.AM.crit <- wPB.AM(raw.pvalues.weighted, weights, critical.values = TRUE)
summary(wPB.AM.crit)

# geometric-weighted Poisson-binomial procedure without critical values
wPB.GM.fast <- wPB.GM(raw.pvalues.weighted, weights)
summary(wPB.GM.fast)

# geometric-weighted Poisson-binomial procedure with critical values
wPB.GM.crit <- wPB.GM(raw.pvalues.weighted, weights, critical.values = TRUE)
summary(wPB.GM.crit)
```

Index

continuous.GR, 4
continuous.GR(), *9*, *17*, *20*, *30*, *53*, *56*, *59*
continuous.LR, 6
continuous.LR(), *6*, *17*, *20*, *30*, *53*, *56*, *59*

DGR, 9
DGR(), *2*, *24*, *27*, *38*, *41*, *44*
direct.discrete, 12
direct.discrete.*(), *31*
discrete.GR, 15
discrete.GR(), *2, 3*, *6*, *9*, *12*, *20*, *24*, *27*, *30*, *36*, *38*, *41*, *44*, *53*, *56*, *59*
discrete.LR, 18
discrete.LR(), *2, 3*, *6*, *9*, *12*, *17*, *22*, *24*, *27*, *30*, *38, 39*, *41*, *44*, *53*, *56*, *59*
discrete.PB *(*DPB.DiscreteTestResults*)*, 28
discrete.PB(), *2, 3*, *6*, *9*, *12*, *17*, *20*, *24, 25*, *27*, *38*, *41, 42*, *44*, *53*, *56*, *59*
DiscreteFDR::fisher.pvalues.support(), *3*, *32*
DiscreteTestResults, *4*, *7*, *10*, *16*, *19*, *22*, *25*, *29*, *36*, *39*, *42*, *51*, *54*, *57*
DiscreteTests, *4*, *7*, *10*, *12, 13*, *16*, *19*, *22*, *25*, *29*, *36*, *39*, *42*, *51*, *54*, *57*
DLR, 22
DLR(), *2*, *12*, *27*, *38*, *41*, *44*
DPB, 25
DPB(), *2*, *12*, *24*, *38*, *41*, *44*
DPB.DiscreteTestResults, 28

fast.Discrete, 31
fast.Discrete.GR(), *3*
fast.Discrete.LR(), *3*
fast.Discrete.PB(), *3*
FDX, *9*, *17*, *20*, *30*, *53*, *56*, *59*
FDX *(*FDX-package*)*, 2
FDX-package, 2
fisher.test(), *32*

GR *(*continuous.GR*)*, 4
graphics::hist(), *35*
graphics::plot.default(), *46*
graphics::plot.histogram(), *35*

hist.FDX, 34

kernel, *6*, *30*, *53*, *56*, *59*
kernel(), *9*

LR *(*continuous.LR*)*, 6

NDGR, 36
NDGR(), *2*, *12*, *24*, *27*, *44*
NDLR, 39
NDLR(), *2*, *12*, *24*, *27*, *38*, *41*, *44*
NDPB, 42
NDPB(), *2*, *12*, *24*, *27*, *38*, *41*
NGR *(*continuous.GR*)*, 4
NLR *(*continuous.LR*)*, 6

plot.FDX, 45
print.FDX, 47
print.summary.FDX *(*summary.FDX*)*, 49

rejection.path, 48

stats::plot.stepfun(), *49*
summary.FDX, 49

weighted.GR, 51
weighted.GR(), *3*, *6*, *9*, *17*, *20*, *30*, *56*, *59*
weighted.LR, 54
weighted.LR(), *3*, *6*, *9*, *17*, *20*, *30*, *53*, *59*
weighted.PB, 57
weighted.PB(), *3*, *6*, *9*, *17*, *20*, *30*, *53*, *56*
wGR.AM *(*weighted.GR*)*, 51
wGR.AM(), *3*
wGR.GM *(*weighted.GR*)*, 51
wLR.AM *(*weighted.LR*)*, 54
wLR.AM(), *3*

wLR.GM *(*weighted.LR*)*, 54
wLR.GM(), *3*
wPB.AM *(*weighted.PB*)*, 57
wPB.AM(), *3*
wPB.GM *(*weighted.PB*)*, 57
wPB.GM(), *3*