

# Package ‘DiscreteFDR’

July 8, 2024

**Type** Package

**Title** FDR Based Multiple Testing Procedures with Adaptation for Discrete Tests

**Version** 2.0.0

**Date** 2024-07-08

**Description** Implementations of the multiple testing procedures for discrete tests described in the paper Döhler, Durand and Roquain (2018) ‘‘New FDR bounds for discrete and heterogeneous tests’’ <[doi:10.1214/18-EJS1441](https://doi.org/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 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 a wrapper allowing to apply discrete procedures directly to data.

**License** GPL-3

**Language** en-US

**LazyData** true

**Encoding** UTF-8

**Depends** R (>= 3.00)

**RoxygenNote** 7.3.2

**Suggests** rmarkdown, knitr, R.rsp, kableExtra

**VignetteBuilder** knitr, R.rsp

**Imports** Rcpp (>= 1.0.12), DiscreteTests, lifecycle, checkmate, DiscreteDatasets

**LinkingTo** Rcpp, RcppArmadillo

**URL** <https://github.com/DISOhda/DiscreteFDR>

**BugReports** <https://github.com/DISOhda/DiscreteFDR/issues>

**NeedsCompilation** yes

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

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

**Repository** CRAN

**Date/Publication** 2024-07-08 09:50:02 UTC

## Contents

ADBH . . . . .	2
amnesia . . . . .	5
DBH . . . . .	6
DBR . . . . .	9
direct.discrete.BH . . . . .	12
discrete.BH . . . . .	13
DiscreteFDR . . . . .	17
fast.Discrete . . . . .	18
fisher.pvalues.support . . . . .	20
generate.pvalues . . . . .	22
hist.DiscreteFDR . . . . .	24
kernel . . . . .	25
match.pvals . . . . .	28
plot.DiscreteFDR . . . . .	29
print.DiscreteFDR . . . . .	31
summary.DiscreteFDR . . . . .	32
<b>Index</b>	<b>34</b>

---

ADBH	<i>Wrapper Functions for the Adaptive Discrete Benjamini-Hochberg Procedure</i>
------	---

---

## Description

ADBH() is a wrapper function of `discrete.BH()` for computing [AHSU] and [AHSD], which are more powerful than [HSU] and [HSD], respectively. It simply passes its arguments to `discrete.BH()` with fixed `adaptive = TRUE` and is computationally more demanding than `DBH()`.

## Usage

```
ADBH(test.results, ...)

## Default S3 method:
ADBH(
```

```

    test.results,
    pCDFlist,
    alpha = 0.05,
    direction = "su",
    ret.crit.consts = FALSE,
    select.threshold = 1,
    pCDFlist.indices = NULL,
    ...
)

## S3 method for class 'DiscreteTestResults'
ADBH(
  test.results,
  alpha = 0.05,
  direction = "su",
  ret.crit.consts = FALSE,
  select.threshold = 1,
  ...
)

```

### Arguments

<code>test.results</code>	either a numeric vector with p-values or an R6 object of class <a href="#">DiscreteTestResults</a> from package <a href="#">DiscreteTests</a> for which a discrete FDR procedure is to be performed.
<code>...</code>	further arguments to be passed to or from other methods. They are ignored here.
<code>pCDFlist</code>	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.
<code>alpha</code>	single real number strictly between 0 and 1 indicating the target FDR level.
<code>direction</code>	single character string specifying whether to perform a step-up ("su"; the default) or step-down procedure ("sd").
<code>ret.crit.consts</code>	single boolean specifying whether critical constants are to be computed.
<code>select.threshold</code>	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 <code>threshold = 1</code> (the default), all raw p-values are selected.
<code>pCDFlist.indices</code>	list of numeric vectors containing the test indices that indicate to which raw p-value each <b>unique</b> support in <code>pCDFlist</code> belongs; ignored if the lengths of <code>test.results</code> and <code>pCDFlist</code> are equal.

### Details

Computing critical constants (`ret.crit.consts = 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 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 values (only exists if computations were performed with <code>ret.crit.consts = TRUE</code> ).
Select\$Threshold	p-value selection threshold (only exists if <code>threshold &lt; 1</code> ).
Select\$Effective.Thresholds	results of each p-value CDF evaluated at the selection threshold (only exists if <code>threshold &lt; 1</code> ).
Select\$Pvalues	selected p-values that are $\leq$ selection threshold (only exists if <code>threshold &lt; 1</code> ).
Select\$Indices	indices of p-values $\leq$ selection threshold (only exists if <code>threshold &lt; 1</code> ).
Select\$Scaled	scaled selected p-values (only exists if <code>threshold &lt; 1</code> ).
Select\$Number	number of selected p-values $\leq$ threshold (only exists if <code>threshold &lt; 1</code> ).
Data\$Method	character string describing the used algorithm, e.g. 'Discrete Benjamini-Hochberg procedure (step-up)'
Data\$raw.pvalues	observed p-values.
Data\$pCDFlist	list of the p-value supports.
Data\$FDR.level	FDR level $\alpha$ .
Data\$Data.name	the respective variable names of the input data.

### References

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

### See Also

[discrete.BH\(\)](#), [DBH\(\)](#), [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

# Compute p-values and their supports of Fisher's exact test
test.result <- generate.pvalues(df, "fisher")
raw.pvalues <- test.result$get_pvalues()
pCDFlist <- test.result$get_pvalue_supports()

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

# ADBH (SD) without critical values; using extracted p-values and supports
ADBH.sd.fast <- ADBH(raw.pvalues, pCDFlist, direction = "sd")
summary(ADBH.sd.fast)

# ADBH (SU) with critical values; using test results
ADBH.su.crit <- ADBH(test.result, ret.crit.consts = TRUE)
summary(ADBH.su.crit)

# ADBH (SD) with critical values; using test results
ADBH.sd.crit <- ADBH(test.result, direction = "sd", ret.crit.consts = TRUE)
summary(ADBH.sd.crit)
```

---

amnesia

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

---

## Description

### [Deprecated]

For each of 2,446 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, 682,648 adverse drug reactions were reported, among them 2,044 cases of amnesia.

**Note:** In future versions, this dataset will be removed. Please use the [amnesia](#) dataset from package [DiscreteDatasets](#).

## Usage

```
data(amnesia)
```

**Format**

A data frame with 2,446 rows representing drugs with the following three 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:1112.4627v2](https://arxiv.org/abs/1112.4627v2) (preprint).

**Source**

[Drug Analysis Prints on MHRA site](#)

---

 DBH

---

*Wrapper Functions for the Discrete Benjamini-Hochberg Procedure*


---

**Description**

DBH() is a wrapper function of `discrete.BH()` for computing [HSU] and [HSD]. It simply passes its arguments to `discrete.BH()` with fixed `adaptive = FALSE`.

**Usage**

```
DBH(test.results, ...)

## Default S3 method:
DBH(
  test.results,
  pCDFlist,
  alpha = 0.05,
  direction = "su",
  ret.crit.consts = FALSE,
  select.threshold = 1,
  pCDFlist.indices = NULL,
  ...
)

## S3 method for class 'DiscreteTestResults'
DBH(
```

```

test.results,
alpha = 0.05,
direction = "su",
ret.crit.consts = FALSE,
select.threshold = 1,
...
)

```

### Arguments

<code>test.results</code>	either a numeric vector with p-values or an R6 object of class <code>DiscreteTestResults</code> from package <code>DiscreteTests</code> for which a discrete FDR procedure is to be performed.
<code>...</code>	further arguments to be passed to or from other methods. They are ignored here.
<code>pCDFlist</code>	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.
<code>alpha</code>	single real number strictly between 0 and 1 indicating the target FDR level.
<code>direction</code>	single character string specifying whether to perform a step-up ("su"; the default) or step-down procedure ("sd").
<code>ret.crit.consts</code>	single boolean specifying whether critical constants are to be computed.
<code>select.threshold</code>	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 <code>threshold = 1</code> (the default), all raw p-values are selected.
<code>pCDFlist.indices</code>	list of numeric vectors containing the test indices that indicate to which raw p-value each <b>unique</b> support in <code>pCDFlist</code> belongs; ignored if the lengths of <code>test.results</code> and <code>pCDFlist</code> are equal.

### Details

Computing critical constants (`ret.crit.consts = 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 `DiscreteFDR` S3 class object whose elements are:

<code>Rejected</code>	rejected raw p-values.
<code>Indices</code>	indices of rejected hypotheses.
<code>Num.rejected</code>	number of rejections.
<code>Adjusted</code>	adjusted p-values (only for step-down direction).

`Critical.constants` critical values (only exists if computations were performed with `ret.crit.consts = TRUE`).

`Select$Threshold` p-value selection threshold (only exists if `threshold < 1`).

`Select$Effective.Thresholds` results of each p-value CDF evaluated at the selection threshold (only exists if `threshold < 1`).

`Select$Pvalues` selected p-values that are  $\leq$  selection threshold (only exists if `threshold < 1`).

`Select$Indices` indices of p-values  $\leq$  selection threshold (only exists if `threshold < 1`).

`Select$Scaled` scaled selected p-values (only exists if `threshold < 1`).

`Select$Number` number of selected p-values  $\leq$  threshold (only exists if `threshold < 1`).

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

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

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

`Data$FDR.level` FDR level alpha.

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

## References

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

## See Also

[discrete.BH\(\)](#), [ADBH\(\)](#), [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

# Compute p-values and their supports of Fisher's exact test
test.result <- generate.pvalues(df, "fisher")
raw.pvalues <- test.result$get_pvalues()
pCDFlist <- test.result$get_pvalue_supports()

# DBH (SU) without critical values; using extracted p-values and supports
DBH.su.fast <- DBH(raw.pvalues, pCDFlist)
```



```
summary(DBH.su.fast)

# DBH (SD) without critical values; using extracted p-values and supports
DBH.sd.fast <- DBH(raw.pvalues, pCDFlist, direction = "sd")
summary(DBH.sd.fast)

# DBH (SU) with critical values; using test results
DBH.su.crit <- DBH(test.result, ret.crit.consts = TRUE)
summary(DBH.su.crit)

# DBH (SD) with critical values; using test results
DBH.sd.crit <- DBH(test.result, direction = "sd", ret.crit.consts = TRUE)
summary(DBH.sd.crit)
```

---

DBR

*The Discrete Blanchard-Roquain Procedure*


---

### Description

Applies the [HBR- $\lambda$ ] procedure, with or without computing the critical constants, to a set of p-values and their respective discrete supports.

### Usage

```
DBR(test.results, ...)

## Default S3 method:
DBR(
  test.results,
  pCDFlist,
  alpha = 0.05,
  lambda = NULL,
  ret.crit.consts = FALSE,
  select.threshold = 1,
  pCDFlist.indices = NULL,
  ...
)

## S3 method for class 'DiscreteTestResults'
DBR(
  test.results,
  alpha = 0.05,
  lambda = NULL,
  ret.crit.consts = FALSE,
  select.threshold = 1,
  ...
)
```

**Arguments**

<code>test.results</code>	either a numeric vector with p-values or an R6 object of class <code>DiscreteTestResults</code> from package <code>DiscreteTests</code> for which a discrete FDR procedure is to be performed.
<code>...</code>	further arguments to be passed to or from other methods. They are ignored here.
<code>pCDFlist</code>	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.
<code>alpha</code>	single real number strictly between 0 and 1 indicating the target FDR level.
<code>lambda</code>	real number strictly between 0 and 1 specifying the DBR tuning parameter; if <code>lambda = NULL</code> (the default), <code>lambda</code> is chosen to be equal to <code>alpha</code> .
<code>ret.crit.consts</code>	single boolean specifying whether critical constants are to be computed.
<code>select.threshold</code>	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 <code>threshold = 1</code> (the default), all raw p-values are selected.
<code>pCDFlist.indices</code>	list of numeric vectors containing the test indices that indicate to which raw p-value each <b>unique</b> support in <code>pCDFlist</code> belongs; ignored if the lengths of <code>test.results</code> and <code>pCDFlist</code> are equal.

**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.

Computing critical constants (`ret.crit.consts = 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 `DiscreteFDR` S3 class object whose elements are:

<code>Rejected</code>	rejected raw p-values.
<code>Indices</code>	indices of rejected hypotheses.
<code>Num.rejected</code>	number of rejections.
<code>Adjusted</code>	adjusted p-values.
<code>Critical.constants</code>	critical values (only exists if computations were performed with <code>ret.crit.consts = TRUE</code> ).
<code>Select\$Threshold</code>	p-value selection threshold (only exists if <code>threshold &lt; 1</code> ).

Select\$Effective.Thresholds	results of each p-value CDF evaluated at the selection threshold (only exists if threshold < 1).
Select\$Pvalues	selected p-values that are $\leq$ selection threshold (only exists if threshold < 1).
Select\$Indices	indices of p-values $\leq$ selection threshold (only exists if threshold < 1).
Select\$Scaled	scaled selected p-values (only exists if threshold < 1).
Select\$Number	number of selected p-values $\leq$ threshold (only exists if threshold < 1).
Data\$Method	character string describing the used algorithm, e.g. 'Discrete Benjamini-Hochberg procedure (step-up)'
Data\$raw.pvalues	observed p-values.
Data\$pCDFlist	list of the p-value supports.
Data\$FDR.level	FDR level alpha.
Data\$Data.name	the respective variable names of the input data.
DBR.Tuning	value of the tuning parameter lambda.

## References

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

## See Also

[discrete.BH\(\)](#), [DBH\(\)](#), [ADBH\(\)](#)

## 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

# Compute p-values and their supports of Fisher's exact test
test.result <- generate.pvalues(df, "fisher")
raw.pvalues <- test.result$get_pvalues()
pCDFlist <- test.result$get_pvalue_supports()

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

# DBR with critical values; using test results
DBR.crit <- DBR(test.result, ret.crit.consts = TRUE)
```

```
summary(DBR.crit)
```

---

```
direct.discrete.BH    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.BH(
  dat,
  test.fun,
  test.args = NULL,
  alpha = 0.05,
  direction = "su",
  adaptive = FALSE,
  ret.crit.consts = FALSE,
  select.threshold = 1,
  preprocess.fun = NULL,
  preprocess.args = NULL
)
```

### Arguments

dat	input data; must be suitable for the first parameter of the provided preprocess.fun function or, if preprocess.fun is NULL, for the first parameter of the test.fun function.
test.fun	function <b>from package DiscreteTests</b> , 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 <b>from that package</b> 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 <b>MUST NOT</b> be included!
alpha	single real number strictly between 0 and 1 indicating the target FDR level.
direction	single character string specifying whether to perform a step-up ("su"; the default) or step-down procedure ("sd").
adaptive	single boolean specifying whether to conduct an adaptive procedure or not.
ret.crit.consts	single boolean specifying whether critical constants are to be computed.

`select.threshold`  
 single real number strictly between 0 and 1 indicating the largest raw p-value to be considered, i.e. only p-values below this threshold are considered and the procedures are adjusted in order to take this selection effect into account; if `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 **MUST NOT** be included!

### 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 <- direct.discrete.BH(df, "fisher", direction = "su")
summary(DBH.su)

DBH.sd <- direct.discrete.BH(df, "fisher", direction = "sd")
DBH.sd$Adjusted
summary(DBH.sd)

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.BH

*The Discrete Benjamini-Hochberg Procedure*


---

### Description

Applies the [HSU], [HSD], [AHSU] and [AHSD] procedures at a given FDR level, with or without computing the critical constants, to a set of p-values and their respective discrete supports.

**Usage**

```

discrete.BH(test.results, ...)

## Default S3 method:
discrete.BH(
  test.results,
  pCDFlist,
  alpha = 0.05,
  direction = "su",
  adaptive = FALSE,
  ret.crit.consts = FALSE,
  select.threshold = 1,
  pCDFlist.indices = NULL,
  ...
)

## S3 method for class 'DiscreteTestResults'
discrete.BH(
  test.results,
  alpha = 0.05,
  direction = "su",
  adaptive = FALSE,
  ret.crit.consts = FALSE,
  select.threshold = 1,
  ...
)

```

**Arguments**

<code>test.results</code>	either a numeric vector with p-values or an R6 object of class <code>DiscreteTestResults</code> from package <code>DiscreteTests</code> for which a discrete FDR procedure is to be performed.
<code>...</code>	further arguments to be passed to or from other methods. They are ignored here.
<code>pCDFlist</code>	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.
<code>alpha</code>	single real number strictly between 0 and 1 indicating the target FDR level.
<code>direction</code>	single character string specifying whether to perform a step-up ("su"; the default) or step-down procedure ("sd").
<code>adaptive</code>	single boolean specifying whether to conduct an adaptive procedure or not.
<code>ret.crit.consts</code>	single boolean specifying whether critical constants are to be computed.
<code>select.threshold</code>	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

The adaptive variants [AHSU] and [AHSD], which are executed via `adaptive = TRUE`, are often slightly more powerful than [HSU] and [HSD], respectively. But they are also computationally more demanding.

Computing critical constants (`ret.crit.consts = 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 DiscreteFDR S3 class object whose elements are:

<code>Rejected</code>	rejected raw p-values.
<code>Indices</code>	indices of rejected hypotheses.
<code>Num.rejected</code>	number of rejections.
<code>Adjusted</code>	adjusted p-values (only for step-down direction).
<code>Critical.constants</code>	critical values (only exists if computations were performed with <code>ret.crit.consts = TRUE</code> ).
<code>Select\$Threshold</code>	p-value selection threshold (only exists if <code>threshold &lt; 1</code> ).
<code>Select\$Effective.Thresholds</code>	results of each p-value CDF evaluated at the selection threshold (only exists if <code>threshold &lt; 1</code> ).
<code>Select\$Pvalues</code>	selected p-values that are $\leq$ selection threshold (only exists if <code>threshold &lt; 1</code> ).
<code>Select\$Indices</code>	indices of p-values $\leq$ selection threshold (only exists if <code>threshold &lt; 1</code> ).
<code>Select\$Scaled</code>	scaled selected p-values (only exists if <code>threshold &lt; 1</code> ).
<code>Select\$Number</code>	number of selected p-values $\leq$ threshold (only exists if <code>threshold &lt; 1</code> ).
<code>Data\$Method</code>	character string describing the used algorithm, e.g. 'Discrete Benjamini-Hochberg procedure (step-up)'
<code>Data\$raw.pvalues</code>	observed p-values.
<code>Data\$pCDFlist</code>	list of the p-value supports.
<code>Data\$FDR.level</code>	FDR level $\alpha$ .
<code>Data\$Data.name</code>	the respective variable names of the input data.

## References

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

## See Also

[DBH\(\)](#), [ADBH\(\)](#), [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

# Compute p-values and their supports of Fisher's exact test
test.result <- generate.pvalues(df, "fisher")
raw.pvalues <- test.result$get_pvalues()
pCDFlist <- test.result$get_pvalue_supports()

# DBH (SU) without critical values; using extracted p-values and supports
DBH.su.fast <- discrete.BH(raw.pvalues, pCDFlist)
summary(DBH.su.fast)

# DBH (SD) without critical values; using extracted p-values and supports
DBH.sd.fast <- discrete.BH(raw.pvalues, pCDFlist, direction = "sd")
summary(DBH.sd.fast)

# DBH (SU) with critical values; using test results
DBH.su.crit <- discrete.BH(test.result, ret.crit.consts = TRUE)
summary(DBH.su.crit)

# DBH (SD) with critical values; using test results
DBH.sd.crit <- discrete.BH(test.result, direction = "sd", ret.crit.consts = TRUE)
summary(DBH.sd.crit)

# ADBH (SU) without critical values; using extracted p-values and supports
ADBH.su.fast <- discrete.BH(raw.pvalues, pCDFlist, adaptive = TRUE)
summary(ADBH.su.fast)

# ADBH (SD) without critical values; using extracted p-values and supports
ADBH.sd.fast <- discrete.BH(raw.pvalues, pCDFlist, direction = "sd", adaptive = TRUE)
summary(ADBH.sd.fast)

# ADBH (SU) with critical values; using test results
ADBH.su.crit <- discrete.BH(test.result, adaptive = TRUE, ret.crit.consts = TRUE)
summary(ADBH.su.crit)
```



```
# ADBH (SD) with critical values; using test results
ADBH.sd.crit <- discrete.BH(test.result, direction = "sd", adaptive = TRUE, ret.crit.consts = TRUE)
summary(ADBH.sd.crit)
```

---

DiscreteFDR	<i>FDR-based Multiple Testing Procedures with Adaptation for Discrete Tests</i>
-------------	---

---

## 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], while `DBR()` (for Discrete Blanchard-Roquain) implements [HBR- $\lambda$ ]. `DBH()` and `ADBH()` are wrapper functions for `discrete.BH()` to access [HSU] and [HSD], as well as [AHSU] and [AHSD] directly.

This package is part of a package family to which the `DiscreteDatasets` and `DiscreteTests` packages also belong. The latter allows to compute p-values and their respective supports for various tests. The objects that contain these results can be used directly by the `discrete.BH()`, `DBH()`, `ADBH()` and `DBR()` functions. Alternatively, these functions also accept a vector of raw observed p-values and a list of the respective discrete supports of the CDFs of the p-values.

**Note:** The former function `fisher.pvalues.support()`, which allows to compute such p-values and supports in the framework of a Fisher's exact test, is now deprecated and should not be used anymore. It has been replaced by `generate.pvalues()`.

The same applies for the function `fast.Discrete()`, which is a wrapper for `fisher.pvalues.support()` and `discrete.BH()` and allows to apply discrete procedures directly to a data set of contingency tables and perform data pre-processing before p-values are computed. It is also now deprecated and has been replaced by `direct.discrete.BH()`, but for more flexibility, users may employ pipes, e.g.

```
data |>
DiscreteDatasets::reconstruct_*(<args>) |>
DiscreteTests::*.test.pv(<args>) |>
discrete.BH(<args>).
```

## Author(s)

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

Authors:

- Sebastian Döhler [contributor]
- Guillermo Durand [contributor]

Other contributors:

- Etienne Roquain [contributor]
- Christina Kihn [contributor]

## References

- Döhler, S., Durand, G., & Roquain, E. (2018). New FDR bounds for discrete and heterogeneous tests. *Electronic Journal of Statistics*, 12(1), pp. 1867-1900. doi:10.1214/18EJS1441
- G. Blanchard and E. Roquain (2009). Adaptive false discovery rate control under independence and dependence. *Journal of Machine Learning Research*, 10, pp. 2837-2871.

## See Also

Useful links:

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

---

fast.Discrete

*Fast Application of Discrete Multiple Testing Procedures*

---

## Description

### [Deprecated]

Apply the [HSU], [HSD], [AHSU] or [AHSD] procedure, without computing the critical constants, to a data set of 2x2 contingency tables which may have to be pre-processed in order to have the correct structure for computing p-values using Fisher's exact test.

**Note:** This function is deprecated and will be removed in a future version. Please use `direct.discrete.BH()` with `test.fun = DiscreteTests::fisher.test.pv` and (optional) `preprocess.fun = DiscreteDatasets::reconstruct` or `preprocess.fun = DiscreteDatasets::reconstruct_four` instead. Alternatively, use a pipeline, e.g.

```
data |>
  DiscreteDatasets::reconstruct_*(<args>) |>
  DiscreteTests::*.test.pv(<args>) |>
  discrete.BH(<args>).
```

## Usage

```
fast.Discrete(
  counts,
  alternative = "greater",
  input = "noassoc",
  alpha = 0.05,
  direction = "su",
  adaptive = FALSE,
  select.threshold = 1
)
```

**Arguments**

counts	a data frame of two or four columns and any number of lines; each line representing a 2x2 contingency table to test. The number of columns and what they must contain depend on the value of the input argument (see Details section of <code>fisher.pvalues.support()</code> ).
alternative	same argument as in <code>stats::fisher.test()</code> . The three possible values are "greater" (default), "two.sided" or "less" (may be abbreviated).
input	the format of the input data frame (see Details section of <code>fisher.pvalues.support()</code> ). The three possible values are "noassoc" (default), "marginal" or "HG2011" (may be abbreviated).
alpha	single real number strictly between 0 and 1 indicating the target FDR level.
direction	single character string specifying whether to perform a step-up ("su"; the default) or step-down procedure ("sd").
adaptive	single boolean specifying whether to conduct an adaptive procedure or not.
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.

**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 values (only exists if computations were performed with <code>ret.crit.consts = TRUE</code> ).
Select\$Threshold	p-value selection threshold (only exists if <code>threshold &lt; 1</code> ).
Select\$Effective.Thresholds	results of each p-value CDF evaluated at the selection threshold (only exists if <code>threshold &lt; 1</code> ).
Select\$Pvalues	selected p-values that are $\leq$ selection threshold (only exists if <code>threshold &lt; 1</code> ).
Select\$Indices	indices of p-values $\leq$ selection threshold (only exists if <code>threshold &lt; 1</code> ).
Select\$Scaled	scaled selected p-values (only exists if <code>threshold &lt; 1</code> ).
Select\$Number	number of selected p-values $\leq$ threshold (only exists if <code>threshold &lt; 1</code> ).
Data\$Method	character string describing the used algorithm, e.g. 'Discrete Benjamini-Hochberg procedure (step-up)'

Data\$raw.pvalues observed p-values.  
 Data\$pCDFlist list of the p-value supports.  
 Data\$FDR.level FDR level alpha.  
 Data\$Data.name the respective variable names of the input data.

**See Also**

[fisher.pvalues.support\(\)](#), [discrete.BH\(\)](#)

**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(df, input = "noassoc", direction = "su")
summary(DBH.su)
```

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

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

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

---

fisher.pvalues.support

*Computing Discrete P-Values and Their Supports for Fisher's Exact Test*

---

**Description****[Deprecated]**

Computes discrete raw p-values and their support for Fisher's exact test applied to 2x2 contingency tables summarizing counts coming from two categorical measurements.

**Note:** This function is deprecated and will be removed in a future version. Please use [generate.pvalues\(\)](#) with `test.fun = DiscreteTests::fisher.test.pv` and (optional) `preprocess.fun = DiscreteDatasets::reconstruct`

```

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

```

## Usage

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

## Arguments

**counts** a data frame of two or four columns and any number of lines; each line represents a 2x2 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 `stats::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 two 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	$X_1$	$Y_1$	$N_1$
Variable 2	$X_2$	$Y_2$	$N_2$
Total	$X_1 + X_2$	$Y_1 + Y_2$	$N_1 + N_2$

If `input="noassoc"`, `counts` has four columns which respectively contain,  $X_1$ ,  $Y_1$ ,  $X_2$  and  $Y_2$ . If `input="marginal"`, `counts` has four columns which respectively contain  $X_1$ ,  $N_1$ ,  $X_2$  and  $N_2$ .

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 $j$	$X_j$	$Y_j$	$N_j$
Variables $\neq j$	$\sum_{i \neq j} X_i$	$\sum_{i \neq j} Y_i$	$\sum_{i \neq j} N_i$
Total	$\sum X_i$	$\sum Y_i$	$\sum N_i$

Hence `counts` needs to have only two columns which respectively contain  $X_j$  and  $Y_j$ .

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` of package `discreteMTP`, which is no longer available on CRAN.

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`.

### Value

A list of two elements:

<code>raw</code>	raw discrete p-values.
<code>support</code>	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](https://arxiv.org/abs/1112.4627v2).

"Fisher's exact test", Wikipedia, The Free Encyclopedia, accessed 2018-03-20, [link](#).

### See Also

[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

# Compute p-values and their supports of Fisher's exact test
df.formatted <- fisher.pvalues.support(counts = df, input = "noassoc")
raw.pvalues <- df.formatted$raw
pCDFlist <- df.formatted$support
```

---

generate.pvalues	<i>Generation of P-Values and Their Supports After Data Transformations</i>
------------------	---

---

### Description

Simple wrapper for generating p-values of discrete tests and their supports after pre-processing the input data. The user only has to provide 1.) a function that generates p-values and supports and 2.) an optional function that pre-processes (i.e. transforms) the input data (if necessary) before it can be used for p-value calculations. The respective arguments are provided by named lists.

**Usage**

```
generate.pvalues(
  dat,
  test.fun,
  test.args = NULL,
  preprocess.fun = NULL,
  preprocess.args = NULL
)
```

**Arguments**

dat	input data; must be suitable for the first parameter of the provided preprocess.fun function or, if preprocess.fun is NULL, for the first parameter of the test.fun function.
test.fun	function <b>from package DiscreteTests</b> , 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 <b>from that package</b> 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 <b>MUST NOT</b> be included!
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 <b>MUST NOT</b> be included!

**Value**

A [DiscreteTestResults](#) R6 class object.

**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

# Compute p-values and their supports of Fisher's exact test
test.result <- generate.pvalues(df, "fisher")
raw.pvalues <- test.result$get_pvalues()
pCDFlist <- test.result$get_pvalue_supports()
```

```

# Compute p-values and their supports of Fisher's exact test with pre-processing
df2 <- data.frame(X1, N1, X2, N2)
generate.pvalues(
  dat = df2,
  test.fun = "fisher.test.pv",
  preprocess.fun = function(tab) {
    for(col in c(2, 4)) tab[, col] <- tab[, col] - tab[, col - 1]
    return(tab)
  }
)

# Compute p-values and their supports of a binomial test with pre-processing
generate.pvalues(
  dat = rbind(c(5, 2, 7), c(3, 4, 0)),
  test.fun = "binom.test.pv",
  test.args = list(n = c(9, 8, 11), p = 0.6, alternative = "two.sided"),
  preprocess.fun = colSums
)

```

---

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 <a href="#">hist</a> ; 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 <a href="#">hist</a> or <a href="#">plot.histogram</a> , 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

# Compute p-values and their supports of Fisher's exact test
test.result <- generate.pvalues(df, "fisher")
raw.pvalues <- test.result$get_pvalues()
pCDFlist <- test.result$get_pvalue_supports()

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- $\lambda$ ]. The output is used by [discrete.BH](#) or [DBR](#), respectively. `kernel_DBH_crit`, `kernel_ADBH_crit` and `kernel_DBR_crit` additionally compute and return the critical constants. The end user should not use these functions directly.

**Note:** As of version 2.0, these functions are purely internal functions! As a consequence, they have to be called directly via `:::`, e.g. `DiscreteFDR:::kernel_DBH_fast()`. But users should **not** rely on them, as parameters (including their names, order, etc.) may be changed without notice!

**Usage**

```

kernel_DBH_fast(
  pCDFlist,
  pvalues,
  stepUp = FALSE,
  alpha = 0.05,
  support = numeric(),
  pCDFcounts = NULL
)

```

```

kernel_DBH_crit(
  pCDFlist,
  support,
  sorted_pv,
  stepUp = FALSE,

```

```

    alpha = 0.05,
    pCDFcounts = NULL
)

kernel_ADBH_fast(
  pCDFlist,
  sorted_pv,
  stepUp = FALSE,
  alpha = 0.05,
  support = numeric(),
  pCDFcounts = NULL
)

kernel_ADBH_crit(
  pCDFlist,
  support,
  sorted_pv,
  stepUp = FALSE,
  alpha = 0.05,
  pCDFcounts = NULL
)

kernel_DBR_fast(pCDFlist, sorted_pv, lambda = 0.05, pCDFcounts = NULL)

kernel_DBR_crit(
  pCDFlist,
  support,
  sorted_pv,
  lambda = 0.05,
  alpha = 0.05,
  pCDFcounts = NULL
)

```

### Arguments

<code>pCDFlist</code>	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.
<code>pvalues</code>	numeric vector, sorted in increasing order, that either must contain the entirety of all observable values of the p-value supports (when computing critical constants) or only the sorted raw p-values.
<code>stepUp</code>	boolean specifying whether to conduct the step-up (TRUE) or step-down (FALSE; the default) procedure.
<code>alpha</code>	single real number strictly between 0 and 1 indicating the target FDR level; for *.fast kernels, it is only needed, if <code>stepUp = TRUE</code> .
<code>support</code>	numeric vector, sorted in increasing order, that contains the entirety of all observable values of the p-value supports; for *.fast kernels, it is ignored if <code>stepUp = FALSE</code> .

pCDFcounts	integer vector of counts that indicates to how many p-values each <b>unique</b> p-value distributions belongs.
sorted_pv	numeric vector containing the raw p-values, sorted in increasing order.
lambda	real number strictly between 0 and 1 specifying the DBR tuning parameter.

### 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.

### 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 with critical constants (`$crit.consts`) and transformed p-values (`$pval.transf`), but if `stepUp = FALSE`, there are critical values only.

### See Also

[discrete.BH](#), [fast.Discrete](#), [DBR](#)

### Examples

```
## Not run:
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

# Compute p-values and their supports of Fisher's exact test
test.result <- generate.pvalues(df, "fisher")
raw.pvalues <- test.result$get_pvalues()
pCDFlist <- test.result$get_pvalue_supports()

alpha <- 0.05

# Compute the step functions from the supports

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

```

y.DBR.fast

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

## End(Not run)

```

---

match.pvals

*Matching Raw P-Values with Supports*


---

## Description

### [Deprecated]

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`, which is no longer available on CRAN).

**Note:** In the next version, this is to become an internal function and will have to be called directly via `:::`, i.e. `DiscreteFDR:::match.pvals()`.

## Usage

```
match.pvals(pCDFlist, raw.pvalues, pCDFlist.indices = NULL)
```

## Arguments

<code>pCDFlist</code>	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.
<code>raw.pvalues</code>	numeric vector with raw p-values for which a discrete FDR procedure is to be performed.
<code>pCDFlist.indices</code>	list of numeric vectors containing the test indices that indicate to which raw p-value each <b>unique</b> support in <code>pCDFlist</code> belongs; ignored if the lengths of <code>test.results</code> and <code>pCDFlist</code> are equal.

## Details

Well computed raw p-values should already belong to their respective CDF support. So this function is called at the beginning of `discrete.BH()`, `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.

## Value

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

## See Also

`discrete.BH()`, `DBR()`

## Examples

```
## Not run:
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)

## End(Not run)
```

---

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(20, 20, 17),
  lwd = rep(par())$lwd, 3),
  cex = rep(par())$cex, 3),
  type.crit = "b",
  legend = NULL,
  ...
)
```

**Arguments**

<code>x</code>	object of class <code>DiscreteFDR</code> .
<code>col</code>	numeric or character vector of length 3 indicating the colors of the <ol style="list-style-type: none"> <li>1. rejected p-values</li> <li>2. accepted p-values</li> <li>3. critical values (if present).</li> </ol>
<code>pch</code>	numeric or character vector of length 3 indicating the point characters of the <ol style="list-style-type: none"> <li>1. rejected p-values</li> <li>2. accepted p-values</li> <li>3. critical values (if present and <code>type.crit</code> is a plot type like 'p', 'b' etc.).</li> </ol>
<code>lwd</code>	numeric vector of length 3 indicating the thickness of the points and lines; defaults to current <code>par()\$lwd</code> setting.
<code>cex</code>	numeric vector of length 3 indicating the size of point characters or lines of the <ol style="list-style-type: none"> <li>1. rejected p-values</li> <li>2. accepted p-values</li> <li>3. critical values (if present).</li> </ol> defaults to current <code>par()\$cex</code> setting.
<code>type.crit</code>	1-character string giving the type of plot desired for the critical values (e.g.: 'p', 'l' etc; see <code>plot()</code> ).
<code>legend</code>	if <code>NULL</code> , no legend is plotted; otherwise expecting a character string like "topleft" etc. or a numeric vector of two elements indicating (x, y) coordinates.
<code>...</code>	further arguments to <code>plot.default()</code> .

**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

# Compute p-values and their supports of Fisher's exact test
test.result <- generate.pvalues(df, "fisher")
raw.pvalues <- test.result$get_pvalues()
pCDFlist <- test.result$get_pvalue_supports()

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

# Compute p-values and their supports of Fisher's exact test
test.result <- generate.pvalues(df, "fisher")
raw.pvalues <- test.result$get_pvalues()
pCDFlist <- test.result$get_pvalue_supports()

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 <i>rows</i> of the p-value table to be printed. By default, when NULL, <code>getOption("max.print")</code> is used.

### Details

summary.DiscreteFDR objects contain 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 object, plus

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



**Examples**

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

# Compute p-values and their supports of Fisher's exact test
test.result <- generate.pvalues(df, "fisher")
raw.pvalues <- test.result$get_pvalues()
pCDFlist <- test.result$get_pvalue_supports()

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

# Index

## \* datasets

amnesia, 5

ADBH, 2

ADBH(), 8, 11, 16, 17, 29

amnesia, 5, 5

DBH, 6

DBH(), 2, 4, 11, 16, 17, 29

DBR, 9, 25, 27

DBR(), 4, 8, 16, 17, 29

direct.discrete.BH, 12

direct.discrete.BH(), 17, 18

discrete.BH, 13, 25, 27

discrete.BH(), 2, 4, 6, 8, 11, 17, 20, 29

DiscreteDatasets, 5, 17

DiscreteFDR, 17

DiscreteFDR-package (DiscreteFDR), 17

DiscreteTestResults, 3, 7, 10, 14, 23

DiscreteTests, 3, 7, 10, 12, 14, 17, 23

fast.Discrete, 18, 27

fast.Discrete(), 17

fisher.pvalues.support, 20

fisher.pvalues.support(), 17, 19, 20

fisher.test(), 22

generate.pvalues, 22

generate.pvalues(), 17, 20

hist, 24

hist.DiscreteFDR, 24

kernel, 25

kernel\_ADBH\_crit (kernel), 25

kernel\_ADBH\_fast (kernel), 25

kernel\_DBH\_crit (kernel), 25

kernel\_DBH\_fast (kernel), 25

kernel\_DBR\_crit (kernel), 25

kernel\_DBR\_fast (kernel), 25

match.pvals, 28

plot(), 30

plot.default(), 30

plot.DiscreteFDR, 29

plot.histogram, 24

print.DiscreteFDR, 31

print.summary.DiscreteFDR

(summary.DiscreteFDR), 32

stats::fisher.test(), 19, 21

summary.DiscreteFDR, 32