Package 'DiscreteFDR'

September 3, 2021

Type Package

Title Multiple Testing Procedures with Adaptation for Discrete Tests

Version 1.3.6

Date 2021-09-01

Description Multiple testing procedures described in the paper Döhler, Durand and Roquain (2018) ``New FDR bounds for discrete and heterogeneous tests" <doi:10.1214/18-EJS1441>. The main procedures of the paper (HSU and HSD), their adaptive counterparts (AHSU and AHSD), and the HBR variant are available and are coded to take as input a set of observed p-values and their discrete support under the null. A function to compute such p-values and supports for Fisher's exact tests is also provided, along with a wrapper allowing to apply discrete procedures directly from contingency tables.

License GPL-3

Encoding UTF-8

Depends R (>= 3.00)

LazyData true

RoxygenNote 7.1.1

Suggests R.rsp, knitr, rmarkdown, discreteMTP

VignetteBuilder R.rsp, knitr

Imports Rcpp (>= 1.0.1), methods

LinkingTo Rcpp

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

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

NeedsCompilation yes

Author Sebastian Döhler [ctb],
Guillermo Durand [aut, ctb],
Florian Junge [aut, cre],
Etienne Roquain [ctb]

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

Repository CRAN

Date/Publication 2021-09-03 10:30:06 UTC

R topics documented:

`amnesia` *Amnesia and other drug reactions in the MHRA pharmacovigilance spontaneous reporting system*

Description

For each of 2446 drugs in the MHRA database (column 1), the number of cases with amnesia as an adverse event (column 2), and the number of cases with other adverse event for this drug (column 3). In total, 682648 adverse drug reactions were reported, among them 2044 cases of amnesia.

Usage

```
data(amnesia)
```

Format

A data frame with 2446 rows representing drugs with the following 3 columns:

DrugName The name of the drug.

AmnesiaCases Number of the amnesia cases reported for the drug.

OtherAdverseCases Number of other adverse drug reactions reported for the drug.

Details

The data was collected from the Drug Analysis Prints published by the Medicines and Healthcare products Regulatory Agency (MHRA), by Heller & Gur. See references for more details.

References

R. Heller and H. Gur (2011). False discovery rate controlling procedures for discrete tests. arXiv preprint arXiv:1112.4627v2 link.

Source

Drug Analysis Prints on MHRA site

`DBR` *[HBR-λ] procedure*

Description

Apply the [HBR-λ] procedure, with or without computing the critical constants, to a set of p-values and their discrete support.

Usage

```
DBR(
  raw.pvalues,
  pCDFlist,
  alpha = 0.05,
  lambda = NULL,
  ret.crit.consts = FALSE
)
```

Arguments

`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.
`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 FDR level, a number strictly between 0 and 1. For `*.fast` kernels, it is only necessary, if `stepUp = TRUE`.
`lambda`	a number strictly between 0 and 1. If `lambda=NULL` (by default), then `lambda` is chosen equal to `alpha`.
`ret.crit.consts`	a boolean. If `TRUE`, critical constants are computed and returned (this is computationally intensive).

Details

[DBR-lambda] is the discrete version of the [Blanchard-Roquain-lambda] procedure (see References), the authors of the latter suggest to take `lambda = alpha` (see their Proposition 17), which explains the choice of the default value here.

This version: 2019-06-18.

Value

A DiscreteFDR S3 class object whose elements are:

Rejected	Rejected raw p-values
Indices	Indices of rejected hypotheses
Num.rejected	Number of rejections
Adjusted	Adjusted p-values
Critical.constants	Critical constants (if requested)
Method	Character string describing the used algorithm, e.g. 'Discrete Benjamini-Hochberg procedure (step-up)'
Signif.level	Significance level alpha
Lambda	Value of lambda.
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

G. Blanchard and E. Roquain (2009). Adaptive false discovery rate control under independence and dependence. Journal of Machine Learning Research, 10, 2837-2871.

See Also

discrete.BH, DiscreteFDR

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

DBR.fast <- DBR(raw.pvalues, pCDFlist)
summary(DBR.fast)
DBR.crit <- DBR(raw.pvalues, pCDFlist, ret.crit.consts = TRUE)
summary(DBR.crit)
```

discrete.BH *[HSU], [HSD], [AHSU] and [AHSD] procedures*

Description

Apply the [HSU], [HSD], [AHSU] and [AHSD] procedures, with or without computing the critical constants, to a set of p-values and their discrete support.

Usage

```
discrete.BH(
  raw.pvalues,
  pCDFlist,
  alpha = 0.05,
  direction = "su",
  adaptive = FALSE,
  ret.crit.consts = FALSE
)

DBH(
  raw.pvalues,
  pCDFlist,
  alpha = 0.05,
  direction = "su",
  ret.crit.consts = FALSE
)

ADBH(
  raw.pvalues,
  pCDFlist,
  alpha = 0.05,
  direction = "su",
  ret.crit.consts = FALSE
)
```

Arguments

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.
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 FDR level, 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", by default) or step-down procedure (direction="sd").
adaptive	a boolean specifying whether to conduct an adaptive procedure or not.

`ret.crit.consts`
: a boolean. If `TRUE`, critical constants are computed and returned (this is computationally intensive).

Details

`DBH` and `ADBH` are wrapper functions for `discrete.BH`. `DBH` simply passes all its parameters to `discrete.BH` with `adaptive = FALSE`. `ADBH` does the same with `adaptive = TRUE`.

This version: 2019-06-18.

Value

A `DiscreteFDR` 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.constants`
: Critical constants (if requested)

`Method` Character string describing the used algorithm, e.g. 'Discrete Benjamini-Hochberg procedure (step-up)'

`Signif.level` Significance level `alpha`

`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`

See Also

`kernel`, `DiscreteFDR`, `DBR`

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

```
DBH.su.fast <- DBH(raw.pvalues, pCDFlist)
summary(DBH.su.fast)
DBH.sd.fast <- DBH(raw.pvalues, pCDFlist, direction = "sd")
DBH.sd.fast$Adjusted
summary(DBH.sd.fast)

DBH.su.crit <- DBH(raw.pvalues, pCDFlist, ret.crit.consts = TRUE)
summary(DBH.su.crit)
DBH.sd.crit <- DBH(raw.pvalues, pCDFlist, direction = "sd", ret.crit.consts = TRUE)
DBH.sd.crit$Adjusted
summary(DBH.sd.crit)

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

ADBH.su.crit <- ADBH(raw.pvalues, pCDFlist, ret.crit.consts = TRUE)
summary(ADBH.su.crit)
ADBH.sd.crit <- ADBH(raw.pvalues, pCDFlist, direction = "sd", ret.crit.consts = TRUE)
ADBH.sd.crit$Adjusted
summary(ADBH.sd.crit)
```

DiscreteFDR *DiscreteFDR*

Description

This package implements the [HSU], [HSD], [AHSU], [AHSD] and [HBR-lambda] procedures for discrete tests (see References).

Details

The functions are reorganized from the reference paper in the following way. `discrete.BH` (for Discrete Benjamini-Hochberg) implements [HSU], [HSD], [AHSU] and [AHSD] and `DBR` (for Discrete Blanchard-Roquain) implements [HBR-lambda]. `DBH` and `ADBH` are wrappers for `discrete.BH` to access [HSU] and [HSD], as well as [AHSU] and [AHSD] directly. Their main arguments are a vector of raw observed p-values, and a list of the same length, which elements are the discrete supports of the CDFs of the p-values.

The function `fisher.pvalues.support` allows to compute such p-values and support in the framework of a Fisher's exact test of association. It has been inspired by an help page of the package `discreteMTP`.

The function `fast.Discrete` is a wrapper for `fisher.pvalues.support` and `discrete.BH` which allows to apply discrete procedures directly to a data set of contingency tables.

We also provide the `amnesia` data set, used in our examples and in our paper. It is basically the `amnesia` data set of package `discreteMTP`, but slightly reformatted (the difference lies in column 3).

No other function of the package should be used, they are only internal functions called by the main ones.

References

Döhler, S., Durand, G., & Roquain, E. (2018). New FDR bounds for discrete and heterogeneous tests. *Electronic Journal of Statistics*, *12*(1), 1867-1900. doi: 10.1214/18EJS1441

Author(s)

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

Authors:

- Guillermo Durand [contributor]

Other contributors:

- Sebastian Döhler [contributor]
- Etienne Roquain [contributor]

See Also

Useful links:

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

---

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

Description

Apply the [HSU], [HSD], [AHSU] or [AHSD] procedure, without computing the critical constants, to a data set of 2 x 2 contingency tables using Fisher's exact tests.

Usage

```
fast.Discrete(
  counts,
  alternative = "greater",
  input = "noassoc",
  alpha = 0.05,
  direction = "su",
  adaptive = 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 fisher.pvalues.support.
alternative	same argument as in fisher.test. The three possible values are "greater" (default), "two.sided" or "less" and you can specify just the initial letter.
input	the format of the input data frame, see Details of fisher.pvalues.support. The three possible values are "noassoc" (default), "marginal" or "HG2011" and you can specify just the initial letter.
alpha	the target FDR level, 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", by default) or step-down procedure (direction="sd").
adaptive	a boolean specifying whether to conduct an adaptive procedure or not.

Details

This version: 2019-06-18.

Value

A DiscreteFDR 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	Character string describing the used algorithm, e.g. 'Discrete Benjamini-Hochberg procedure (step-up)'
Signif.level	Significance level alpha
Data$raw.pvalues	The values of raw.pvalues
Data$pCDFlist	The values of pCDFlist
Data$data.name	The variable name of the counts dataset

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
```

```
DBH.su <- fast.Discrete(counts = df, input = "noassoc", direction = "su")
summary(DBH.su)

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

ADBH.su <- fast.Discrete(counts = df, input = "noassoc", direction = "su", adaptive = TRUE)
summary(ADBH.su)

ADBH.sd <- fast.Discrete(counts = df, input = "noassoc", direction = "sd", adaptive = TRUE)
ADBH.sd$Adjusted
summary(ADBH.sd)
```

`fisher.pvalues.support` *Computing discrete p-values and their support for binomial and Fisher's exact tests*

Description

Computes discrete raw p-values and their support for binomial test or Fisher's exact test applied to 2 x 2 contingency tables summarizing counts coming from two categorical measurements.

Usage

```
fisher.pvalues.support(counts, alternative = "greater", input = "noassoc")
```

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.
`alternative`	same argument as in `fisher.test`. The three possible values are `"greater"` (default), `"two.sided"` or `"less"` and you can specify just the initial letter.
`input`	the format of the input data frame, see Details. The three possible values are `"noassoc"` (default), `"marginal"` or `"HG2011"` and you can specify just the initial letter.

Details

Assume that each contingency tables compares 2 variables and resumes the counts of association or not with a condition. This can be resumed in the following table:

	Association	No association	Total
Variable 1	X1	Y1	N1
Variable 2	X2	Y2	N2
Total	X1 + X2	Y1 + Y2	N1 + N2

If `input="noassoc"`, `counts` has 4 columns which respectively contain X1, Y1, X2 and Y2. If `input="marginal"`, `counts` has 4 columns which respectively contain X1, N1, X2 and N2.

If `input="HG2011"`, we are in the situation of the `amnesia` data set as in Heller & Gur (2011, see References). Each contingency table is obtained from one variable which is compared to all other variables of the study. That is, counts for "second variable" are replaced by the sum of the counts of the other variables:

	Association	No association	Total
Variable j	Xj	Yj	Nj
Variables !=j	SUM(Xi) - Xj	SUM(Yi) - Yj	SUM(Ni) - Nj
Total	SUM(Xi)	SUM(Yi)	SUM(Ni)

Hence `counts` needs to have only 2 columns which respectively contain Xj and Yj.

`binomial.pvalues.support` and `fisher.pvalues.support` are wrapper functions for `pvalues.support`, setting `test.type = "binomial"` and `test.type = "fisher"`, respectively.

The code for the computation of the p-values of Fisher's exact test is inspired by the example in the help page of `p.discrete.adjust`.

See the Wikipedia article about Fisher's exact test, paragraph Example, for a good depiction of what the code does for each possible value of `alternative`.

The binomial test simply tests for p = 0.5 by using X1 as the test statistic and N1 as the number of trials.

This version: 2021-05-23.

Value

A list of two elements:

`raw`	raw discrete p-values.
`support`	a list of the supports of the CDFs of the p-values. Each support is represented by a vector in increasing order.

References

R. Heller and H. Gur (2011). False discovery rate controlling procedures for discrete tests. arXiv preprint arXiv:1112.4627v2 link.

"Fisher's exact test", Wikipedia, The Free Encyclopedia, accessed 2018-03-20, link.

See Also

`p.discrete.adjust`, `fisher.test`

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

data(amnesia)
#We only keep the first 100 lines to keep the computations fast.
#We also drop the first column to keep only columns of counts, in the Heller & Gur (2011) setting.
amnesia <- amnesia[1:100,2:3]

#Construction of the p-values and their support
amnesia.formatted <- fisher.pvalues.support(counts = amnesia, input = "HG2011")
raw.pvalues <- amnesia.formatted$raw
pCDFlist <- amnesia.formatted$support
```

`hist.DiscreteFDR` *Histogram of Raw p-Values*

Description

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

Usage

```
## S3 method for class 'DiscreteFDR'
hist(x, breaks = "FD", plot = TRUE, ...)
```

Arguments

`x`	an object of class `"DiscreteFDR"`.
`breaks`	as in `hist`; here, the Friedman-Diaconis algorithm(`"FD"`) is used as default.
`plot`	a boolean If `TRUE` (the default), a histogram is plotted, otherwise a list of breaks and counts is returned.
`...`	further arguments to `hist` or `plot.histogram`, respectively.

Details

This method simply calls `hist` and passes the raw p-values of the object.

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

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

---

kernel *Kernel functions*

Description

Kernel functions that transform observed p-values or their support according to [HSU], [HSD], [AHSU], [AHSD] and [HBR-λ]. The output is used by `discrete.BH` or `DBR`, respectively. Additionally, `kernel.DBH.crit`, `kernel.ADBH.crit` and `kernel.DBR.crit` compute and return the critical constants. The end user should not use these functions directly.

Usage

```
kernel_DBH_fast(pCDFlist, pvalues, stepUp = FALSE, alpha = 0.05, support = 0L)

kernel_DBH_crit(pCDFlist, pvalues, sorted_pv, stepUp = FALSE, alpha = 0.05)

kernel_ADBH_fast(pCDFlist, pvalues, stepUp = FALSE, alpha = 0.05, support = 0L)

kernel_ADBH_crit(pCDFlist, pvalues, sorted_pv, stepUp = FALSE, alpha = 0.05)

kernel_DBR_fast(pCDFlist, pvalues, lambda = 0.05)

kernel_DBR_crit(pCDFlist, pvalues, sorted_pv, lambda = 0.05, alpha = 0.05)
```

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.

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!
stepUp	a numeric vector. Identical to pvalues for a step-down procedure. Equals c.m for a step-up procedure.
alpha	the target FDR level, a number strictly between 0 and 1. For *.fast kernels, it is only necessary, if stepUp = TRUE.
support	a numeric vector. Contains all values of the p-values supports. Ignored, if stepUp = FALSE. Must be sorted in increasing order!
sorted_pv	a vector of observed p-values, in increasing order.
lambda	a number strictly between 0 and 1. If lambda=NULL (by default), then lambda is chosen equal to alpha.

Details

When computing critical constants under step-down, that is, when using kernel.DBH.crit, kernel.ADBH.crit or kernel.DBR.crit with stepUp = FALSE (i.e. the step-down case), we still need to get transformed p-values to compute the adjusted p-values.

This version: 2019-11-15.

Value

For kernel.DBH.fast, kernel.ADBH.fast and kernel.DBR.fast, a vector of transformed p-values is returned. kernel.DBH.crit, kernel.ADBH.crit and kernel.DBR.crit return a list object with critical constants ($crit.consts) and transformed p-values ($pval.transf), but if stepUp = FALSE, there are critical values only.

See Also

discrete.BH, DiscreteFDR, DBR

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

alpha <- 0.05
```

```
# Compute the step functions from the supports

# We stay in a step-down context, where pv.numer = pv.denom,
# for the sake of simplicity

# If not searching for critical constants, we use only the observed p-values
sorted.pvals <- sort(raw.pvalues)
y.DBH.fast <- kernel_DBH_fast(pCDFlist, sorted.pvals)
y.ADBH.fast <- kernel_ADBH_fast(pCDFlist, sorted.pvals)
# transformed values
y.DBH.fast
y.ADBH.fast

# compute transformed support
pv.list <- sort(unique(unlist(pCDFlist)))
y.DBH.crit <- kernel_DBH_crit(pCDFlist, pv.list, sorted.pvals)
y.ADBH.crit <- kernel_ADBH_crit(pCDFlist, pv.list, sorted.pvals)
y.DBR.crit <- kernel_DBR_crit(pCDFlist, pv.list, sorted.pvals)
# critical constants
y.DBH.crit$crit.consts
y.ADBH.crit$crit.consts
y.DBR.crit$crit.consts
# The following exist only for step-down direction or DBR
y.DBH.crit$pval.transf
y.ADBH.crit$pval.transf
y.DBR.crit$pval.transf
```

`match.pvals` *Matching raw p-values with supports*

Description

Constructs the observed p-values from the raw observed p-values, by rounding them to their nearest neighbor matching with the supports of their respective CDFs (as in function `p.discrete.adjust` of package `discreteMTP`). The end user should not use it.

Usage

```
match.pvals(pCDFlist, raw.pvalues)
```

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.
`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

Well computed raw p-values should already belong to their respective CDF support. So this function is called at the beginning of DBH, ADBH, and DBR, just in case raw p-values are biased.

For each raw p-value that needs to be rounded, a warning is issued.

This version: 2017-08-16.

Value

A vector where each raw p-value has been replaced by its nearest neighbor.

See Also

discrete.BH, DBR

Examples

```
toyList <- list(c(0.3,0.7,1),c(0.1,0.65,1))
toyRaw1 <- c(0.3,0.65)
match.pvals(toyList,toyRaw1)
toyRaw2 <- c(0.31,0.6)
match.pvals(toyList,toyRaw2)
```

---

plot.DiscreteFDR *Plot Method for* DiscreteFDR *objects*

Description

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

Usage

```
## S3 method for class 'DiscreteFDR'
plot(
  x,
  col = c(2, 4, 1),
  pch = c(1, 1, 1),
  lwd = c(1, 1, 1),
  type.crit = "b",
  legend = NULL,
  ...
)
```

Arguments

x	an object of class "DiscreteFDR".
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	1-character string giving the type of plot desired for the critical values (e.g.: 'p', 'l' etc; see plot).
legend	if NULL, no legend is plotted; otherwise expecting a character string like "topleft" etc. or a numeric vector of two elements indicating (x, y) coordinates.
...	further arguments to plot.default.

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

DBH.su.fast <- DBH(raw.pvalues, pCDFlist)
DBH.su.crit <- DBH(raw.pvalues, pCDFlist, ret.crit.consts = TRUE)
DBH.sd.fast <- DBH(raw.pvalues, pCDFlist, direction = "sd")
DBH.sd.crit <- DBH(raw.pvalues, pCDFlist, direction = "sd", ret.crit.consts = TRUE)

plot(DBH.sd.fast)
plot(DBH.sd.crit, xlim = c(1, 5), ylim = c(0, 0.4))
plot(DBH.su.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(DBH.su.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.DiscreteFDR` *Printing DiscreteFDR results*

Description

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

Usage

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

Arguments

`x`	an object of class "`DiscreteFDR`".
`...`	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 support
df.formatted <- fisher.pvalues.support(counts = df, input = "noassoc")
raw.pvalues <- df.formatted$raw
pCDFlist <- df.formatted$support

DBH.su.crit <- DBH(raw.pvalues, pCDFlist, direction = "su", ret.crit.consts = TRUE)
print(DBH.su.crit)
```

summary.DiscreteFDR *Summarizing Discrete FDR Results*

Description

summary method for class "DiscreteFDR"

Usage

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

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

Arguments

object	an object of class "DiscreteFDR".
...	further arguments passed to or from other methods.
x	an object of class "summary.DiscreteFDR".
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.DiscreteFDR objects include all data of an DiscreteFDR 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.DiscreteFDR simply prints the same output as print.DiscreteFDR, but also prints the p-value table.

Value

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

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

DBH.sd.crit <- DBH(raw.pvalues, pCDFlist, direction = "sd", ret.crit.consts = TRUE)
summary(DBH.sd.crit)
```

Index