Package 'onlineFDR'

July 3, 2025

Version 2.16.0

Date 2023-04-12

Title Online error rate control

Description This package allows users to control the false discovery rate (FDR) or familywise error rate (FWER) for online multiple hypothesis testing, where hypotheses arrive in a stream. In this framework, a null hypothesis is rejected based on the evidence against it and on the previous rejection decisions.

URL https://dsrobertson.github.io/onlineFDR/index.html

License GPL-3

RoxygenNote 7.2.1

Encoding UTF-8

Imports stats, Rcpp, progress

Suggests knitr, rmarkdown, testthat, covr

LinkingTo Rcpp, RcppProgress

VignetteBuilder knitr

biocViews MultipleComparison, Software, StatisticalMethod

git_url https://git.bioconductor.org/packages/onlineFDR

git_branch RELEASE_3_21

git_last_commit 62cfe20

git_last_commit_date 2025-04-15

Repository Bioconductor 3.21

Date/Publication 2025-07-02

Author David S. Robertson [aut, cre],
Lathan Liou [aut],
Aaditya Ramdas [aut],
Adel Javanmard [ctb],
Andrea Montanari [ctb],
Jinjin Tian [ctb],
Tijana Zrnic [ctb],
Natasha A. Karp [aut]

Maintainer David S. Robertson <david.robertson@mrc-bsu.cam.ac.uk>

Contents

`onlineFDR-package` *onlineFDR: A package for online error rate control*

Description

The onlineFDR package provides methods to control the false discovery rate (FDR) or familywise error rate (FWER) for online hypothesis testing, where hypotheses arrive in a stream. A null hypothesis is rejected based on the evidence against it and on the previous rejection decisions.

Details

Package:	onlineFDR
Type:	Package
Version:	2.5.1
Date:	2022-08-24
License:	GPL-3

Javanmard and Montanari (2015, 2018) proposed two methods for online FDR control. The first is LORD, which stands for (significance) Levels based On Recent Discovery and is implemented by the function `LORD`. This function also includes the extension to the LORD procedure, called LORD++ (`version='++'`), proposed by Ramdas et al. (2017). Setting `version='discard'` implements a modified version of LORD that can improve the power of the procedure in the presence of conservative nulls by adaptively 'discarding' these p-values, as proposed by Tian and Ramdas (2019a). All these LORD procedures provably control the FDR under independence of the p-values. However, setting `version='dep'` provides a modified version of LORD that is valid for arbitrarily dependent p-values.

The second method is LOND, which stands for (significance) Levels based On Number of Discoveries and is implemented by the function `LOND`. This procedure controls the FDR under independence of the p-values, but the slightly modified version of LOND proposed by Zrnic et al. (2018) also provably controls the FDR under positive dependence (PRDS conditioN). In addition, by specifying `dep = TRUE`, thus function runs a modified version of LOND which is valid for arbitrarily dependent p-values.

Another method for online FDR control proposed by Ramdas et al. (2018) is the `SAFFRON` procedure, which stands for Serial estimate of the Alpha Fraction that is Futiley Rationed On true Null hypotheses. This provides an adaptive algorithm for online FDR control. SAFFRON is related to the Alpha-investing procedure of Foster and Stine (2008), a monotone version of which is implemented by the function `Alpha_investing`. Both these procedure provably control the FDR under independence of the p-values.

Tian and Ramdas (2019) proposed the `ADDIS` algorithm, which stands for an ADaptive algorithm that DIScards conservative nulls. The algorithm compensates for the power loss of SAFFRON with conservative nulls, by including both adaptivity in the fraction of null hypotheses (like SAFFRON) and the conservativeness of nulls (unlike SAFFRON). The ADDIS procedure provably controls the FDR for independent p-values. Tian and Ramdas (2019) also presented a version for an asynchronous testing process, consisting of tests that start and finish at (potentially) random times.

For testing batches of hypotheses, Zrnic et al. (2020) proposed batched online testing algorithms that control the FDR, where the p-values across different batches are independent, and within a batch the p-values are either positively dependent or independent.

Zrnic et al. (2021) generalised LOND, LORD and SAFFRON for asynchronous online testing, where each hypothesis test can itself be a sequential process and the tests can overlap in time. Note though that these algorithms are designed for the control of a modified FDR (mFDR). They are implemented by the functions `LONDstar`, `LORDstar` and `SAFFRONstar`. Zrnic et al. (2021) presented three explicit versions of these algorithms:

1) `version='async'` is for an asynchronous testing process, consisting of tests that start and finish at (potentially) random times. The discretised finish times of the test correspond to the decision times.

2) `version='dep'` is for online testing under local dependence of the p-values. More precisely, for any $t > 0$ we allow the p-value p_t to have arbitrary dependence on the previous L_t p-values. The fixed sequence L_t is referred to as 'lags'.

3) `version='batch'` is for controlling the mFDR in mini-batch testing, where a mini-batch represents a grouping of tests run asynchronously which result in dependent p-values. Once a mini-batch of tests is fully completed, a new one can start, testing hypotheses independent of the previous batch.

Recently, Xu and Ramdas (2021) proposed the supLORD algorithm, which provably controls the false discovery exceedance (FDX) for p-values that are conditionally superuniform under the null. supLORD also controls the supFDR and hence the FDR (even at stopping times).

Finally, Tian and Ramdas (2021) proposed a number of algorithms for online FWER control. The only previously existing procedure for online FWER control is Alpha-spending, which is an online analog of the Bonferroni procedure. This is implemented by the function Alpha_spending, and provides strong FWER control for arbitrarily dependent p-values. A uniformly more powerful method is online_fallback, which again strongly controls the FWER even under arbitrary dependence amongst the p-values. The ADDIS_spending procedure compensates for the power loss of Alpha-spending and online fallback, by including both adapativity in the fraction of null hypotheses and the conservativeness of nulls. This procedure controls the FWER in the strong sense for independent p-values. Tian and Ramdas (2021) also presented a version for handling local dependence, which can be specified by setting dep=TRUE.

Further details on all these procedures can be found in Javanmard and Montanari (2015, 2018), Ramdas et al. (2017, 2018), Robertson and Wason (2018), Tian and Ramdas (2019, 2021), Xu and Ramdas (2021), and Zrnic et al. (2020, 2021).

Author(s)

David S. Robertson (<david.robertson@mrc-bsu.cam.ac.uk>), Lathan Liou, Adel Javanmard, Aaditya Ramdas, Jinjin Tian, Tijana Zrnic, Andrea Montanari and Natasha A. Karp.

References

Aharoni, E. and Rosset, S. (2014). Generalized α-investing: definitions, optimality results and applications to publci databases. *Journal of the Royal Statistical Society (Series B)*, 76(4):771–794.

Foster, D. and Stine R. (2008). α-investing: a procedure for sequential control of expected false discoveries. *Journal of the Royal Statistical Society (Series B)*, 29(4):429-444.

Javanmard, A. and Montanari, A. (2015) On Online Control of False Discovery Rate. *arXiv preprint*, https://arxiv.org/abs/1502.06197.

Javanmard, A. and Montanari, A. (2018) Online Rules for Control of False Discovery Rate and False Discovery Exceedance. *Annals of Statistics*, 46(2):526-554.

Ramdas, A., Yang, F., Wainwright M.J. and Jordan, M.I. (2017). Online control of the false discovery rate with decaying memory. *Advances in Neural Information Processing Systems 30*, 5650-5659.

Ramdas, A., Zrnic, T., Wainwright M.J. and Jordan, M.I. (2018). SAFFRON: an adaptive algorithm for online control of the false discovery rate. *Proceedings of the 35th International Conference in Machine Learning*, 80:4286-4294.

Robertson, D.S. and Wason, J.M.S. (2018). Online control of the false discovery rate in biomedical research. *arXiv preprint*, https://arxiv.org/abs/1809.07292.

Robertson, D.S., Wason, J.M.S. and Ramdas, A. (2022). Online multiple hypothesis testing for reproducible research.*arXiv preprint*, https://arxiv.org/abs/2208.11418.

Robertson, D.S., Wildenhain, J., Javanmard, A. and Karp, N.A. (2019). onlineFDR: an R package to control the false discovery rate for growing data repositories. *Bioinformatics*, 35:4196-4199, https://doi.org/10.1093/bioinformatics/btz191.

Tian, J. and Ramdas, A. (2019). ADDIS: an adaptive discarding algorithm for online FDR control with conservative nulls. *Advances in Neural Information Processing Systems*, 9388-9396.

Tian, J. and Ramdas, A. (2021). Online control of the familywise error rate. *Statistical Methods for Medical Research*, 30(4):976–993.

Xu, Z. and Ramdas, A. (2021). Dynamic Algorithms for Online Multiple Testing. *Annual Conference on Mathematical and Scientific Machine Learning*, PMLR, 145:955-986.

Zrnic, T., Jiang D., Ramdas A. and Jordan M. (2020). The Power of Batching in Multiple Hypothesis Testing. *International Conference on Artificial Intelligence and Statistics*, PMLR, 108:3806-3815.

Zrnic, T., Ramdas, A. and Jordan, M.I. (2021). Asynchronous Online Testing of Multiple Hypotheses. *Journal of Machine Learning Research*, 22:1-33.

`ADDIS` *ADDIS: Adaptive discarding algorithm for online FDR control*

Description

Implements the ADDIS algorithm for online FDR control, where ADDIS stands for an ADaptive algorithm that DIScards conservative nulls, as presented by Tian and Ramdas (2019). The algorithm compensates for the power loss of SAFFRON with conservative nulls, by including both adaptivity in the fraction of null hypotheses (like SAFFRON) and the conservativeness of nulls (unlike SAFFRON).

Usage

```
ADDIS(
  d,
  alpha = 0.05,
  gammai,
  w0,
  lambda = 0.25,
  tau = 0.5,
  async = FALSE,
  random = TRUE,
  display_progress = FALSE,
  date.format = "%Y-%m-%d"
)
```

Arguments

`d`	Either a vector of p-values, or a dataframe with three columns: an identifier ('id'), p-value ('pval'), and decision times ('decision.times').
`alpha`	Overall significance level of the procedure, the default is 0.05.
`gammai`	Optional vector of γ_i. A default is provided with γ_j proportional to $1/j^(1.6)$.
`w0`	Initial 'wealth' of the procedure, defaults to $\alpha/2$.

`lambda`	Optional parameter that sets the threshold for 'candidate' hypotheses. Must be between 0 and tau, defaults to 0.25.
`tau`	Optional threshold for hypotheses to be selected for testing. Must be between 0 and 1, defaults to 0.5.
`async`	Logical. If `TRUE` runs the version for an asynchronous testing process. Defaults to FALSE.
`random`	Logical. If `TRUE` (the default), then the order of the p-values in each batch (i.e. those that have exactly the same date) is randomised. Only needed if async=FALSE.
`display_progress`	Logical. If `TRUE` prints out a progress bar for the algorithm runtime.
`date.format`	Optional string giving the format that is used for dates.

Details

The function takes as its input either a vector of p-values, or a dataframe with three columns. The dataframe requires an identifier ('id'), date ('date') and p-value ('pval'). If the asynchronous version is specified (see below), then the column date should be replaced by the decision times.

Given an overall significance level α, ADDIS depends on constants w_0, λ and τ. Here w_0 represents the initial 'wealth' of the procedure and satisfies $0 \leq w_0 \leq \alpha$. $\tau \in (0, 1)$ represents the threshold for a hypothesis to be selected for testing: p-values greater than τ are implicitly 'discarded' by the procedure. Finally, $\lambda \in [0, \tau)$ sets the threshold for a p-value to be a candidate for rejection: ADDIS will never reject a p-value larger than λ. The algorithm also require a sequence of non-negative non-increasing numbers γ_i that sum to 1.

The ADDIS procedure provably controls the FDR for independent p-values. Tian and Ramdas (2019) also presented a version for an asynchronous testing process, consisting of tests that start and finish at (potentially) random times. The discretised finish times of the test correspond to the decision times. These decision times are given as the input `decision.times`. Note that this asynchronous version controls a modified version of the FDR.

Further details of the ADDIS algorithms can be found in Tian and Ramdas (2019).

Value

`out`	A dataframe with the original p-values `pval`, the adjusted testing levels α_i and the indicator function of discoveries `R`. Hypothesis i is rejected if the i-th p-value is less than or equal to α_i, in which case `R[i] = 1` (otherwise `R[i] = 0`).

References

Tian, J. and Ramdas, A. (2019). ADDIS: an adaptive discarding algorithm for online FDR control with conservative nulls. *Advances in Neural Information Processing Systems*, 9388-9396.

See Also

ADDIS is identical to `SAFFRON` with option `discard=TRUE`.

ADDIS with option `async=TRUE` is identical to `SAFFRONstar` with option `discard=TRUE`.

Examples

```
sample.df1 <- data.frame(
id = c('A15432', 'B90969', 'C18705', 'B49731', 'E99902',
    'C38292', 'A30619', 'D46627', 'E29198', 'A41418',
    'D51456', 'C88669', 'E03673', 'A63155', 'B66033'),
date = as.Date(c(rep('2014-12-01',3),
                 rep('2015-09-21',5),
                 rep('2016-05-19',2),
                 '2016-11-12',
                 rep('2017-03-27',4))),
pval = c(2.90e-08, 0.06743, 0.01514, 0.08174, 0.00171,
        3.60e-05, 0.79149, 0.27201, 0.28295, 7.59e-08,
        0.69274, 0.30443, 0.00136, 0.72342, 0.54757))

ADDIS(sample.df1, random=FALSE)


sample.df2 <- data.frame(
id = c('A15432', 'B90969', 'C18705', 'B49731', 'E99902',
    'C38292', 'A30619', 'D46627', 'E29198', 'A41418',
    'D51456', 'C88669', 'E03673', 'A63155', 'B66033'),
pval = c(2.90e-08, 0.06743, 0.01514, 0.08174, 0.00171,
        3.60e-05, 0.79149, 0.27201, 0.28295, 7.59e-08,
        0.69274, 0.30443, 0.00136, 0.72342, 0.54757),
decision.times = seq_len(15) + 1)

ADDIS(sample.df2, async = TRUE) # Asynchronous
```

`ADDIS_spending` *ADDIS-spending: Adaptive discarding algorithm for online FWER control*

Description

Implements the ADDIS algorithm for online FWER control, where ADDIS stands for an ADaptive algorithm that DIScards conservative nulls, as presented by Tian and Ramdas (2021). The procedure compensates for the power loss of Alpha-spending, by including both adaptivity in the fraction of null hypotheses and the conservativeness of nulls.

Usage

```
ADDIS_spending(
  d,
  alpha = 0.05,
  gammai,
  lambda = 0.25,
  tau = 0.5,
```

```
  dep = FALSE,
  display_progress = FALSE
)
```

Arguments

d	Either a vector of p-values, or a dataframe with three columns: an identifier ('id'), p-value ('pval'), and lags ('lags').
alpha	Overall significance level of the procedure, the default is 0.05.
gammai	Optional vector of γ_i. A default is provided with γ_j proportional to $1/j^(1.6)$.
lambda	Optional parameter that sets the threshold for 'candidate' hypotheses. Must be between 0 and 1, defaults to 0.25.
tau	Optional threshold for hypotheses to be selected for testing. Must be between 0 and 1, defaults to 0.5.
dep	Logical. If TRUE runs the version for locally dependent p-values
display_progress	Logical. If TRUE prints out a progress bar for the algorithm runtime.

Details

The function takes as its input either a vector of p-values, or a dataframe with three columns: an identifier ('id'), p-value ('pval'), and lags, if the dependent version is specified (see below). Given an overall significance level α, ADDIS depends on constants λ and τ, where $\lambda < \tau$. Here $\tau \in (0, 1)$ represents the threshold for a hypothesis to be selected for testing: p-values greater than τ are implicitly 'discarded' by the procedure, while $\lambda \in (0, 1)$ sets the threshold for a p-value to be a candidate for rejection: ADDIS-spending will never reject a p-value larger than λ. The algorithms also require a sequence of non-negative non-increasing numbers γ_i that sum to 1.

The ADDIS-spending procedure provably controls the FWER in the strong sense for independent p-values. Note that the procedure also controls the generalised familywise error rate (k-FWER) for $k > 1$ if α is replaced by min$(1, k\alpha)$.

Tian and Ramdas (2021) also presented a version for handling local dependence. More precisely, for any $t > 0$ we allow the p-value p_t to have arbitrary dependence on the previous L_t p-values. The fixed sequence L_t is referred to as 'lags', and is given as the input lags for this version of the ADDIS-spending algorithm.

Further details of the ADDIS-spending algorithms can be found in Tian and Ramdas (2021).

Value

out	A dataframe with the original p-values pval, the adjusted testing levels α_i and the indicator function of discoveries R. Hypothesis i is rejected if the i-th p-value is less than or equal to α_i, in which case R[i] = 1 (otherwise R[i] = 0).

References

Tian, J. and Ramdas, A. (2021). Online control of the familywise error rate. *Statistical Methods for Medical Research* 30(4):976–993.

See Also

ADDIS provides online control of the FDR.

Examples

```
sample.df <- data.frame(
id = c('A15432', 'B90969', 'C18705', 'B49731', 'E99902',
    'C38292', 'A30619', 'D46627', 'E29198', 'A41418',
    'D51456', 'C88669', 'E03673', 'A63155', 'B66033'),
pval = c(2.90e-08, 0.06743, 0.01514, 0.08174, 0.00171,
        3.60e-05, 0.79149, 0.27201, 0.28295, 7.59e-08,
        0.69274, 0.30443, 0.00136, 0.72342, 0.54757),
lags = rep(1,15))

ADDIS_spending(sample.df) #independent

ADDIS_spending(sample.df, dep = TRUE) #Locally dependent
```

Alpha_investing *Alpha-investing for online FDR control*

Description

Implements a variant of the Alpha-investing algorithm of Foster and Stine (2008) that guarantees FDR control, as proposed by Ramdas et al. (2018). This procedure uses SAFFRON's update rule with the constant λ replaced by a sequence $\lambda_i = \alpha_i$. This is also equivalent to using the ADDIS algorithm with $\tau = 1$ and $\lambda_i = \alpha_i$.

Usage

```
Alpha_investing(
  d,
  alpha = 0.05,
  gammai,
  w0,
  random = TRUE,
  display_progress = FALSE,
  date.format = "%Y-%m-%d"
)
```

Arguments

d	Either a vector of p-values, or a dataframe with three columns: an identifier ('id'), date ('date') and p-value ('pval'). If no column of dates is provided, then the p-values are treated as being ordered in sequence.
alpha	Overall significance level of the FDR procedure, the default is 0.05.

gammai	Optional vector of γ_i. A default is provided with γ_j proportional to $1/j^(1.6)$.
w0	Initial 'wealth' of the procedure, defaults to $\alpha/2$. Must be between 0 and α.
random	Logical. If TRUE (the default), then the order of the p-values in each batch (i.e. those that have exactly the same date) is randomised.
display_progress	Logical. If TRUE prints out a progress bar for the algorithm runtime.
date.format	Optional string giving the format that is used for dates.

Details

The function takes as its input either a vector of p-values or a dataframe with three columns: an identifier ('id'), date ('date') and p-value ('pval'). The case where p-values arrive in batches corresponds to multiple instances of the same date. If no column of dates is provided, then the p-values are treated as being ordered in sequence.

The Alpha-investing procedure provably controls FDR for independent p-values. Given an overall significance level α, we choose a sequence of non-negative non-increasing numbers γ_i that sum to 1. Alpha-investing depends on a constant w_0, which satisfies $0 \leq w_0 \leq \alpha$ and represents the initial 'wealth' of the procedure.

Further details of the Alpha-investing procedure and its modification can be found in Foster and Stine (2008) and Ramdas et al. (2018).

Value

out	A dataframe with the original data d (which will be reordered if there are batches and random = TRUE), the LORD-adjusted significance thresholds α_i and the indicator function of discoveries R. Hypothesis i is rejected if the i-th p-value is less than or equal to α_i, in which case R[i] = 1 (otherwise R[i] = 0).

References

Foster, D. and Stine R. (2008). α-investing: a procedure for sequential control of expected false discoveries. *Journal of the Royal Statistical Society (Series B)*, 29(4):429-444.

Ramdas, A., Zrnic, T., Wainwright M.J. and Jordan, M.I. (2018). SAFFRON: an adaptive algorithm for online control of the false discovery rate. *Proceedings of the 35th International Conference in Machine Learning*, 80:4286-4294.

See Also

SAFFRON uses the update rule of Alpha-investing but with constant λ.

Examples

```
sample.df <- data.frame(
id = c('A15432', 'B90969', 'C18705', 'B49731', 'E99902',
    'C38292', 'A30619', 'D46627', 'E29198', 'A41418',
    'D51456', 'C88669', 'E03673', 'A63155', 'B66033'),
date = as.Date(c(rep('2014-12-01',3),
               rep('2015-09-21',5),
```

```
                rep('2016-05-19',2),
                '2016-11-12',
               rep('2017-03-27',4))),
pval = c(2.90e-08, 0.06743, 0.01514, 0.08174, 0.00171,
        3.60e-05, 0.79149, 0.27201, 0.28295, 7.59e-08,
        0.69274, 0.30443, 0.00136, 0.72342, 0.54757))

Alpha_investing(sample.df, random=FALSE)

set.seed(1); Alpha_investing(sample.df)

set.seed(1); Alpha_investing(sample.df, alpha=0.1, w0=0.025)
```

`Alpha_spending` *Alpha-spending for online FWER control*

Description

Implements online FWER control using a Bonferroni-like test.

Usage

```
Alpha_spending(
  d,
  alpha = 0.05,
  gammai,
  random = TRUE,
  date.format = "%Y-%m-%d"
)
```

Arguments

`d`	Either a vector of p-values, or a dataframe with three columns: an identifier ('id'), date ('date') and p-value ('pval'). If no column of dates is provided, then the p-values are treated as being ordered in sequence, arriving one at a time.
`alpha`	Overall significance level of the FDR procedure, the default is 0.05.
`gammai`	Optional vector of γ_i, where hypothesis i is rejected if the i-th p-value is less than or equal to $\alpha\gamma_i$. A default is provided as proposed by Javanmard and Montanari (2018), equation 31.
`random`	Logical. If `TRUE` (the default), then the order of the p-values in each batch (i.e. those that have exactly the same date) is randomised.
`date.format`	Optional string giving the format that is used for dates.

Details

The function takes as its input either a vector of p-values, or a dataframe with three columns: an identifier ('id'), date ('date') and p-value ('pval'). The case where p-values arrive in batches corresponds to multiple instances of the same date. If no column of dates is provided, then the p-values are treated as being ordered in sequence, arriving one at a time.

Alpha-spending provides strong FWER control for a potentially infinite stream of p-values by using a Bonferroni-like test. Given an overall significance level α, we choose a (potentially infinite) sequence of non-negative numbers γ_i such that they sum to 1. Hypothesis i is rejected if the i-th p-value is less than or equal to $\alpha\gamma_i$.

Note that the procedure controls the generalised familywise error rate (k-FWER) for $k > 1$ if α is replaced by $\min(1, k\alpha)$.

Value

`out` A dataframe with the original data `d` (which will be reordered if there are batches and `random = TRUE`), the adjusted signifcance thresholds `alphai` and the indicator function of discoveries `R`, where `R[i] = 1` corresponds to hypothesis i being rejected (otherwise `R[i] = 0`).

References

Javanmard, A. and Montanari, A. (2018) Online Rules for Control of False Discovery Rate and False Discovery Exceedance. *Annals of Statistics*, 46(2):526-554.

Tian, J. and Ramdas, A. (2021). Online control of the familywise error rate. *Statistical Methods for Medical Research* (to appear), https://arxiv.org/abs/1910.04900.

Examples

```
sample.df <- data.frame(
id = c('A15432', 'B90969', 'C18705', 'B49731', 'E99902',
    'C38292', 'A30619', 'D46627', 'E29198', 'A41418',
    'D51456', 'C88669', 'E03673', 'A63155', 'B66033'),
date = as.Date(c(rep('2014-12-01',3),
                 rep('2015-09-21',5),
                 rep('2016-05-19',2),
                 '2016-11-12',
                 rep('2017-03-27',4))),
pval = c(2.90e-17, 0.06743, 0.01514, 0.08174, 0.00171,
        3.60e-05, 0.79149, 0.27201, 0.28295, 7.59e-08,
        0.69274, 0.30443, 0.00136, 0.72342, 0.54757))

set.seed(1); Alpha_spending(sample.df)

Alpha_spending(sample.df, random=FALSE)

set.seed(1); Alpha_spending(sample.df, alpha=0.1)
```

BatchBH *BatchBH: Online batch FDR control using the BH procedure*

Description

Implements the BatchBH algorithm for online FDR control, as presented by Zrnic et al. (2020).

Usage

```
BatchBH(d, alpha = 0.05, gammai, display_progress = FALSE)
```

Arguments

d	A dataframe with three columns: identifiers ('id'), batch numbers ('batch') and p-values ('pval').
alpha	Overall significance level of the FDR procedure, the default is 0.05.
gammai	Optional vector of γ_i. A default is provided with γ_j proportional to $1/j^(1.6)$.
display_progress	Logical. If TRUE prints out a progress bar for the algorithm runtime.

Details

The function takes as its input a dataframe with three columns: identifiers ('id'), batch numbers ('batch') and p-values ('pval').

The BatchBH algorithm controls the FDR when the p-values in a batch are independent, and independent across batches. Given an overall significance level α, we choose a sequence of non-negative numbers γ_i such that they sum to 1. The algorithm runs the Benjamini-Hochberg procedure on each batch, where the values of the adjusted significance thresholds α_{t+1} depend on the number of previous discoveries.

Further details of the BatchBH algorithm can be found in Zrnic et al. (2020).

Value

out	A dataframe with the original data d and the indicator function of discoveries R. Hypothesis i is rejected if the i-th p-value within the t-th batch is less than or equal to $(r/n)\alpha_t$, where r is the rank of the i-th p-value within an ordered set and n is the total number of hypotheses within the t-th batch. If hypothesis i is rejected, R[i] = 1 (otherwise R[i] = 0).

References

Zrnic, T., Jiang D., Ramdas A. and Jordan M. (2020). The Power of Batching in Multiple Hypothesis Testing. *International Conference on Artificial Intelligence and Statistics*, 3806-3815.

Examples

```
sample.df <- data.frame(
id = c('A15432', 'B90969', 'C18705', 'B49731', 'E99902',
    'C38292', 'A30619', 'D46627', 'E29198', 'A41418',
    'D51456', 'C88669', 'E03673', 'A63155', 'B66033'),
pval = c(2.90e-08, 0.06743, 0.01514, 0.08174, 0.00171,
        3.60e-05, 0.79149, 0.27201, 0.28295, 7.59e-08,
        0.69274, 0.30443, 0.00136, 0.72342, 0.54757),
batch = c(rep(1,5), rep(2,6), rep(3,4)))

BatchBH(sample.df)
```

`BatchPRDS` *BatchPRDS: Online batch FDR control under Positive Dependence*

Description

Implements the BatchPRDS algorithm for online FDR control, where PRDS stands for positive regression dependency on a subset, as presented by Zrnic et al. (2020).

Usage

```
BatchPRDS(d, alpha = 0.05, gammai, display_progress = FALSE)
```

Arguments

`d`	A dataframe with three columns: identifiers ('id'), batch numbers ('batch') and p-values ('pval').
`alpha`	Overall significance level of the FDR procedure, the default is 0.05.
`gammai`	Optional vector of γ_i. A default is provided with γ_j proportional to $1/j^(1.6)$.
`display_progress`	Logical. If `TRUE` prints out a progress bar for the algorithm runtime.

Details

The function takes as its input a dataframe with three columns: identifiers ('id'), batch numbers ('batch') and p-values ('pval').

The BatchPRDS algorithm controls the FDR when the p-values in one batch are positively dependent, and independent across batches. Given an overall significance level α, we choose a sequence of non-negative numbers γ_i such that they sum to 1. The algorithm runs the Benjamini-Hochberg procedure on each batch, where the values of the adjusted significance thresholds α_{t+1} depend on the number of previous discoveries.

Further details of the BatchPRDS algorithm can be found in Zrnic et al. (2020).

Value

`out` A dataframe with the original data `d` and the indicator function of discoveries `R`. Hypothesis i is rejected if the i-th p-value within the t-th batch is less than or equal to $(r/n)\alpha_t$, where r is the rank of the i-th p-value within an ordered set and n is the total number of hypotheses within the t-th batch. If hypothesis i is rejected, `R[i] = 1` (otherwise `R[i] = 0`).

References

Zrnic, T., Jiang D., Ramdas A. and Jordan M. (2020). The Power of Batching in Multiple Hypothesis Testing. *International Conference on Artificial Intelligence and Statistics*: 3806-3815

Examples

```
sample.df <- data.frame(
id = c('A15432', 'B90969', 'C18705', 'B49731', 'E99902',
    'C38292', 'A30619', 'D46627', 'E29198', 'A41418',
    'D51456', 'C88669', 'E03673', 'A63155', 'B66033'),
pval = c(2.90e-08, 0.06743, 0.01514, 0.08174, 0.00171,
        3.60e-05, 0.79149, 0.27201, 0.28295, 7.59e-08,
        0.69274, 0.30443, 0.00136, 0.72342, 0.54757),
batch = c(rep(1,5), rep(2,6), rep(3,4)))

BatchPRDS(sample.df)
```

`BatchStBH` *BatchStBH: Online batch FDR control using the St-BH procedure*

Description

Implements the BatchSt-BH algorithm for online FDR control, as presented by Zrnic et al. (2020). This algorithm makes one modification to the original Storey-BH algorithm (Storey 2002), by adding 1 to the numerator of the null proportion estimate for more stable results.

Usage

```
BatchStBH(d, alpha = 0.05, gammai, lambda = 0.5, display_progress = FALSE)
```

Arguments

`d` A dataframe with three columns: identifiers ('id'), batch numbers ('batch') and p-values ('pval').

`alpha` Overall significance level of the FDR procedure, the default is 0.05.

`gammai` Optional vector of γ_i. A default is provided with γ_j proportional to $1/j^(1.6)$.

`lambda` Threshold for Storey-BH, must be between 0 and 1. Defaults to 0.5.

`display_progress` Logical. If `TRUE` prints out a progress bar for the algorithm runtime.

Details

The function takes as its input a dataframe with three columns: identifiers ('id'), batch numbers ('batch') and p-values ('pval').

The BatchSt-BH algorithm controls the FDR when the p-values in a batch are independent, and independent across batches. Given an overall significance level α, we choose a sequence of non-negative numbers γ_i such that they sum to 1. The algorithm runs the Storey Benjamini-Hochberg procedure on each batch, where the values of the adjusted significance thresholds α_{t+1} depend on the number of previous discoveries.

Further details of the BatchSt-BH algorithm can be found in Zrnic et al. (2020).

Value

`out` A dataframe with the original data `d` and the indicator function of discoveries `R`. Hypothesis i is rejected if the i-th p-value within the t-th batch is less than or equal to $(r/n)\alpha_t$, where r is the rank of the i-th p-value within an ordered set and n is the total number of hypotheses within the t-th batch. If hypothesis i is rejected, `R[i] = 1` (otherwise `R[i] = 0`).

References

Storey, J.D. (2002). A direct approach to false discovery rates. *J. R. Statist. Soc. B*: 64, Part 3, 479-498.

Zrnic, T., Jiang D., Ramdas A. and Jordan M. (2020). The Power of Batching in Multiple Hypothesis Testing. *International Conference on Artificial Intelligence and Statistics*: 3806-3815

Examples

```
sample.df <- data.frame(
id = c('A15432', 'B90969', 'C18705', 'B49731', 'E99902',
    'C38292', 'A30619', 'D46627', 'E29198', 'A41418',
    'D51456', 'C88669', 'E03673', 'A63155', 'B66033'),
pval = c(2.90e-08, 0.06743, 0.01514, 0.08174, 0.00171,
        3.60e-05, 0.79149, 0.27201, 0.28295, 7.59e-08,
        0.69274, 0.30443, 0.00136, 0.72342, 0.54757),
batch = c(rep(1,5), rep(2,6), rep(3,4)))

BatchStBH(sample.df)
```

---

bonfInfinite *Online FDR control based on a Bonferroni-like test*

Description

This funcion is deprecated, please use `Alpha_spending` instead.

Usage

```
bonfInfinite(
  d,
  alpha = 0.05,
  alphai,
  random = TRUE,
  date.format = "%Y-%m-%d"
)
```

Arguments

d	Either a vector of p-values, or a dataframe with three columns: an identifier ('id'), date ('date') and p-value ('pval'). If no column of dates is provided, then the p-values are treated as being ordered in sequence, arriving one at a time.
alpha	Overall significance level of the FDR procedure, the default is 0.05.
alphai	Optional vector of α_i, where hypothesis i is rejected if the i-th p-value is less than or equal to α_i. A default is provided as proposed by Javanmard and Montanari (2018), equation 31.
random	Logical. If TRUE (the default), then the order of the p-values in each batch (i.e. those that have exactly the same date) is randomised.
date.format	Optional string giving the format that is used for dates.

Details

Implements online FDR control using a Bonferroni-like test.

The function takes as its input either a vector of p-values, or a dataframe with three columns: an identifier ('id'), date ('date') and p-value ('pval'). The case where p-values arrive in batches corresponds to multiple instances of the same date. If no column of dates is provided, then the p-values are treated as being ordered in sequence, arriving one at a time.

The procedure controls FDR for a potentially infinite stream of p-values by using a Bonferroni-like test. Given an overall significance level α, we choose a (potentially infinite) sequence of non-negative numbers α_i such that they sum to α. Hypothesis i is rejected if the i-th p-value is less than or equal to α_i.

Value

d.out	A dataframe with the original data d (which will be reordered if there are batches and random = TRUE), the adjusted signifcance thresholds alphai and the indicator function of discoveries R, where R[i] = 1 corresponds to hypothesis i being rejected (otherwise R[i] = 0).

References

Javanmard, A. and Montanari, A. (2018) Online Rules for Control of False Discovery Rate and False Discovery Exceedance. *Annals of Statistics*, 46(2):526-554.

LOND *LOND: Online FDR control based on number of discoveries*

Description

Implements the LOND algorithm for online FDR control, where LOND stands for (significance) Levels based On Number of Discoveries, as presented by Javanmard and Montanari (2015).

Usage

```
LOND(
  d,
  alpha = 0.05,
  betai,
  dep = FALSE,
  random = TRUE,
  display_progress = FALSE,
  date.format = "%Y-%m-%d",
  original = TRUE
)
```

Arguments

d	Either a vector of p-values, or a dataframe with three columns: an identifier ('id'), date ('date') and p-value ('pval'). If no column of dates is provided, then the p-values are treated as being ordered in sequence, arriving one at a time.
alpha	Overall significance level of the FDR procedure, the default is 0.05.
betai	Optional vector of β_i. A default is provided as proposed by Javanmard and Montanari (2018), equation 31.
dep	Logical. If TRUE, runs the modified LOND algorithm which guarantees FDR control for *dependent* p-values. Defaults to FALSE.
random	Logical. If TRUE (the default), then the order of the p-values in each batch (i.e. those that have exactly the same date) is randomised.
display_progress	Logical. If TRUE prints out a progress bar for the algorithm runtime.
date.format	Optional string giving the format that is used for dates.
original	Logical. If TRUE, runs the original LOND algorithm of Javanmard and Montanari (2015), otherwise runs the modified algorithm of Zrnic et al. (2018). Defaults to TRUE.

Details

The function takes as its input either a vector of p-values, or a dataframe with three columns: an identifier ('id'), date ('date') and p-value ('pval'). The case where p-values arrive in batches

corresponds to multiple instances of the same date. If no column of dates is provided, then the p-values are treated as being ordered in sequence, arriving one at a time.

The LOND algorithm controls the FDR for independent p-values (see below for the modification for dependent p-values). Given an overall significance level α, we choose a sequence of non-negative numbers β_i such that they sum to α. The values of the adjusted significance thresholds α_i are chosen as follows:

$$\alpha_i = (D(i-1)+1)\beta_i$$

where $D(n)$ denotes the number of discoveries in the first n hypotheses.

A slightly modified version of LOND with thresholds $\alpha_i = max(D(i-1),1)\beta_i$ provably controls the FDR under positive dependence (PRDS condition), see Zrnic et al. (2021).

For arbitrarily dependent p-values, LOND controls the FDR if it is modified with $\beta_i/H(i)$ in place of β_i, where $H(j)$ is the i-th harmonic number.

Further details of the LOND algorithm can be found in Javanmard and Montanari (2015).

Value

`out` A dataframe with the original data `d` (which will be reordered if there are batches and `random = TRUE`), the LOND-adjusted significance thresholds α_i and the indicator function of discoveries `R`. Hypothesis i is rejected if the i-th p-value is less than or equal to α_i, in which case `R[i] = 1` (otherwise `R[i] = 0`).

References

Javanmard, A. and Montanari, A. (2015) On Online Control of False Discovery Rate. *arXiv preprint*, https://arxiv.org/abs/1502.06197.

Javanmard, A. and Montanari, A. (2018) Online Rules for Control of False Discovery Rate and False Discovery Exceedance. *Annals of Statistics*, 46(2):526-554.

Zrnic, T., Ramdas, A. and Jordan, M.I. (2021). Asynchronous Online Testing of Multiple Hypotheses. *Journal of Machine Learning Research* (to appear), https://arxiv.org/abs/1812.05068.

See Also

LONDstar presents versions of LORD for *synchronous* p-values, i.e. where each test can only start when the previous test has finished.

Examples

```
sample.df <- data.frame(
id = c('A15432', 'B90969', 'C18705', 'B49731', 'E99902',
    'C38292', 'A30619', 'D46627', 'E29198', 'A41418',
    'D51456', 'C88669', 'E03673', 'A63155', 'B66033'),
date = as.Date(c(rep('2014-12-01',3),
                 rep('2015-09-21',5),
                 rep('2016-05-19',2),
                 '2016-11-12',
                 rep('2017-03-27',4))),
pval = c(2.90e-08, 0.06743, 0.01514, 0.08174, 0.00171,
        3.60e-05, 0.79149, 0.27201, 0.28295, 7.59e-08,
```

```
        0.69274, 0.30443, 0.00136, 0.72342, 0.54757))

set.seed(1); LOND(sample.df)

LOND(sample.df, random=FALSE)

set.seed(1); LOND(sample.df, alpha=0.1)
```

`LONDstar` *LONDstar: Asynchronous online mFDR control based on number of discoveries*

Description

Implements the LOND algorithm for asynchronous online testing, as presented by Zrnic et al. (2021).

Usage

```
LONDstar(
  d,
  alpha = 0.05,
  version,
  betai,
  batch.sizes,
  display_progress = FALSE
)
```

Arguments

`d`	Either a vector of p-values, or a dataframe with three columns: an identifier ('id'), p-value ('pval'), and either 'decision.times', or 'lags', depending on which version you're using. See version for more details.
`alpha`	Overall significance level of the procedure, the default is 0.05.
`version`	Takes values 'async', 'dep' or 'batch'. This specifies the version of LONDstar to use. `version='async'` requires a column of decision times ('decision.times'). `version='dep'` requires a column of lags ('lags'). `version='batch'` requires a vector of batch sizes ('batch.sizes').
`betai`	Optional vector of β_i. A default is provided as proposed by Javanmard and Montanari (2018), equation 31.
`batch.sizes`	A vector of batch sizes, this is required for `version='batch'`.
`display_progress`	Logical. If `TRUE` prints out a progress bar for the algorithm runtime.

Details

The function takes as its input either a vector of p-values, or a dataframe with three columns: an identifier ('id'), p-value ('pval'), or a column describing the conflict sets for the hypotheses. This takes the form of a vector of decision times or lags. Batch sizes can be specified as a separate argument (see below).

Zrnic et al. (2021) present explicit three versions of LONDstar:

1) `version='async'` is for an asynchronous testing process, consisting of tests that start and finish at (potentially) random times. The discretised finish times of the test correspond to the decision times. These decision times are given as the input `decision.times` for this version of the LONDstar algorithm.

2) `version='dep'` is for online testing under local dependence of the p-values. More precisely, for any $t > 0$ we allow the p-value p_t to have arbitrary dependence on the previous L_t p-values. The fixed sequence L_t is referred to as 'lags', and is given as the input `lags` for this version of the LONDstar algorithm.

3) `version='batch'` is for controlling the mFDR in mini-batch testing, where a mini-batch represents a grouping of tests run asynchronously which result in dependent p-values. Once a mini-batch of tests is fully completed, a new one can start, testing hypotheses independent of the previous batch. The batch sizes are given as the input `batch.sizes` for this version of the LONDstar algorithm.

Given an overall significance level α, LONDstar requires a sequence of non-negative non-increasing numbers β_i that sum to α.

Note that these LONDstar algorithms control the *modified* FDR (mFDR).

Further details of the LONDstar algorithms can be found in Zrnic et al. (2021).

Value

`out` A dataframe with the original p-values `pval`, the adjusted testing levels α_i and the indicator function of discoveries `R`. Hypothesis i is rejected if the i-th p-value is less than or equal to α_i, in which case `R[i] = 1` (otherwise `R[i] = 0`).

References

Javanmard, A. and Montanari, A. (2018) Online Rules for Control of False Discovery Rate and False Discovery Exceedance. *Annals of Statistics*, 46(2):526-554.

Zrnic, T., Ramdas, A. and Jordan, M.I. (2021). Asynchronous Online Testing of Multiple Hypotheses. *Journal of Machine Learning Research* (to appear), https://arxiv.org/abs/1812.05068.

See Also

LOND presents versions of LOND for *synchronous* p-values, i.e. where each test can only start when the previous test has finished.

Examples

```
sample.df <- data.frame(
id = c('A15432', 'B90969', 'C18705', 'B49731', 'E99902',
    'C38292', 'A30619', 'D46627', 'E29198', 'A41418',
    'D51456', 'C88669', 'E03673', 'A63155', 'B66033'),
```

```
pval = c(2.90e-08, 0.06743, 0.01514, 0.08174, 0.00171,
        3.60e-05, 0.79149, 0.27201, 0.28295, 7.59e-08,
        0.69274, 0.30443, 0.00136, 0.72342, 0.54757),
decision.times = seq_len(15) + 1)

LONDstar(sample.df, version='async')

sample.df2 <- data.frame(
id = c('A15432', 'B90969', 'C18705', 'B49731', 'E99902',
    'C38292', 'A30619', 'D46627', 'E29198', 'A41418',
    'D51456', 'C88669', 'E03673', 'A63155', 'B66033'),
pval = c(2.90e-08, 0.06743, 0.01514, 0.08174, 0.00171,
        3.60e-05, 0.79149, 0.27201, 0.28295, 7.59e-08,
        0.69274, 0.30443, 0.00136, 0.72342, 0.54757),
lags = rep(1,15))

LONDstar(sample.df2, version='dep')

sample.df3 <- data.frame(
id = c('A15432', 'B90969', 'C18705', 'B49731', 'E99902',
    'C38292', 'A30619', 'D46627', 'E29198', 'A41418',
    'D51456', 'C88669', 'E03673', 'A63155', 'B66033'),
pval = c(2.90e-08, 0.06743, 0.01514, 0.08174, 0.00171,
        3.60e-05, 0.79149, 0.27201, 0.28295, 7.59e-08,
        0.69274, 0.30443, 0.00136, 0.72342, 0.54757))

LONDstar(sample.df3, version='batch', batch.sizes = c(4,6,5))
```

LORD *LORD: Online FDR control based on recent discovery*

Description

Implements the LORD procedure for online FDR control, where LORD stands for (significance) Levels based On Recent Discovery, as presented by Javanmard and Montanari (2018) and Ramdas et al. (2017).

Usage

```
LORD(
  d,
  alpha = 0.05,
  gammai,
  version = "++",
  w0,
  b0,
  tau.discard = 0.5,
  random = TRUE,
```

```
  display_progress = FALSE,
  date.format = "%Y-%m-%d"
)
```

Arguments

d	Either a vector of p-values, or a dataframe with three columns: an identifier ('id'), date ('date') and p-value ('pval'). If no column of dates is provided, then the p-values are treated as being ordered in sequence, arriving one at a time.
alpha	Overall significance level of the FDR procedure, the default is 0.05.
gammai	Optional vector of γ_i. A default is provided as proposed by Javanmard and Montanari (2018), equation 31 for all versions of LORD except 'dep'. The latter is provided a default to satisfy a condition given in Javanmard and Montanari (2018), example 3.8.
version	Takes values '++', 3, 'discard', or 'dep'. This specifies the version of LORD to use, and defaults to '++'.
w0	Initial 'wealth' of the procedure, defaults to $\alpha/10$.
b0	The 'payout' for rejecting a hypothesis in all versions of LORD except for '++'. Defaults to $\alpha - w_0$.
tau.discard	Optional threshold for hypotheses to be selected for testing. Must be between 0 and 1, defaults to 0.5. This is required if `version='discard'`.
random	Logical. If `TRUE` (the default), then the order of the p-values in each batch (i.e. those that have exactly the same date) is randomised.
display_progress	Logical. If `TRUE` prints out a progress bar for the algorithm runtime.
date.format	Optional string giving the format that is used for dates.

Details

The function takes as its input either a vector of p-values or a dataframe with three columns: an identifier ('id'), date ('date') and p-value ('pval'). The case where p-values arrive in batches corresponds to multiple instances of the same date. If no column of dates is provided, then the p-values are treated as being ordered in sequence, arriving one at a time..

The LORD procedure provably controls FDR for independent p-values (see below for dependent p-values). Given an overall significance level α, we choose a sequence of non-negative non-increasing numbers γ_i that sum to 1.

Javanmard and Montanari (2018) presented versions of LORD which differ in the way the adjusted significance thresholds α_i are calculated. The significance thresholds for LORD 2 are based on all previous discovery times. LORD 2 has been superseded by the algorithm given in Ramdas et al. (2017), LORD++ (`version='++'`), which is the default version. The significance thresholds for LORD 3 (`version=3`) are based on the time of the last discovery as well as the 'wealth' accumulated at that time. Finally, Tian and Ramdas (2019) presented a version of LORD (`version='discard'`) that can improve the power of the procedure in the presence of conservative nulls by adaptively 'discarding' these p-values.

LORD depends on constants w_0 and (for versions 3 and 'dep') b_0, where $0 \leq w_0 \leq \alpha$ represents the initial 'wealth' of the procedure and $b_0 > 0$ represents the 'payout' for rejecting a hypothesis.

We require $w_0 + b_0 \leq \alpha$ for FDR control to hold. Version 'discard' also depends on a constant τ, where $\tau \in (0, 1)$ represents the threshold for a hypothesis to be selected for testing: p-values greater than τ are implicitly 'discarded' by the procedure.

Note that FDR control also holds for the LORD procedure if only the p-values corresponding to true nulls are mutually independent, and independent from the non-null p-values.

For dependent p-values, a modified LORD procedure was proposed in Javanmard and Montanari (2018), which is called be setting `version='dep'`. Given an overall significance level α, we choose a sequence of non-negative numbers ξ_i such that they satisfy a condition given in Javanmard and Montanari (2018), example 3.8.

Further details of the LORD procedures can be found in Javanmard and Montanari (2018), Ramdas et al. (2017) and Tian and Ramdas (2019).

Value

`d.out` A dataframe with the original data `d` (which will be reordered if there are batches and `random = TRUE`), the LORD-adjusted significance thresholds α_i and the indicator function of discoveries `R`. Hypothesis i is rejected if the i-th p-value is less than or equal to α_i, in which case `R[i] = 1` (otherwise `R[i] = 0`).

References

Javanmard, A. and Montanari, A. (2018) Online Rules for Control of False Discovery Rate and False Discovery Exceedance. *Annals of Statistics*, 46(2):526-554.

Ramdas, A., Yang, F., Wainwright M.J. and Jordan, M.I. (2017). Online control of the false discovery rate with decaying memory. *Advances in Neural Information Processing Systems 30*, 5650-5659.

Tian, J. and Ramdas, A. (2019). ADDIS: an adaptive discarding algorithm for online FDR control with conservative nulls. *Advances in Neural Information Processing Systems*, 9388-9396.

See Also

`LORDstar` presents versions of LORD for *asynchronous* testing, i.e. where each hypothesis test can itself be a sequential process and the tests can overlap in time.

Examples

```
sample.df <- data.frame(
id = c('A15432', 'B90969', 'C18705', 'B49731', 'E99902',
    'C38292', 'A30619', 'D46627', 'E29198', 'A41418',
    'D51456', 'C88669', 'E03673', 'A63155', 'B66033'),
date = as.Date(c(rep('2014-12-01',3),
                 rep('2015-09-21',5),
                 rep('2016-05-19',2),
                 '2016-11-12',
                 rep('2017-03-27',4))),
pval = c(2.90e-08, 0.06743, 0.01514, 0.08174, 0.00171,
        3.60e-05, 0.79149, 0.27201, 0.28295, 7.59e-08,
        0.69274, 0.30443, 0.00136, 0.72342, 0.54757))
```

```
LORD(sample.df, random=FALSE)

set.seed(1); LORD(sample.df, version='dep')

set.seed(1); LORD(sample.df, version='discard')

set.seed(1); LORD(sample.df, alpha=0.1, w0=0.05)
```

`LORDdep` *LORD (dep): Online FDR control based on recent discovery for dependent p-values*

Description

This funcion is deprecated, please use `LORD` instead with `version = 'dep'`.

Usage

```
LORDdep(
  d,
  alpha = 0.05,
  xi,
  w0 = alpha/10,
  b0 = alpha - w0,
  random = TRUE,
  date.format = "%Y-%m-%d"
)
```

Arguments

`d`	Either a vector of p-values, or a dataframe with three columns: an identifier ('id'), date ('date') and p-value ('pval'). If no column of dates is provided, then the p-values are treated as being ordered in sequence, arriving one at a time.
`alpha`	Overall significance level of the FDR procedure, the default is 0.05.
`xi`	Optional vector of ξ_i. A default is provided to satisfy the condition given in Javanmard and Montanari (2018), example 3.7.
`w0`	Initial 'wealth' of the procedure. Defaults to $\alpha/10$.
`b0`	The 'payout' for rejecting a hypothesis. Defaults to $\alpha - w_0$.
`random`	Logical. If `TRUE` (the default), then the order of the p-values in each batch (i.e. those that have exactly the same date) is randomised.
`date.format`	Optional string giving the format that is used for dates.

Details

LORDdep implements the LORD procedure for online FDR control for dependent p-values, where LORD stands for (significance) Levels based On Recent Discovery, as presented by Javanmard and Montanari (2018).

The function takes as its input either a vector of p-values or a dataframe with three columns: an identifier ('id'), date ('date') and p-value ('pval'). The case where p-values arrive in batches corresponds to multiple instances of the same date. If no column of dates is provided, then the p-values are treated as being ordered in sequence, arriving one at a time.

This modified LORD procedure controls FDR for dependent p-values. Given an overall significance level α, we choose a sequence of non-negative numbers ξ_i such that they satisfy a condition given in Javanmard and Montanari (2018), example 3.8.

The procedure depends on constants w_0 and b_0, where $w_0 \geq 0$ represents the intial 'wealth' and $b_0 > 0$ represents the 'payout' for rejecting a hypothesis. We require $w_0 + b_0 \leq \alpha$ for FDR control to hold.

Further details of the modified LORD procedure can be found in Javanmard and Montanari (2018).

Value

`d.out` A dataframe with the original data `d` (which will be reordered if there are batches and `random = TRUE`), the LORD-adjusted significance thresholds α_i and the indicator function of discoveries `R`. Hypothesis i is rejected if the i-th p-value is less than or equal to α_i, in which case `R[i] = 1` (otherwise `R[i] = 0`).

References

Javanmard, A. and Montanari, A. (2018) Online Rules for Control of False Discovery Rate and False Discovery Exceedance. *Annals of Statistics*, 46(2):526-554.

`LORDstar` *LORDstar: Asynchronous online mFDR control based on recent discovery*

Description

Implements LORD algorithms for asynchronous online testing, as presented by Zrnic et al. (2021).

Usage

```
LORDstar(
  d,
  alpha = 0.05,
  version,
  gammai,
  w0,
  batch.sizes,
  display_progress = FALSE
)
```

Arguments

d	Either a vector of p-values, or a dataframe with three columns: an identifier ('id'), p-value ('pval'), and either 'decision.times', or 'lags', depending on which version you're using. See version for more details.
alpha	Overall significance level of the procedure, the default is 0.05.
version	Takes values 'async', 'dep' or 'batch'. This specifies the version of LORDstar to use. `version='async'` requires a column of decision times ('decision.times'). `version='dep'` requires a column of lags ('lags'). `version='batch'` requires a vector of batch sizes ('batch.sizes').
gammai	Optional vector of γ_i. A default is provided as proposed by Javanmard and Montanari (2018), equation 31.
w0	Initial 'wealth' of the procedure, defaults to $\alpha/10$.
batch.sizes	A vector of batch sizes, this is required for `version='batch'`.
display_progress	Logical. If `TRUE` prints out a progress bar for the algorithm runtime.

Details

The function takes as its input either a vector of p-values, or a dataframe with three columns: an identifier ('id'), p-value ('pval'), and a column describing the conflict sets for the hypotheses. This takes the form of a vector of decision times or lags. Batch sizes can be specified as a separate argument (see below).

Zrnic et al. (2021) present explicit three versions of LORDstar:

1) `version='async'` is for an asynchronous testing process, consisting of tests that start and finish at (potentially) random times. The discretised finish times of the test correspond to the decision times. These decision times are given as the input `decision.times` for this version of the LORDstar algorithm.

2) `version='dep'` is for online testing under local dependence of the p-values. More precisely, for any $t > 0$ we allow the p-value p_t to have arbitrary dependence on the previous L_t p-values. The fixed sequence L_t is referred to as 'lags', and is given as the input `lags` for this version of the LORDstar algorithm.

3) `version='batch'` is for controlling the mFDR in mini-batch testing, where a mini-batch represents a grouping of tests run asynchronously which result in dependent p-values. Once a mini-batch of tests is fully completed, a new one can start, testing hypotheses independent of the previous batch. The batch sizes are given as the input `batch.sizes` for this version of the LORDstar algorithm.

Given an overall significance level α, LORDstar depends on w_0 (where $0 \leq w_0 \leq \alpha$), which represents the intial 'wealth' of the procedure. The algorithms also require a sequence of non-negative non-increasing numbers γ_i that sum to 1.

Note that these LORDstar algorithms control the *modified* FDR (mFDR). The 'async' version also controls the usual FDR if the p-values are assumed to be independent.

Further details of the LORDstar algorithms can be found in Zrnic et al. (2021).

Value

`out` A dataframe with the original p-values `pval`, the adjusted testing levels α_i and the indicator function of discoveries `R`. Hypothesis i is rejected if the i-th p-value is less than or equal to α_i, in which case `R[i] = 1` (otherwise `R[i] = 0`).

References

Javanmard, A. and Montanari, A. (2018) Online Rules for Control of False Discovery Rate and False Discovery Exceedance. *Annals of Statistics*, 46(2):526-554.

Zrnic, T., Ramdas, A. and Jordan, M.I. (2021). Asynchronous Online Testing of Multiple Hypotheses. *Journal of Machine Learning Research* 22:1-33.

See Also

LORD presents versions of LORD for *synchronous* p-values, i.e. where each test can only start when the previous test has finished.

Examples

```
sample.df <- data.frame(
id = c('A15432', 'B90969', 'C18705', 'B49731', 'E99902',
    'C38292', 'A30619', 'D46627', 'E29198', 'A41418',
    'D51456', 'C88669', 'E03673', 'A63155', 'B66033'),
pval = c(2.90e-08, 0.06743, 0.01514, 0.08174, 0.00171,
        3.60e-05, 0.79149, 0.27201, 0.28295, 7.59e-08,
        0.69274, 0.30443, 0.00136, 0.72342, 0.54757),
decision.times = seq_len(15) + 1)

LORDstar(sample.df, version='async')

sample.df2 <- data.frame(
id = c('A15432', 'B90969', 'C18705', 'B49731', 'E99902',
    'C38292', 'A30619', 'D46627', 'E29198', 'A41418',
    'D51456', 'C88669', 'E03673', 'A63155', 'B66033'),
pval = c(2.90e-08, 0.06743, 0.01514, 0.08174, 0.00171,
        3.60e-05, 0.79149, 0.27201, 0.28295, 7.59e-08,
        0.69274, 0.30443, 0.00136, 0.72342, 0.54757),
lags = rep(1,15))

LORDstar(sample.df2, version='dep')
```

---

onlineFDR-deprecated *Deprecated functions in package 'onlineFDR'*

Description

These functions are provided for compatibility with older versions of 'onlineFDR' only, and will be defunct at the next release.

Details

The following functions are deprecated and will be made defunct; use the replacement indicated below:

- LORDdep: LORD with version='dep'
- bonfInfinite: Alpha_spending

`online_fallback` *Online fallback procedure for FWER control*

Description

Implements the online fallback procedure of Tian and Ramdas (2021), which guarantees strong FWER control under arbitrary dependence of the p-values.

Usage

```
online_fallback(
  d,
  alpha = 0.05,
  gammai,
  random = TRUE,
  display_progress = FALSE,
  date.format = "%Y-%m-%d"
)
```

Arguments

`d`	Either a vector of p-values, or a dataframe with three columns: an identifier ('id'), date ('date') and p-value ('pval'). If no column of dates is provided, then the p-values are treated as being ordered in sequence, arriving one at a time.
`alpha`	Overall significance level of the FDR procedure, the default is 0.05.
`gammai`	Optional vector of γ_i. A default is provided as proposed by Javanmard and Montanari (2018), equation 31.
`random`	Logical. If `TRUE` (the default), then the order of the p-values in each batch (i.e. those that have exactly the same date) is randomised.
`display_progress`	Logical. If `TRUE` prints out a progress bar for the algorithm runtime.
`date.format`	Optional string giving the format that is used for dates.

Details

The function takes as its input either a vector of p-values or a dataframe with three columns: an identifier ('id'), date ('date') and p-value ('pval'). The case where p-values arrive in batches corresponds to multiple instances of the same date. If no column of dates is provided, then the p-values are treated as being ordered in sequence, arriving one at a time. Given an overall significance level α, we choose a sequence of non-negative non-increasing numbers γ_i that sum to 1.

The online fallback procedure provides a uniformly more powerful method than Alpha-spending, by saving the significance level of a previous rejection. More specifically, the procedure tests hypothesis H_i at level

$$\alpha_i = \alpha\gamma_i + R_{i-1}\alpha_{i-1}$$

where $R_i = 1\{p_i \leq \alpha_i\}$ denotes a rejected hypothesis.

Further details of the online fallback procedure can be found in Tian and Ramdas (2021).

Value

`out` A dataframe with the original data `d` (which will be reordered if there are batches and `random = TRUE`), the LORD-adjusted significance thresholds α_i and the indicator function of discoveries `R`. Hypothesis i is rejected if the i-th p-value is less than or equal to α_i, in which case `R[i] = 1` (otherwise `R[i] = 0`).

References

Tian, J. and Ramdas, A. (2021). Online control of the familywise error rate. *Statistical Methods for Medical Research*, 30(4):976–993.

Examples

```
sample.df <- data.frame(
id = c('A15432', 'B90969', 'C18705', 'B49731', 'E99902',
    'C38292', 'A30619', 'D46627', 'E29198', 'A41418',
    'D51456', 'C88669', 'E03673', 'A63155', 'B66033'),
date = as.Date(c(rep('2014-12-01',3),
                 rep('2015-09-21',5),
                  rep('2016-05-19',2),
                  '2016-11-12',
                 rep('2017-03-27',4))),
pval = c(2.90e-08, 0.06743, 0.01514, 0.08174, 0.00171,
        3.60e-05, 0.79149, 0.27201, 0.28295, 7.59e-08,
        0.69274, 0.30443, 0.00136, 0.72342, 0.54757))

online_fallback(sample.df, random=FALSE)

set.seed(1); online_fallback(sample.df)

set.seed(1); online_fallback(sample.df, alpha=0.1)
```

SAFFRON *SAFFRON: Adaptive online FDR control*

Description

Implements the SAFFRON procedure for online FDR control, where SAFFRON stands for Serial estimate of the Alpha Fraction that is Futilely Rationed On true Null hypotheses, as presented by Ramdas et al. (2018). The algorithm is based on an estimate of the proportion of true null hypotheses. More precisely, SAFFRON sets the adjusted test levels based on an estimate of the amount of alpha-wealth that is allocated to testing the true null hypotheses.

Usage

```
SAFFRON(
  d,
  alpha = 0.05,
  gammai,
  w0,
  lambda = 0.5,
  random = TRUE,
  display_progress = FALSE,
  date.format = "%Y-%m-%d"
)
```

Arguments

d	Either a vector of p-values, or a dataframe with three columns: an identifier ('id'), date ('date') and p-value ('pval'). If no column of dates is provided, then the p-values are treated as being ordered in sequence, arriving one at a time.
alpha	Overall significance level of the FDR procedure, the default is 0.05.
gammai	Optional vector of γ_i. A default is provided with γ_j proportional to $1/j^(1.6)$.
w0	Initial 'wealth' of the procedure, defaults to $\alpha/2$. Must be between 0 and α.
lambda	Optional threshold for a 'candidate' hypothesis, must be between 0 and 1. Defaults to 0.5.
random	Logical. If TRUE (the default), then the order of the p-values in each batch (i.e. those that have exactly the same date) is randomised.
display_progress	Logical. If TRUE prints out a progress bar for the algorithm runtime.
date.format	Optional string giving the format that is used for dates.

Details

The function takes as its input either a vector of p-values or a dataframe with three columns: an identifier ('id'), date ('date') and p-value ('pval'). The case where p-values arrive in batches corresponds to multiple instances of the same date. If no column of dates is provided, then the p-values are treated as being ordered in sequence, arriving one at a time.

SAFFRON procedure provably controls FDR for independent p-values. Given an overall significance level α, we choose a sequence of non-negative non-increasing numbers γ_i that sum to 1.

SAFFRON depends on constants w_0 and λ, where w_0 satisfies $0 \leq w_0 \leq \alpha$ and represents the initial 'wealth' of the procedure, and $0 < \lambda < 1$ represents the threshold for a 'candidate' hypothesis. A 'candidate' refers to p-values smaller than λ, since SAFFRON will never reject a p-value larger than λ.

Note that FDR control also holds for the SAFFRON procedure if only the p-values corresponding to true nulls are mutually independent, and independent from the non-null p-values.

The SAFFRON procedure can lose power in the presence of conservative nulls, which can be compensated for by adaptively 'discarding' these p-values. This option is called by setting `discard=TRUE`, which is the same algorithm as ADDIS.

Further details of the SAFFRON procedure can be found in Ramdas et al. (2018).

Value

`out` A dataframe with the original data `d` (which will be reordered if there are batches and `random = TRUE`), the LORD-adjusted significance thresholds α_i and the indicator function of discoveries `R`. Hypothesis i is rejected if the i-th p-value is less than or equal to α_i, in which case `R[i] = 1` (otherwise `R[i] = 0`).

References

Ramdas, A., Zrnic, T., Wainwright M.J. and Jordan, M.I. (2018). SAFFRON: an adaptive algorithm for online control of the false discovery rate. *Proceedings of the 35th International Conference in Machine Learning*, 80:4286-4294.

See Also

`SAFFRONstar` presents versions of SAFFRON for *asynchronous* testing, i.e. where each hypothesis test can itself be a sequential process and the tests can overlap in time.

Examples

```
sample.df <- data.frame(
id = c('A15432', 'B90969', 'C18705', 'B49731', 'E99902',
    'C38292', 'A30619', 'D46627', 'E29198', 'A41418',
    'D51456', 'C88669', 'E03673', 'A63155', 'B66033'),
date = as.Date(c(rep('2014-12-01',3),
               rep('2015-09-21',5),
                rep('2016-05-19',2),
                '2016-11-12',
               rep('2017-03-27',4))),
pval = c(2.90e-08, 0.06743, 0.01514, 0.08174, 0.00171,
        3.60e-05, 0.79149, 0.27201, 0.28295, 7.59e-08,
        0.69274, 0.30443, 0.00136, 0.72342, 0.54757))

SAFFRON(sample.df, random=FALSE)

set.seed(1); SAFFRON(sample.df)
```

```
set.seed(1); SAFFRON(sample.df, alpha=0.1, w0=0.025)
```

SAFFRONstar *SAFFRONstar: Adaptive online mFDR control for asynchronous testing*

Description

Implements the SAFFRON algorithm for asynchronous online testing, as presented by Zrnic et al. (2021).

Usage

```
SAFFRONstar(
  d,
  alpha = 0.05,
  version,
  gammai,
  w0,
  lambda = 0.5,
  batch.sizes,
  display_progress = FALSE
)
```

Arguments

d	Either a vector of p-values, or a dataframe with three columns: an identifier ('id'), p-value ('pval'), and either decision.times', or 'lags', depending on which version you're using. See version for more details.
alpha	Overall significance level of the procedure, the default is 0.05.
version	Takes values 'async', 'dep' or 'batch'. This specifies the version of SAFFRONstar to use. `version='async'` requires a column of decision times ('decision.times'). `version='dep'` requires a column of lags ('lags'). `version='batch'` requires a vector of batch sizes ('batch.sizes').
gammai	Optional vector of γ_i. A default is provided with γ_j proportional to $1/j^(1.6)$.
w0	Initial 'wealth' of the procedure, defaults to $\alpha/10$.
lambda	Optional threshold for a 'candidate' hypothesis, must be between 0 and 1. Defaults to 0.5.
batch.sizes	A vector of batch sizes, this is required for `version='batch'`.
display_progress	Logical. If `TRUE` prints out a progress bar for the algorithm runtime.

Details

The function takes as its input either a vector of p-values, or a dataframe with three columns: an identifier ('id'), p-value ('pval'), or a column describing the conflict sets for the hypotheses. This takes the form of a vector of decision times or lags. Batch sizes can be specified as a separate argument (see below).

Zrnic et al. (2021) present explicit three versions of SAFFRONstar:

1) `version='async'` is for an asynchronous testing process, consisting of tests that start and finish at (potentially) random times. The discretised finish times of the test correspond to the decision times. These decision times are given as the input `decision.times` for this version of the SAFFRONstar algorithm. For this version of SAFFRONstar, Tian and Ramdas (2019) presented an algorithm that can improve the power of the procedure in the presence of conservative nulls by adaptively 'discarding' these p-values. This can be called by setting the option `discard=TRUE`.

2) `version='dep'` is for online testing under local dependence of the p-values. More precisely, for any $t > 0$ we allow the p-value p_t to have arbitrary dependence on the previous L_t p-values. The fixed sequence L_t is referred to as 'lags', and is given as the input `lags` for this version of the SAFFRONstar algorithm.

3) `version='batch'` is for controlling the mFDR in mini-batch testing, where a mini-batch represents a grouping of tests run asynchronously which result in dependent p-values. Once a mini-batch of tests is fully completed, a new one can start, testing hypotheses independent of the previous batch. The batch sizes are given as the input `batch.sizes` for this version of the SAFFRONstar algorithm.

Given an overall significance level α, SAFFRONstar depends on constants w_0 and λ, where w_0 satisfies $0 \leq w_0 \leq \alpha$ and represents the intial 'wealth' of the procedure, and $0 < \lambda < 1$ represents the threshold for a 'candidate' hypothesis. A 'candidate' refers to p-values smaller than λ, since SAFFRONstar will never reject a p-value larger than λ. The algorithms also require a sequence of non-negative non-increasing numbers γ_i that sum to 1.

Note that these SAFFRONstar algorithms control the *modified* FDR (mFDR). The 'async' version also controls the usual FDR if the p-values are assumed to be independent.

Further details of the SAFFRONstar algorithms can be found in Zrnic et al. (2021).

Value

`out` A dataframe with the original p-values `pval`, the adjusted testing levels α_i and the indicator function of discoveries `R`. Hypothesis i is rejected if the i-th p-value is less than or equal to α_i, in which case `R[i] = 1` (otherwise `R[i] = 0`).

References

Zrnic, T., Ramdas, A. and Jordan, M.I. (2021). Asynchronous Online Testing of Multiple Hypotheses. *Journal of Machine Learning Research*, 22:1-33.

See Also

`SAFFRON` presents versions of SAFFRON for *synchronous* p-values, i.e. where each test can only start when the previous test has finished.

Examples

```
sample.df <- data.frame(
id = c('A15432', 'B90969', 'C18705', 'B49731', 'E99902',
    'C38292', 'A30619', 'D46627', 'E29198', 'A41418',
    'D51456', 'C88669', 'E03673', 'A63155', 'B66033'),
pval = c(2.90e-08, 0.06743, 0.01514, 0.08174, 0.00171,
        3.60e-05, 0.79149, 0.27201, 0.28295, 7.59e-08,
        0.69274, 0.30443, 0.00136, 0.72342, 0.54757),
decision.times = seq_len(15) + 1)

SAFFRONstar(sample.df, version='async')

sample.df2 <- data.frame(
id = c('A15432', 'B90969', 'C18705', 'B49731', 'E99902',
    'C38292', 'A30619', 'D46627', 'E29198', 'A41418',
    'D51456', 'C88669', 'E03673', 'A63155', 'B66033'),
pval = c(2.90e-08, 0.06743, 0.01514, 0.08174, 0.00171,
        3.60e-05, 0.79149, 0.27201, 0.28295, 7.59e-08,
        0.69274, 0.30443, 0.00136, 0.72342, 0.54757),
lags = rep(1,15))

SAFFRONstar(sample.df2, version='dep')
```

---

setBound *setBound*

Description

Calculates a default sequence of non-negative numbers γ_i that sum to 1, given an upper bound N on the number of hypotheses to be tested.

Usage

```
setBound(alg, alpha = 0.05, N)
```

Arguments

alg	A string that takes the value of one of the following: LOND, LORD, LORDdep, SAFFRON, ADDIS, LONDstar, LORDstar, SAFFRONstar, or Alpha_investing
alpha	Overall significance level of the FDR procedure, the default is 0.05. The bounds for LOND and LORDdep depend on alpha.
N	An upper bound on the number of hypotheses to be tested

Value

bound	A vector giving the values of a default sequence γ_i of nonnegative numbers.

`StoreyBH` *StoreyBH: Offline FDR control using the St-BH procedure*

Description

Implements the Storey-BH algorithm for offline FDR control, as presented by Storey (2002).

Usage

```
StoreyBH(d, alpha = 0.05, lambda = 0.5)
```

Arguments

`d`	Either a vector of p-values, or a dataframe with the column: p-value ('pval').
`alpha`	Overall significance level of the FDR procedure, the default is 0.05.
`lambda`	Threshold for Storey-BH, must be between 0 and 1. Defaults to 0.5.

Details

The function takes as its input either a vector of p-values, or a dataframe with a column of p-values ('pval').

Value

`ordered_d`	A dataframe with the original data `d` and the indicator function of discoveries `R`. Hypothesis i is rejected if the i-th p-value is less than or equal to $(r/n)\alpha$, where r is the rank of the i-th p-value within an ordered set and n is the total number of hypotheses. If hypothesis i is rejected, `R[i] = 1` (otherwise `R[i] = 0`).

References

Storey, J.D. (2002). A direct approach to false discovery rates. *J. R. Statist. Soc. B*: 64, Part 3, 479-498.

Examples

```
pvals <- c(2.90e-08, 0.06743, 0.01514, 0.08174, 0.00171,
        3.60e-05, 0.79149, 0.27201, 0.28295, 7.59e-08,
        0.69274, 0.30443, 0.00136, 0.72342, 0.54757)

StoreyBH(pvals)
```

supLORD *supLORD: Online control of the false discovery exceedance (FDX) and the FDR at stopping times*

Description

Implements the supLORD procedure, which controls both FDX and FDR, including the FDR at stopping times, as presented by Xu and Ramdas (2021).

Usage

```
supLORD(
  d,
  delta = 0.05,
  eps,
  r,
  eta,
  rho,
  gammai,
  random = TRUE,
  display_progress = FALSE,
  date.format = "%Y-%m-%d"
)
```

Arguments

d	Either a vector of p-values, or a dataframe with three columns: an identifier ('id'), date ('date') and p-value ('pval'). If no column of dates is provided, then the p-values are treated as being ordered in sequence, arriving one at a time.
delta	The probability at which the FDP exceeds eps (at any time step after making r rejections). Must be between 0 and 1, defaults to 0.05.
eps	The upper bound on the FDP. Must be between 0 and 1.
r	The threshold of rejections after which the error control begins to apply. Must be a positive integer.
eta	Controls the pace at which wealth is spent as a function of the algorithm's current wealth. Must be a positive real number.
rho	Controls the length of time before the spending sequence exhausts the wealth earned from a rejection. Must be a positive integer.
gammai	Optional vector of γ_i. A default is provided as proposed by Javanmard and Montanari (2018).
random	Logical. If TRUE (the default), then the order of the p-values in each batch (i.e. those that have exactly the same date) is randomised.
display_progress	Logical. If TRUE prints out a progress bar for the algorithm runtime.
date.format	Optional string giving the format that is used for dates.

Details

The function takes as its input either a vector of p-values or a dataframe with three columns: an identifier ('id'), date ('date') and p-value ('pval'). The case where p-values arrive in batches corresponds to multiple instances of the same date. If no column of dates is provided, then the p-values are treated as being ordered in sequence, arriving one at a time..

The supLORD procedure provably controls the FDX for p-values that are conditionally superuniform under the null. supLORD also controls the supFDR and hence the FDR (even at stopping times). Given an overall significance level α, we choose a sequence of non-negative non-increasing numbers γ_i that sum to 1.

supLORD requires the user to specify r, a threshold of rejections after which the error control begins to apply, eps, the upper bound on the false discovery proportion (FDP), and delta, the probability at which the FDP exceeds eps at any time step after making r rejections. As well, the user should specify the variables eta, which controls the pace at which wealth is spent (as a function of the algorithm's current wealth), and rho, which controls the length of time before the spending sequence exhausts the wealth earned from a rejection.

Further details of the supLORD procedure can be found in Xu and Ramdas (2021).

Value

`d.out` A dataframe with the original data `d` (which will be reordered if there are batches and `random = TRUE`), the supLORD-adjusted significance thresholds α_i and the indicator function of discoveries `R`. Hypothesis i is rejected if the i-th p-value is less than or equal to α_i, in which case `R[i] = 1` (otherwise `R[i] = 0`).

References

Xu, Z. and Ramdas, A. (2021). Dynamic Algorithms for Online Multiple Testing. *Annual Conference on Mathematical and Scientific Machine Learning*, PMLR, 145:955-986.

Examples

```
set.seed(1)
N <- 1000
B <- rbinom(N, 1, 0.5)
Z <- rnorm(N, mean = 3*B)
pval <- pnorm(-Z)

out <- supLORD(pval, eps=0.15, r=30, eta=0.05, rho=30, random=FALSE)
head(out)
sum(out$R)
```

Index