Package ‘IHWpaper’

August 15, 2024

Title  Reproduce figures in IHW paper

Version  1.32.0

Description  This package conveniently wraps all functions needed to reproduce the figures in the IHW paper (https://www.nature.com/articles/nmeth.3885) and the data analysis in https://rss.onlinelibrary.wiley.com/doi/10.1111/rssb.12411, cf. the arXiv preprint (http://arxiv.org/abs/1701.05179). Thus it is a companion package to the Bioconductor IHW package.

Depends  R (>= 3.3), IHW

License  Artistic-2.0

LazyData  true

LinkingTo  Rcpp

Imports  Rcpp, stats, splines, methods, utils, DESeq2, SummarizedExperiment, fdrtool, genefilter, qvalue, Biobase, BiocGenerics, BiocParallel, dplyr, grid, ggrepplot2, cowplot

VignetteBuilder  knitr

biocViews  ReproducibleResearch, ExperimentData, RNASeqData, ExpressionData

RoxygenNote  7.1.2

Suggests  testthat, RColorBrewer, wesanderson, scales, gridExtra, BiocStyle, knitr, rmarkdown, airway, locfdr, tidyr, latex2exp

NeedsCompilation  yes

Author  Nikos Ignatiadis [aut, cre]

Maintainer  Nikos Ignatiadis <nikos.ignatiadis01@gmail.com>

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

git_branch  RELEASE_3_19

git_last_commit  f16ee69

git_last_commit_date  2024-04-30

Repository  Bioconductor 3.19

Date/Publication  2024-08-15
analyze_dataset

analyze_dataset: Basically performs preprocessing and then returns analyzed RNASeq dataset (diff. expression), i.e. the DESeq2 result whose p-values and baseMean statistics can then be used with IHW

Description

analyze_dataset: Basically performs preprocessing and then returns analyzed RNASeq dataset (diff. expression), i.e. the DESeq2 result whose p-values and baseMean statistics can then be used with IHW

Usage

analyze_dataset(dataset = c("airway", "bottomly", "hammer"), res = TRUE)

Arguments

dataset Character, name of dataset to be preprocessed, only 3 choices currently available ("airway", "bottomly", "hammer")

res (default TRUE): return result table, rather than DESeq2 object

Value

Preprocessed dataset
Examples

```r
bottomly <- analyze_dataset("bottomly")
```

---

**Description**

`bh`: Wrapper for Benjamini Hochberg

**Usage**

```r
bh(unadj_p, alpha)
```

**Arguments**

- `unadj_p`: Numeric vector of unadjusted p-values.
- `alpha`: Significance level at which to apply method

**Value**

BH multiple testing object

**Examples**

```r
sim_df <- du_ttest_sim(20000, 0.95, 1.5)
obj <- bh(sim_df$pvalue, .1)
sum(rejected_hypotheses(obj))
```

---

**Description**

`bonf`: Wrapper for Bonferroni

**Usage**

```r
bonf(unadj_p, alpha)
```

**Arguments**

- `unadj_p`: Numeric vector of unadjusted p-values.
- `alpha`: Significance level at which to apply method
Value

Bonferroni multiple testing object

Examples

```r
sim_df <- du_ttest_sim(20000, 0.95, 1.5)
obj <- bonf(sim_df$pvalue, .1)
sum(rejected_hypotheses(obj))
```

clfdr

clfdr: Cai's local fdr based method

Description

clfdr: Cai's local fdr based method

Usage

clfdr(unadj_p, groups, alpha, lfdr_estimation = "fdrtool")

Arguments

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>unadj_p</td>
<td>Numeric vector of unadjusted p-values.</td>
</tr>
<tr>
<td>groups</td>
<td>Factor to which different hypotheses belong</td>
</tr>
<tr>
<td>alpha</td>
<td>Significance level at which to apply method</td>
</tr>
<tr>
<td>lfdr_estimation</td>
<td>Method used to estimate the local fdr, defaults to &quot;fdrtool&quot;</td>
</tr>
</tbody>
</table>

Value

Clfdr multiple testing object

References


Examples

```r
sim_df <- du_ttest_sim(20000, 0.95, 1.5)
sim_df$group <- groups_by_filter(sim_df$filterstat, 20)
obj <- clfdr(sim_df$pvalue, sim_df$group, .1)
sum(rejected_hypotheses(obj))
```
### continuous_wrap

**Benchmarking wrapper:** Given a multiple testing method, convert it so that it takes a simulation object (see simulation function) and a nominal level alpha as inputs

#### Description

Benchmarking wrapper: Given a multiple testing method, convert it so that it takes a simulation object (see simulation function) and a nominal level alpha as inputs

#### Usage

```r
continuous_wrap(mt_method, nbins = 20)
```

#### Arguments

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>mt_method</td>
<td>Multiple testing method (e.g. a function such as gbh or ddhf)</td>
</tr>
<tr>
<td>nbins</td>
<td>Integer, number of equally sized bins into which to stratify hypotheses</td>
</tr>
</tbody>
</table>

#### Value

A new multiple testing function which has an interface of the form `f(sim_data_frame, alpha)`

#### Examples

```r
sim_df <- du_ttest_sim(20000, 0.95, 1.5)
sim_df$group <- groups_by_filter(sim_df$filterstat, 20)
obj <- tst_gbh(sim_df$pvalue, sim_df$group, .1)
sum(rejected_hypotheses(obj))

tst_gbh_continuous <- continuous_wrap(tst_gbh)
obj2 <- tst_gbh_continuous(sim_df, .1)
sum(rejected_hypotheses(obj2))
```

### ddhf

**ddhf:** Greedy independent filtering

#### Description

ddhf: Greedy independent filtering

#### Usage

```r
ddhf(unadj_p, filterstat, alpha)
```
**Arguments**

- `unadj_p`: Numeric vector of unadjusted p-values.
- `filterstat`: Factor to which different hypotheses belong
- `alpha`: Significance level at which to apply method

**Value**

DDHF multiple testing object

**Examples**

```r
sim_df <- du_ttest_sim(20000,0.95, 1.5)
obj <- ddhf(sim_df$pvalue, sim_df$filterstat, .1)
sum(rejected_hypotheses(obj))
```

**du_ttest_sim**

*t-test simulation: Simulate rowwise t-tests*

**Description**

t-test simulation: Simulate rowwise t-tests

**Usage**

```r
du_ttest_sim(
  m,
  pi0,
  effect_size,
  n_samples = 10,
  uninformative_filter = FALSE,
  seed = NULL
)
```

```r
du_ttest_sim_fun(
  m,
  pi0,
  effect_size,
  n_samples = 10,
  uninformative_filter = FALSE
)
```
Arguments

- **m**: Integer, total number of hypotheses
- **pi0**: Numeric, proportion of null hypotheses
- **effect_size**: Numeric, the alternative hypotheses will be
- **n_samples**: Integer, number of samples for t-test, i.e. the comparison will be n_samples/2 vs n_samples/2
- **uninformative_filter**: Boolean, if TRUE will generate uniformly distributed filter statistic Otherwise will use the pooled standard deviations
- **seed**: Integer, Random seed to be used for simulation (default: NULL, i.e. RNG state will be used as is)

Value

A data frame containing all information about the simulation experiment

Functions

- **du_ttest_sim_fun**: Creates a closure function for a given seed

Examples

```r
sim_df <- du_ttest_sim(20000, 0.95, 1.5)
```

Description

**gbh**: Grouped Benjamini Hochberg

**tst_gbh**: wrapper for **gbh** with method="TST"  
**lsl_gbh**: wrapper for **gbh** with method="LSL"

Usage

```r
gbh(unadj_p, groups, alpha, method = "TST", pi0_global = "weighted_average")
tst_gbh(unadj_p, groups, alpha, ...)
lsl_gbh(unadj_p, groups, alpha, ...)
```
### Arguments

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>unadj_p</td>
<td>Numeric vector of unadjusted p-values.</td>
</tr>
<tr>
<td>groups</td>
<td>Factor to which different hypotheses belong</td>
</tr>
<tr>
<td>alpha</td>
<td>Significance level at which to apply method</td>
</tr>
<tr>
<td>method</td>
<td>What pi0 estimator should be used (available &quot;TST&quot;,&quot;LSL&quot;)</td>
</tr>
<tr>
<td>pi0_global</td>
<td>GBH requires also a pi0 estimate for the marginal p-value distribution. Can either apply pi0 estimation method to all p-values (pi0_global=&quot;global&quot;) or use a weighted average (pi0_global=&quot;weighted_average&quot;) of the pi0 estimates within each stratum. This is not explicitly stated in the paper, but based on a reproduction of their paper figures it seems to be the weighted_average. Additional arguments passed from tst_gbh/lsl_gbh to gbh</td>
</tr>
</tbody>
</table>

### Value

GBH multiple testing object

### Functions

- `tst_gbh`: Wrapper of GBH with TST pi0 estimator
- `lsl_gbh`: Wrapper of GBH with LSL pi0 estimator

### References


### Examples

```r
sim_df <- du_ttest_sim(20000,0.95, 1.5)
sim_df$group <- groups_by_filter(sim_df$filterstat, 20)
obj <- tst_gbh(sim_df$pvalue, sim_df$group, .1)
sum(rejected_hypotheses(obj))
```
Usage

ihw_naive(unadj_p, filterstat, alpha)

ihw_ecdf_5fold(unadj_p, filterstat, alpha)

ihw_5fold(unadj_p, filterstat, alpha)

ihw_5fold_reg(unadj_p, filterstat, alpha)

ihw_bonf_5fold_reg(unadj_p, filterstat, alpha)

ihw_storey_5fold(unadj_p, filterstat, alpha)

Arguments

unadj_p Numeric vector of unadjusted p-values.

filterstat Factor to which different hypotheses belong

alpha Significance level at which to apply method

Details

These are closures, which apply IHW with custom prespecified parameters. These correspond to
interesting settings, for which it is convenient to be able to immediately call the corresponding func-
tions, rather than having to specify parameters each time. Thus they make it easier to benchmark.
All of these wrappers are defined in 2 lines of code, so the settings pertaining to each one can be
inspected by typing the functions name into the console.

Value

ihwResult multiple testing object

Functions

• ihw_naive: IHW naive
• ihw_ecdf_5fold: IHW (E2) with 5 folds
• ihw_5fold: IHW (E1-E2) with 5 folds
• ihw_5fold_reg: IHW (E1-E2-E3) with 5 folds
• ihw_bonf_5fold_reg: IHW-Bonferroni (E1-E2-E3) with 5 folds
• ihw_storey_5fold: IHW (E1-E2) with 5 folds and Storey’s pi0 estimator

Examples

sim_df <- du_ttest_sim(20000,0.95, 1.5)
obj <- ihw_5fold(sim_df$pvalue, sim_df$filterstat, .1)
sum(rejected_hypotheses(obj))
lsl_pi0_est  

*LSL (Least-Slope) pi0 estimator*

**Description**
LSL (Least-Slope) pi0 estimator

**Usage**
lsl_pi0_est(pvalue)

**Arguments**
- `pvalue` Numeric vector of unadjusted p-values.

**Value**
estimated proportion of null hypotheses (pi0)

**Examples**
sim_df <- du_ttest_sim(20000, 0.95, 1.5)
lsl_pi0_est(sim_df$pvalue)

null_sim  

*Null simulation: Generate uniformly distributed p-values and covariates*

**Description**
Null simulation: Generate uniformly distributed p-values and covariates

**Usage**
null_sim(m, seed = NULL)
null_sim_fun(m)

**Arguments**
- `m` Integer, total number of hypotheses
- `seed` Integer, Random seed to be used for simulation (default: NULL, i.e. RNG state will be used as is)
**pretty_legend**

**Value**

A data frame containing all information about the simulation experiment

**Functions**

- **null_sim_fun**: Creates a closure function for a given seed

**Examples**

```r
sim_df <- null_sim(20000)
```

---

**pretty_legend**

*helper function to create nice legends*

**Description**

helper function to create nice legends

**Usage**

`pretty_legend(gg, last_vals, xmin, fontsize = 13)`

**Arguments**

- `gg`: ggplot2 object
- `last_vals`: data frame with columns `label`, `colour`, `last_vals` (i.e. place label with colour at y-coordinate `last_vals`)
- `xmin`: Numeric, x axis position at which labels should be placed
- `fontsize`: Integer, fontsize

**Value**

Another ggplot2 object

This replaces the default legend of a ggplot2 object. In particular, given a ggplot2 object, it removes the existing legend and then places new labels based on the annotation data frame `last_vals` (see parameter description) at a given x-coordinate of the original plot.

This function can be attributed to and is described in more detail in the following blog post: [http://www.r-bloggers.com/coloring-and-drawing-outside-the-lines-in-ggplot/](http://www.r-bloggers.com/coloring-and-drawing-outside-the-lines-in-ggplot/)
Examples

```r
library("ggplot2")
labels <- c("A","B","C")
mypoints <- rbind(data.frame(y=1:3, x=1, label=as.factor(labels)),
                   data.frame(y=2:4, x=2, label=as.factor(labels)))
mycolours <- c("#F8766D", "#00BA38", "#619CFF")
gg <- ggplot(mypoints,aes(x=x,y=y,color=label)) +
         geom_line(size=2) +
         scale_color_manual(values=mycolours) +
         xlim(c(0,2.2))
gg
annotation_df <- data.frame(colour=mycolours, last_vals=2:4, label=labels)
pretty_legend(gg, annotation_df, 2.1)
```

---

**run_evals**

**run_evals**: Main function to benchmark FDR methods on given simulations.

**Description**

run_evals: Main function to benchmark FDR methods on given simulations.

**Usage**

```r
run_evals(sim_funs, fdr_methods, nreps, alphas, ...)
```

**Arguments**

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>sim_funs</td>
<td>List of simulation settings</td>
</tr>
<tr>
<td>fdr_methods</td>
<td>List of FDR controlling methods to be benchmarked</td>
</tr>
<tr>
<td>nreps</td>
<td>Integer, number of Monte Carlo replicates for the simulations</td>
</tr>
<tr>
<td>alphas</td>
<td>Numeric, vector of nominal significance levels at which to apply FDR controlling methods</td>
</tr>
<tr>
<td>...</td>
<td>Additional arguments passed to sim_fun_eval</td>
</tr>
</tbody>
</table>

**Details**

This is the main workhorse function which runs all simulation benchmarks for IHWpaper. It receives input as described above, and the output is a data.frame with the following columns:

- **fdr_method**: Multiple testing method which was used
- **fdr_pars**: Custom parameters of the multiple testing method
- **alpha**: Nominal significance level at which the benchmark was run
- **FDR**: False Discovery Rate of benchmarked method on simulated dataset
- **power**: Power of benchmarked method on simulated dataset
• rj_ratio: Average rejections divided by total number of hypotheses
• FPR: False positive rate of benchmarked method on simulated dataset
• FWER: Familywise Error Rate of benchmarked method on simulated dataset
• nsuccessful: Number of successful evaluations of the method
• sim_method: Simulation scenario under which benchmark was run
• m: Total number of hypotheses
• sim_pars: Custom parameters of the simulation scenario

Value
data.frame which summarizes results of numerical experiment

Examples
nreps <- 3 # monte carlo replicates
ms <- 5000 # number of hypothesis tests
eff_sizes <- c(2,3)
sim_funs <- lapply(eff_sizes,
  function(x) du_ttest_sim_fun(ms,0.95,x, uninformative_filter=FALSE))
continuous_methods_list <- list(bh,
  lsl_gbh,
  clfdr,
  ddhf)
fdr_methods <- lapply(continuous_methods_list, continuous_wrap)
eval_table <- run_evals(sim_funs, fdr_methods, nreps, 0.1, BiocParallel=FALSE)

scott_fdrreg


Description

Usage
scott_fdrreg(unadj_p, filterstat, alpha, df = 3, lambda = 0.01)

Arguments
unadj_p Numeric vector of unadjusted p-values.
filterstat Factor to which different hypotheses belong
alpha Significance level at which to apply method
df Degrees of freedom for B-slines
lambda Ridge regularization parameter
storey_qvalue

Value

FDRreg multiple testing object

References


storey_qvalue: Wrapper for Storey's qvalue package

Description

storey_qvalue: Wrapper for Storey's qvalue package

Usage

storey_qvalue(unadj_p, alpha)

Arguments

unadj_p Numeric vector of unadjusted p-values.
alpha Significance level at which to apply method

Value

StoreyQValue multiple testing object

Examples

sim_df <- du_ttest_sim(20000, 0.95, 1.5)
obj <- storey_qvalue(sim_df$pvalue, .1)
sum(rejected_hypotheses(obj))
stratified_bh

stratified_bh: Stratified Benjamini Hochberg

Description

stratified_bh: Stratified Benjamini Hochberg

Usage

stratified_bh(unadj_p, groups, alpha)

Arguments

unadj_p Numeric vector of unadjusted p-values.
groups Factor to which different hypotheses belong
alpha Significance level at which to apply method

Value

SBH multiple testing object

References


Examples

```r
sim_df <- du_ttest_sim(20000,0.95, 1.5)
sim_df$group <- groups_by_filter(sim_df$filterstat, 20)
obj <- stratified_bh(sim_df$pvalue, sim_df$group, .1)
sum(rejected_hypotheses(obj))
```
tst_pi0_est  
*TST (Two-Step) pi0 estimator*

**Description**

TST (Two-Step) pi0 estimator

**Usage**

tst_pi0_est(pvalue, alpha)

**Arguments**

- `pvalue` : Numeric vector of unadjusted p-values.
- `alpha` : Nominal level for applying the TST procedure

**Value**

Estimated proportion of null hypotheses (pi0)

**Examples**

```r
sim_df <- du_ttest_sim(20000,0.95, 1.5)
tst_pi0_est(sim_df$pvalue, .1)
```

---

`wasserman_normal_prds_sim`

*Normal PRDS simulation: Covariate is effect size under alternative, there are latent factors driving PRDS correlations among hypotheses*

**Description**

Normal PRDS simulation: Covariate is effect size under alternative, there are latent factors driving PRDS correlations among hypotheses

**Usage**

```r
wasserman_normal_prds_sim(
m, pi0, rho = 0, latent_factors = 1, xi_min = 0, xi_max = 2.5, seed = NULL
```
Normal simulation: Covariate is effect size under alternative

Description
Normal simulation: Covariate is effect size under alternative

Usage
wasserman_normal_sim(m, pi0, xi_min, xi_max, seed = NULL)
wasserman_normal_sim_fun(m, pi0, xi_min, xi_max)
Arguments

- \( m \) Integer, total number of hypotheses
- \( \pi_0 \) Numeric, proportion of null hypotheses
- \( \xi_{\text{min}}, \xi_{\text{max}} \) Numeric, covariates are drawn as uniform on \( \xi_{\text{min}}, \xi_{\text{max}} \)
- seed Integer, Random seed to be used for simulation (default: NULL, i.e. RNG state will be used as is)

Value

A data frame containing all information about the simulation experiment

Functions

- \( \text{wasserman\_normal\_sim\_fun} \): Creates a closure function for a given seed

Examples

\[
\text{sim\_df} \leftarrow \text{wasserman\_normal\_sim}(20000, 0.9, 1, 5)
\]
Index

<table>
<thead>
<tr>
<th>Function</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>analyze_dataset</td>
<td>2</td>
</tr>
<tr>
<td>bh</td>
<td>3</td>
</tr>
<tr>
<td>bonf</td>
<td>3</td>
</tr>
<tr>
<td>clfdr</td>
<td>4</td>
</tr>
<tr>
<td>continuous_wrap</td>
<td>5</td>
</tr>
<tr>
<td>ddhf</td>
<td>5</td>
</tr>
<tr>
<td>du_ttest_sim</td>
<td>6</td>
</tr>
<tr>
<td>du_ttest_sim_fun(du_ttest_sim)</td>
<td>6</td>
</tr>
<tr>
<td>gbh</td>
<td>7</td>
</tr>
<tr>
<td>ihw_5fold(ihw_naive)</td>
<td>8</td>
</tr>
<tr>
<td>ihw_5fold_reg(ihw_naive)</td>
<td>8</td>
</tr>
<tr>
<td>ihw_bonf_5fold_reg(ihw_naive)</td>
<td>8</td>
</tr>
<tr>
<td>ihw_ecdf_5fold(ihw_naive)</td>
<td>8</td>
</tr>
<tr>
<td>ihw_naive</td>
<td>8</td>
</tr>
<tr>
<td>ihw_storey_5fold(ihw_naive)</td>
<td>8</td>
</tr>
<tr>
<td>lsl_gbh(gbh)</td>
<td>7</td>
</tr>
<tr>
<td>lsl_pi0_est</td>
<td>10</td>
</tr>
<tr>
<td>null_sim</td>
<td>10</td>
</tr>
<tr>
<td>null_sim_fun(null_sim)</td>
<td>10</td>
</tr>
<tr>
<td>pretty_legend</td>
<td>11</td>
</tr>
<tr>
<td>run_evals</td>
<td>12</td>
</tr>
<tr>
<td>scott_fdrreg</td>
<td>13</td>
</tr>
<tr>
<td>storey_qvalue</td>
<td>14</td>
</tr>
<tr>
<td>stratified_bh</td>
<td>15</td>
</tr>
<tr>
<td>tst_gbh(gbh)</td>
<td>7</td>
</tr>
<tr>
<td>tst_pi0_est</td>
<td>16</td>
</tr>
<tr>
<td>wasserman_normal_prds_sim</td>
<td>16</td>
</tr>
<tr>
<td>wasserman_normal_prds_sim_fun</td>
<td>16</td>
</tr>
<tr>
<td>wasserman_normal_sim</td>
<td>17</td>
</tr>
<tr>
<td>wasserman_normal_sim_fun</td>
<td>17</td>
</tr>
</tbody>
</table>