Package: FDX (via r-universe)

September 11, 2024

Type Package

Title False Discovery Exceedance Controlling Multiple Testing Procedures

Version 1.0.8-240910

Date 2024-09-10

Description Multiple testing procedures for heterogeneous and discrete tests as described in Döhler and Roquain (2019) <arXiv:1912.04607v1>. 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), methods, PoissonBinomial (>= 1.2.0), pracma, DiscreteFDR (>= 2.0.0), checkmate, DiscreteTests

LinkingTo Rcpp, RcppArmadillo, PoissonBinomial

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

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

RoxygenNote 7.3.2

Roxygen list(markdown = TRUE)

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

RemoteUrl https://github.com/disohda/fdx

RemoteRef HEAD

RemoteSha bf0305eeabfaed13ae81c91871ebd36e7512dbb3

Contents

`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.

References

S. Döhler and E. Roquain (2019). Controlling False Discovery Exceedance for Heterogeneous Tests. arXiv:1912.04607v1.

Author(s)

Maintainer: Florian Junge <`florian.junge@h-da.de`>

Authors:

- Sebastian Döhler

Other contributors:

- Etienne Roquain [contributor]

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
)

continuous.GR2(
  raw.pvalues,
  alpha = 0.05,
  zeta = 0.5,
  adaptive = TRUE,
  critical.values = FALSE
)
```

```
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

`alpha`	the target FDP, a number strictly between 0 and 1. For `*.fast` kernels, it is only necessary, if `stepUp = TRUE`.
`zeta`	the target probability of not exceeding the desired FDP, a number strictly between 0 and 1. If `zeta=NULL` (the default), then `zeta` is chosen equal to `alpha`.
`adaptive`	a boolean specifying whether to conduct an adaptive procedure or not.
`critical.values`	a boolean. If `TRUE`, critical constants are computed and returned (this is computationally intensive).
`raw.pvalues`	vector of the raw observed p-values, as provided by the end user and before matching with their nearest neighbor in the CDFs supports.

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 hypotheses.
`Num.rejected`	Number of rejections.
`Adjusted`	Adjusted p-values (only for step-down direction).
`Critical.values`	Critical values (if requested).
`Method`	A character string describing the used algorithm, e.g. 'Discrete Lehmann-Romano procedure (step-up)'.

`FDP.threshold`	FDP threshold `alpha`.
`Exceedance.probability`	Probability `zeta` of FDP exceeding `alpha`; thus, FDP is being controlled at level `alpha` with confidence `1 - zeta`.
`Adaptive`	A boolean specifying whether an adaptive procedure was conducted or not.
`Data$raw.pvalues`	The values of `raw.pvalues`.
`Data$data.name`	The respective variable names of `raw.pvalues` and `pCDFlist`.

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 (fisher.pvalues.support
# is from 'DiscreteFDR' package!)
df.formatted <- fisher.pvalues.support(counts = df, input = "noassoc")
raw.pvalues <- df.formatted$raw
pCDFlist <- df.formatted$support

GR.fast <- GR(raw.pvalues)
summary(GR.fast)

GR.crit <- GR(raw.pvalues, critical.values = TRUE)
summary(GR.crit)

NGR.fast <- NGR(raw.pvalues)
summary(NGR.fast)

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
)

continuous.LR2(
  raw.pvalues,
  alpha = 0.05,
  zeta = 0.5,
  adaptive = TRUE,
  critical.values = FALSE
)

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

`alpha`	the target FDP, a number strictly between 0 and 1. For `*.fast` kernels, it is only necessary, if `stepUp = TRUE`.
`zeta`	the target probability of not exceeding the desired FDP, a number strictly between 0 and 1. If `zeta=NULL` (the default), then `zeta` is chosen equal to `alpha`.
`adaptive`	a boolean specifying whether to conduct an adaptive procedure or not.

`critical.values`	a boolean. If `TRUE`, critical constants are computed and returned (this is computationally intensive).
`raw.pvalues`	vector of the raw observed p-values, as provided by the end user and before matching with their nearest neighbor in the CDFs supports.

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 hypotheses.
`Num.rejected`	Number of rejections.
`Adjusted`	Adjusted p-values (only for step-down direction).
`Critical.values`	Critical values (if requested).
`Method`	A character string describing the used algorithm, e.g. 'Discrete Lehmann-Romano procedure (step-up)'.
`FDP.threshold`	FDP threshold `alpha`.
`Exceedance.probability`	Probability `zeta` of FDP exceeding `alpha`; thus, FDP is being controlled at level `alpha` with confidence `1 - zeta`.
`Adaptive`	A boolean specifying whether an adaptive procedure was conducted or not.
`Data$raw.pvalues`	The values of `raw.pvalues`.
`Data$data.name`	The respective variable names of `raw.pvalues` and `pCDFlist`.

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 (fisher.pvalues.support
# is from 'DiscreteFDR' package!)
df.formatted <- fisher.pvalues.support(counts = df, input = "noassoc")
raw.pvalues <- df.formatted$raw
pCDFlist <- df.formatted$support

LR.fast <- LR(raw.pvalues)
summary(LR.fast)

LR.crit <- LR(raw.pvalues, critical.values = TRUE)
summary(LR.crit)

NLR.fast <- NLR(raw.pvalues)
summary(NLR.fast)

NLR.crit <- NLR(raw.pvalues, critical.values = TRUE)
summary(NLR.crit)
```

`direct.Discrete` *Direct Application of Multiple Testing Procedures to Dataset*

Description

Apply the [HSU], [HSD], [AHSU] or [AHSD] procedure, with or without computing the critical constants, to a data set of 2x2 contingency tables using Fisher's exact tests which may have to be transformed before computing p-values.

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

`alpha`	the target FDP, a number strictly between 0 and 1. For `*.fast` kernels, it is only necessary, if `stepUp` = `TRUE`.
`direction`	a character string specifying whether to conduct a step-up (`direction="su"`, the default) or step-down procedure (`direction="sd"`).
`adaptive`	a boolean specifying whether to conduct an adaptive procedure or not.

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 (fisher.pvalues.support
# is from 'DiscreteFDR' package!)
df.formatted <- fisher.pvalues.support(counts = df, input = "noassoc")
raw.pvalues <- df.formatted$raw
pCDFlist <- df.formatted$support
DLR.sd <- direct.discrete.LR(df, "fisher")
summary(DLR.sd)
```

```
NDLR.su <- direct.discrete.LR(df, "fisher", direction = "su", adaptive = FALSE)
NDLR.su$Adjusted
summary(NDLR.su)

ADBH.su <- direct.discrete.BH(df, "fisher", direction = "su", adaptive = TRUE)
summary(ADBH.su)

ADBH.sd <- direct.discrete.BH(df, "fisher", direction = "sd", adaptive = TRUE)
ADBH.sd$Adjusted
summary(ADBH.sd)
```

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,
  ...
)
```

```
discrete.GR2(
  raw.pvalues,
  pCDFlist,
  alpha = 0.05,
  zeta = 0.5,
  adaptive = TRUE,
  critical.values = FALSE
)

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,
  ...
)

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

`pCDFlist`	a list of the supports of the CDFs of the p-values. Each support is represented by a vector that must be in increasing order.
`alpha`	the target FDP, a number strictly between 0 and 1. For `*.fast` kernels, it is only necessary, if `stepUp = TRUE`.
`zeta`	the target probability of not exceeding the desired FDP, a number strictly between 0 and 1. If `zeta=NULL` (the default), then `zeta` is chosen equal to `alpha`.
`adaptive`	a boolean specifying whether to conduct an adaptive procedure or not.
`critical.values`	a boolean. If `TRUE`, critical constants are computed and returned (this is computationally intensive).
`raw.pvalues`	vector of the raw observed p-values, as provided by the end user and before matching with their nearest neighbor in the CDFs supports.

Details

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

Value

A `FDX` S3 class object whose elements are:

`Rejected`	Rejected raw p-values.
`Indices`	Indices of rejected hypotheses.
`Num.rejected`	Number of rejections.
`Adjusted`	Adjusted p-values (only for step-down direction).
`Critical.values`	Critical values (if requested).
`Method`	A character string describing the used algorithm, e.g. 'Discrete Lehmann-Romano procedure (step-up)'.
`FDP.threshold`	FDP threshold `alpha`.
`Exceedance.probability`	Probability `zeta` of FDP exceeding `alpha`; thus, FDP is being controlled at level `alpha` with confidence `1 - zeta`.
`Adaptive`	A boolean specifying whether an adaptive procedure was conducted or not.

`Data$raw.pvalues`	The values of `raw.pvalues`.
`Data$pCDFlist`	The values of `pCDFlist`.
`Data$data.name`	The respective variable names of `raw.pvalues` and `pCDFlist`.

References

S. Döhler and E. Roquain (2019). Controlling False Discovery Exceedance for Heterogeneous Tests. arXiv:1912.04607v1.

See Also

`kernel`, `FDX`, `continuous.LR()`, `continuous.GR()`, `discrete.LR()`, `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 (fisher.pvalues.support
# is from 'DiscreteFDR' package!)
df.formatted <- fisher.pvalues.support(counts = df, input = "noassoc")
raw.pvalues <- df.formatted$raw
pCDFlist <- df.formatted$support

DGR.fast <- DGR(raw.pvalues, pCDFlist)
summary(DGR.fast)

DGR.crit <- DGR(raw.pvalues, pCDFlist, critical.values = TRUE)
summary(DGR.crit)

NDGR.fast <- NDGR(raw.pvalues, pCDFlist)
summary(NDGR.fast)

NDGR.crit <- NDGR(raw.pvalues, pCDFlist, 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,
  ...
)

discrete.LR2(
  raw.pvalues,
  pCDFlist,
  alpha = 0.05,
  zeta = 0.5,
  direction = "sd",
  adaptive = TRUE,
  critical.values = FALSE
```

```
)

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,
  ...
)

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

`pCDFlist`	a list of the supports of the CDFs of the p-values. Each support is represented by a vector that must be in increasing order.
`alpha`	the target FDP, a number strictly between 0 and 1. For `*.fast` kernels, it is only necessary, if `stepUp = TRUE`.
`zeta`	the target probability of not exceeding the desired FDP, a number strictly between 0 and 1. If `zeta=NULL` (the default), then `zeta` is chosen equal to `alpha`.
`direction`	a character string specifying whether to conduct a step-up (`direction="su"`, the default) or step-down procedure (`direction="sd"`).
`adaptive`	a boolean specifying whether to conduct an adaptive procedure or not.
`critical.values`	a boolean. If `TRUE`, critical constants are computed and returned (this is computationally intensive).
`raw.pvalues`	vector of the raw observed p-values, as provided by the end user and before matching with their nearest neighbor in the CDFs supports.

Details

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

Value

A `FDX` S3 class object whose elements are:

`Rejected`	Rejected raw p-values.
`Indices`	Indices of rejected hypotheses.
`Num.rejected`	Number of rejections.
`Adjusted`	Adjusted p-values (only for step-down direction).
`Critical.values`	Critical values (if requested).
`Method`	A character string describing the used algorithm, e.g. 'Discrete Lehmann-Romano procedure (step-up)'.
`FDP.threshold`	FDP threshold `alpha`.
`Exceedance.probability`	Probability `zeta` of FDP exceeding `alpha`; thus, FDP is being controlled at level `alpha` with confidence `1 - zeta`.
`Data$raw.pvalues`	The values of `raw.pvalues`.
`Data$pCDFlist`	The values of `pCDFlist`.
`Data$data.name`	The respective variable names of `raw.pvalues` and `pCDFlist`.

References

S. Döhler and E. Roquain (2019). Controlling False Discovery Exceedance for Heterogeneous Tests. arXiv:1912.04607v1.

See Also

kernel, FDX, continuous.LR(), continuous.GR(), 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 (fisher.pvalues.support
# is from 'DiscreteFDR' package!)
df.formatted <- fisher.pvalues.support(counts = df, input = "noassoc")
raw.pvalues <- df.formatted$raw
pCDFlist <- df.formatted$support

DLR.sd.fast <- DLR(raw.pvalues, pCDFlist)
summary(DLR.sd.fast)
DLR.su.fast <- DLR(raw.pvalues, pCDFlist, direction = "su")
summary(DLR.su.fast)

DLR.sd.crit <- DLR(raw.pvalues, pCDFlist, critical.values = TRUE)
summary(DLR.sd.crit)
DLR.su.crit <- DLR(raw.pvalues, pCDFlist, direction = "su", critical.values = TRUE)
summary(DLR.su.crit)

NDLR.sd.fast <- NDLR(raw.pvalues, pCDFlist)
summary(NDLR.sd.fast)
NDLR.su.fast <- NDLR(raw.pvalues, pCDFlist, direction = "su")
summary(NDLR.su.fast)

NDLR.sd.crit <- NDLR(raw.pvalues, pCDFlist, critical.values = TRUE)
summary(NDLR.sd.crit)
NDLR.su.crit <- NDLR(raw.pvalues, pCDFlist, direction = "su", critical.values = TRUE)
summary(NDLR.su.crit)
```

discrete.PB *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

```
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,
  ...
)

discrete.PB2(
  raw.pvalues,
  pCDFlist,
  alpha = 0.05,
  zeta = 0.5,
  adaptive = TRUE,
  critical.values = FALSE,
  exact = TRUE
```

```
)

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,
  ...
)

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

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

`pCDFlist`	a list of the supports of the CDFs of the p-values. Each support is represented by a vector that must be in increasing order.
`alpha`	the target FDP, a number strictly between 0 and 1. For `*.fast` kernels, it is only necessary, if `stepUp = TRUE`.
`zeta`	the target probability of not exceeding the desired FDP, a number strictly between 0 and 1. If `zeta=NULL` (the default), then `zeta` is chosen equal to `alpha`.
`adaptive`	a boolean specifying whether to conduct an adaptive procedure or not.
`critical.values`	a boolean. If `TRUE`, critical constants are computed and returned (this is computationally intensive).
`exact`	a boolean specifying whether to compute the Poisson-Binomial distribution exactly or by a normal approximation.
`raw.pvalues`	vector of the raw observed p-values, as provided by the end user and before matching with their nearest neighbor in the CDFs supports.

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 hypotheses.
`Num.rejected`	Number of rejections.
`Adjusted`	Adjusted p-values (only for step-down direction).
`Critical.values`	Critical values (if requested).
`Method`	A character string describing the used algorithm, e.g. 'Discrete Lehmann-Romano procedure (step-up)'.
`FDP.threshold`	FDP threshold `alpha`.
`Exceedance.probability`	Probability `zeta` of FDP exceeding `alpha`; thus, FDP is being controlled at level `alpha` with confidence `1 - zeta`.
`Data$raw.pvalues`	The values of `raw.pvalues`.
`Data$pCDFlist`	The values of `pCDFlist`.
`Data$data.name`	The respective variable names of `raw.pvalues` and `pCDFlist`.

References

S. Döhler and E. Roquain (2019). Controlling False Discovery Exceedance for Heterogeneous Tests. arXiv:1912.04607v1.

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 (fisher.pvalues.support
# is from 'DiscreteFDR' package!)
df.formatted <- fisher.pvalues.support(counts = df, input = "noassoc")
raw.pvalues <- df.formatted$raw
pCDFlist <- df.formatted$support

DPB.fast <- DPB(raw.pvalues, pCDFlist)
summary(DPB.fast)

DPB.crit <- DPB(raw.pvalues, pCDFlist, critical.values = TRUE)
summary(DPB.crit)

NDPB.fast <- NDPB(raw.pvalues, pCDFlist)
summary(NDPB.fast)

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

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

Description

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.

Usage

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

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

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

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`	the target FDP, a number strictly between 0 and 1. For `*.fast` kernels, it is only necessary, if `stepUp = TRUE`.
`zeta`	the target probability of not exceeding the desired FDP, a number strictly between 0 and 1. If `zeta=NULL` (the default), then `zeta` is chosen equal to `alpha`.
`direction`	a character string specifying whether to conduct a step-up (`direction="su"`, the default) or step-down procedure (`direction="sd"`).
`adaptive`	a boolean specifying whether to conduct an adaptive procedure or not.

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

Value

A `FDX` S3 class object whose elements are:

`Rejected`	Rejected raw p-values.
`Indices`	Indices of rejected hypotheses.
`Num.rejected`	Number of rejections.
`Adjusted`	Adjusted p-values (only for step-down direction).
`Method`	A character string describing the used algorithm, e.g. 'Discrete Lehmann-Romano procedure (step-up)'.
`FDP.threshold`	FDP threshold `alpha`.
`Exceedance.probability`	Probability `zeta` of FDP exceeding `alpha`; thus, FDP is being controlled at level `alpha` with confidence `1 - zeta`.
`Adaptive`	A boolean specifying whether an adaptive procedure was conducted or not.
`Data$raw.pvalues`	The values of `raw.pvalues`.
`Data$data.name`	The respective variable names of `raw.pvalues` and `pCDFlist`.

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

NDLR.sd <- fast.Discrete.LR(counts = df, input = "noassoc", adaptive = FALSE)
NDLR.sd$Adjusted
summary(NDLR.sd)
NDLR.su <- fast.Discrete.LR(counts = df, input = "noassoc", direction = "su", adaptive = FALSE)
summary(NDLR.su)

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

```
NDGR <- fast.Discrete.GR(counts = df, input = "noassoc", adaptive = FALSE)
NDGR$Adjusted
summary(NDGR)

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

NDPB <- fast.Discrete.PB(counts = df, input = "noassoc", adaptive = FALSE)
NDPB$Adjusted
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.
...	further arguments to graphics::hist() or graphics::plot.histogram(), respectively.

Details

If x contains results of a weighted approach, a histogram of the weighted p-values is constructed. Otherwise, it is constituted by the raw ones.

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 (fisher.pvalues.support
# is from 'DiscreteFDR' package!)
df.formatted <- fisher.pvalues.support(counts = df, input = "noassoc")
raw.pvalues <- df.formatted$raw
pCDFlist <- df.formatted$support

DGR <- DGR(raw.pvalues, pCDFlist)
hist(DGR)
```

`kernel_DLR_fast` *Kernel functions*

Description

Kernel functions transform observed p-values or their support according to [HLR], [PB] and [HGR]. The output is used by `discrete.LR()`, `discrete.PB()` and `discrete.GR()`, respectively. For each procedure, there is a kernel for fast computation and one for calculation of critical values. Kernels followed by ".crit", e.g. `kernel.DGR.crit`, compute and return these critical values, while kernels ending in ".fast" only transform p-values and are therefore faster. The end user should not use these functions directly.

Usage

```
kernel_DLR_fast(
  pCDFlist,
  sorted_pv,
  adaptive = TRUE,
  alpha = 0.05,
  stepUp = FALSE,
  zeta = 0.5,
  support = numeric(),
  pCDFcounts = NULL
)

kernel_DLR_crit(
  pCDFlist,
  support,
  sorted_pv,
  adaptive = TRUE,
  alpha = 0.05,
  zeta = 0.5,
  stepUp = FALSE,
  pCDFcounts = NULL
)

kernel_wLR_fast(qvalues, weights, alpha = 0.05, geom_weighting = FALSE)

kernel_DGR_fast(
  pCDFlist,
  sorted_pv,
  adaptive = TRUE,
  alpha = 0.05,
  pCDFcounts = NULL
)

kernel_DGR_crit(
  pCDFlist,
  support,
  sorted_pv,
  adaptive = TRUE,
  alpha = 0.05,
  zeta = 0.5,
  pCDFcounts = NULL
)

kernel_wGR_fast(qvalues, weights, alpha = 0.05, geom_weighting = FALSE)

kernel_DPB_fast(
  pCDFlist,
  sorted_pv,
  adaptive = TRUE,
  alpha = 0.05,
  exact = TRUE,
  pCDFcounts = NULL
)

kernel_DPB_crit(
  pCDFlist,
  support,
  sorted_pv,
  adaptive = TRUE,
  alpha = 0.05,
  zeta = 0.5,
  exact = TRUE,
  pCDFcounts = NULL
```

```
)

kernel_wPB_fast(
  qvalues,
  weights,
  alpha = 0.05,
  geom_weighting = FALSE,
  exact = TRUE
)
```

Arguments

`pCDFlist`	a list of the supports of the CDFs of the p-values. Each support is represented by a vector that must be in increasing order.
`sorted_pv`	a vector of observed p-values, in increasing order.
`adaptive`	a boolean specifying whether to conduct an adaptive procedure or not.
`alpha`	the target FDP, a number strictly between 0 and 1. For `*.fast` kernels, it is only necessary, if `stepUp` = `TRUE`.
`stepUp`	a numeric vector. Identical to `pvalues` for a step-down procedure. Equals `c.m` for a step-up procedure.
`zeta`	the target probability of not exceeding the desired FDP, a number strictly between 0 and 1. If `zeta=NULL` (the default), then `zeta` is chosen equal to `alpha`.
`support`	a numeric vector. Contains all values of the p-values supports. Ignored, if `stepUp` = `FALSE`. Must be sorted in increasing order!
`qvalues`	a numeric vector. Contains weighted raw p-values.
`weights`	a numeric vector. Contains the weights of the p-values.
`geom_weighting`	a boolean specifying whether to conduct geometric (`TRUE`) or arithmetic (`FALSE`) weighting.
`exact`	a boolean specifying whether to compute the Poisson-Binomial distribution exactly or by a normal approximation.
`pvalues`	a numeric vector. Contains all values of the p-values supports if we search for the critical constants. If not, contains only the observed p-values. Must be sorted in increasing order!

Value

For ".fast" kernels, a vector of transformed p-values is returned; ".crit" kernels return a list object with critical constants (`$crit.consts`) and transformed p-values (`$pval.transf`).

See Also

`FDX`, `discrete.LR()` `discrete.GR()`, `discrete.PB()`, `weighted.LR()`, `weighted.GR()`, `discrete.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 (fisher.pvalues.support
# is from 'DiscreteFDR' package!)
df.formatted <- fisher.pvalues.support(counts = df, input = "noassoc")
raw.pvalues <- df.formatted$raw
pCDFlist <- df.formatted$support

alpha <- 0.05

# If not searching for critical constants, we use only the observed p-values
sorted.pvals <- sort(raw.pvalues)
y.DLR.fast <- kernel_DLR_fast(pCDFlist, sorted.pvals, TRUE)
y.NDGR.fast <- kernel_DGR_fast(pCDFlist, sorted.pvals, FALSE)$pval.transf
# transformed values
y.DLR.fast
y.NDGR.fast

# compute support
pv.list <- sort(unique(unlist(pCDFlist)))
y.DGR.crit <- kernel_DGR_crit(pCDFlist, pv.list, sorted.pvals, TRUE)
y.NDPB.crit <- kernel_DPB_crit(pCDFlist, pv.list, sorted.pvals, FALSE)
# critical constants
y.DGR.crit$crit.consts
y.NDPB.crit$crit.consts
# transformed values
y.DGR.crit$pval.transf
y.NDPB.crit$pval.transf
```

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

Description

Plots raw p-values of a `FDX` object and highlights rejected and accepted 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`	a numeric or character vector of length 3 indicating the colors of the 1. rejected p-values 2. accepted p-values 3. critical values (if present).
`pch`	a numeric or character vector of length 3 indicating the point characters of the 1. rejected p-values 2. accepted p-values 3. critical values (if present and `type.crit` is a plot type like `'p'`, `'b'` etc.).
`lwd`	a numeric vector of length 3 indicating the thickness of the points and lines.
`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 (fisher.pvalues.support
# is from 'DiscreteFDR' package!)
```

```
df.formatted <- fisher.pvalues.support(counts = df, input = "noassoc")
raw.pvalues <- df.formatted$raw
pCDFlist <- df.formatted$support

DLR.sd.fast <- DLR(raw.pvalues, pCDFlist)
DLR.sd.crit <- DLR(raw.pvalues, pCDFlist, critical.values = TRUE)
DLR.su.fast <- DLR(raw.pvalues, pCDFlist, direction = "su")
DLR.su.crit <- DLR(raw.pvalues, pCDFlist, direction = "su", critical.values = TRUE)

plot(DLR.su.fast)
plot(DLR.su.crit, xlim = c(1, 5), ylim = c(0, 0.4))
plot(DLR.sd.fast, col = c(2, 4), pch = c(2, 3), lwd = c(2, 2),
     legend = "topleft", xlim = c(1, 5), ylim = c(0, 0.4))
plot(DLR.sd.crit, col = c(2, 4, 1), pch = c(1, 1, 4), lwd = c(1, 1, 2),
     type.crit = 'o', legend = c(1, 0.4), lty = 1, xlim = c(1, 5),
     ylim = c(0, 0.4))
```

print.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 (fisher.pvalues.support
# is from 'DiscreteFDR' package!)
df.formatted <- fisher.pvalues.support(counts = df, input = "noassoc")
raw.pvalues <- df.formatted$raw
pCDFlist <- df.formatted$support

DPB.crit <- DPB(raw.pvalues, pCDFlist, critical.values = TRUE)
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 rejectionsin 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 (fisher.pvalues.support
# is from 'DiscreteFDR' package!)
df.formatted <- fisher.pvalues.support(counts = df, input = "noassoc")
raw.pvalues <- df.formatted$raw
pCDFlist <- df.formatted$support

DLR <- DLR(raw.pvalues, pCDFlist)
NDLR <- NDLR(raw.pvalues, pCDFlist)

rejection.path(DLR, xlim = c(0, 1), ref.show = TRUE, ref.col = "green", ref.lty = 4)
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`	an object of class "`FDX`".
`...`	further arguments passed to or from other methods.
`x`	an object of class "`summary.FDX`".
`max`	numeric or `NULL`, specifying the maximal number of *rows* of the p-value table to be printed. By default, when `NULL`, `getOption("max.print")` is used.

Details

`summary.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 (fisher.pvalues.support
# is from 'DiscreteFDR' package!)
df.formatted <- fisher.pvalues.support(counts = df, input = "noassoc")
raw.pvalues <- df.formatted$raw
pCDFlist <- df.formatted$support
```

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

weighted.GR2(
  raw.pvalues,
  weights,
  alpha = 0.05,
  zeta = 0.5,
  weighting.method = "AM",
  critical.values = FALSE
)

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

`weights`	a numeric vector. Contains the weights of the p-values.
`alpha`	the target FDP, a number strictly between 0 and 1. For `*.fast` kernels, it is only necessary, if `stepUp = TRUE`.
`zeta`	the target probability of not exceeding the desired FDP, a number strictly between 0 and 1. If `zeta=NULL` (the default), then `zeta` is chosen equal to `alpha`.
`weighting.method`	a character string specifying whether to conduct arithmetic (`direction="AM"`, the default) or geometric weighting (`direction="GM"`) of p-values.
`critical.values`	a boolean. If `TRUE`, critical constants are computed and returned (this is computationally intensive).
`raw.pvalues`	vector of the raw observed p-values, as provided by the end user and before matching with their nearest neighbor in the CDFs supports.

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 hypotheses.
`Num.rejected`	Number of rejections.
`Adjusted`	Adjusted p-values (only for step-down direction).
`Weighted`	Weighted p-values.
`Critical.values`	Critical values (if requested).
`Method`	A character string describing the used algorithm, e.g. 'Discrete Lehmann-Romano procedure (step-up)'.
`FDP.threshold`	FDP threshold `alpha`.
`Exceedance.probability`	Probability `zeta` of FDP exceeding `alpha`; thus, FDP is being controlled at level `alpha` with confidence `1 - zeta`.
`Weighting`	A character string describing the weighting method.

`Data$raw.pvalues`	The values of `raw.pvalues`.
`Data$weights`	The values of `weights`.
`Data$data.name`	The respective variable names of `raw.pvalues` and `pCDFlist`.

References

S. Döhler and E. Roquain (2019). Controlling False Discovery Exceedance for Heterogeneous Tests. arXiv:1912.04607v1.

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)

wGR.AM.fast <- wGR.AM(raw.pvalues.weighted, weights)
summary(wGR.AM.fast)

wGR.AM.crit <- wGR.AM(raw.pvalues.weighted, weights, critical.values = TRUE)
summary(wGR.AM.crit)

wGR.GM.fast <- wGR.GM(raw.pvalues.weighted, weights)
summary(wGR.GM.fast)

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
)

weighted.LR2(
  raw.pvalues,
  weights,
  alpha = 0.05,
  zeta = 0.5,
  weighting.method = "AM",
  critical.values = FALSE
)

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

`weights`	a numeric vector. Contains the weights of the p-values.
`alpha`	the target FDP, a number strictly between 0 and 1. For `*.fast` kernels, it is only necessary, if `stepUp = TRUE`.
`zeta`	the target probability of not exceeding the desired FDP, a number strictly between 0 and 1. If `zeta=NULL` (the default), then `zeta` is chosen equal to `alpha`.
`weighting.method`	a character string specifying whether to conduct arithmetic (`direction="AM"`, the default) or geometric weighting (`direction="GM"`) of p-values.

`critical.values`	a boolean. If `TRUE`, critical constants are computed and returned (this is computationally intensive).
`raw.pvalues`	vector of the raw observed p-values, as provided by the end user and before matching with their nearest neighbor in the CDFs supports.

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 hypotheses.
`Num.rejected`	Number of rejections.
`Adjusted`	Adjusted p-values (only for step-down direction).
`Weighted`	Weighted p-values.
`Critical.values`	Critical values (if requested).
`Method`	A character string describing the used algorithm, e.g. 'Discrete Lehmann-Romano procedure (step-up)'.
`FDP.threshold`	FDP threshold `alpha`.
`Exceedance.probability`	Probability `zeta` of FDP exceeding `alpha`; thus, FDP is being controlled at level `alpha` with confidence `1 - zeta`.
`Weighting`	A character string describing the weighting method.
`Data$raw.pvalues`	The values of `raw.pvalues`.
`Data$weights`	The values of `weights`.
`Data$data.name`	The respective variable names of `raw.pvalues` and `pCDFlist`.

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)
```

```
wLR.AM.fast <- wLR.AM(raw.pvalues.weighted, weights)
summary(wLR.AM.fast)

wLR.AM.crit <- wLR.AM(raw.pvalues.weighted, weights, critical.values = TRUE)
summary(wLR.AM.crit)

wLR.GM.fast <- wLR.GM(raw.pvalues.weighted, weights)
summary(wLR.GM.fast)

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
)

weighted.PB2(
  raw.pvalues,
  weights,
  alpha = 0.05,
  zeta = 0.05,
  weighting.method = "AM",
  critical.values = FALSE,
  exact = TRUE
)

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

`weights`	a numeric vector. Contains the weights of the p-values.
`alpha`	the target FDP, a number strictly between 0 and 1. For `*.fast` kernels, it is only necessary, if `stepUp = TRUE`.
`zeta`	the target probability of not exceeding the desired FDP, a number strictly between 0 and 1. If `zeta=NULL` (the default), then `zeta` is chosen equal to `alpha`.
`weighting.method`	a character string specifying whether to conduct arithmetic (`direction="AM"`, the default) or geometric weighting (`direction="GM"`) of p-values.
`critical.values`	a boolean. If `TRUE`, critical constants are computed and returned (this is computationally intensive).
`exact`	a boolean specifying whether to compute the Poisson-Binomial distribution exactly or by a normal approximation.
`raw.pvalues`	vector of the raw observed p-values, as provided by the end user and before matching with their nearest neighbor in the CDFs supports.

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 hypotheses.

`Num.rejected`	Number of rejections.
`Adjusted`	Adjusted p-values (only for step-down direction).
`Weighted`	Weighted p-values.
`Critical.values`	Critical values (if requested).
`Method`	A character string describing the used algorithm, e.g. 'Discrete Lehmann-Romano procedure (step-up)'.
`FDP.threshold`	FDP threshold `alpha`.
`Exceedance.probability`	Probability `zeta` of FDP exceeding `alpha`; thus, FDP is being controlled at level `alpha` with confidence `1 - zeta`.
`Weighting`	A character string describing the weighting method.
`Data$raw.pvalues`	The values of `raw.pvalues`.
`Data$weights`	The values of `weights`.
`Data$data.name`	The respective variable names of `raw.pvalues` and `pCDFlist`.

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)

wPB.AM.fast <- wPB.AM(raw.pvalues.weighted, weights)
summary(wPB.AM.fast)

wPB.AM.crit <- wPB.AM(raw.pvalues.weighted, weights, critical.values = TRUE)
summary(wPB.AM.crit)

wPB.GM.fast <- wPB.GM(raw.pvalues.weighted, weights)
summary(wPB.GM.fast)

wPB.GM.crit <- wPB.GM(raw.pvalues.weighted, weights, critical.values = TRUE)
summary(wPB.GM.crit)
```

Index