Package ‘IHWpaper’

March 23, 2017

**Title** Reproduce figures in IHW paper

**Version** 1.2.0

**Description** This package conveniently wraps all functions needed to reproduce the figures in the IHW preprint, available on biorXiv under http://biorxiv.org/content/early/2016/03/30/034330. 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, ggplot2, cowplot

**VignetteBuilder** knitr

**biocViews** ReproducibleResearch, ExperimentData, RNASeqData, ExpressionData

**RoxygenNote** 5.0.1

**Suggests** testthat, RColorBrewer, wesanderson, scales, gridExtra, BiocStyle, knitr, rmardown, airway, pasilla, DESeq, locfdr, tidy

**NeedsCompilation** yes

**Author** Nikos Ignatiadis [aut, cre]

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

**R topics documented:**

- analyze_dataset
- bh
- bonf
- clfdr
- continuous_wrap
- ddhf
- du_ttest_sim
- gbh
- ihw_naive
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 DDHW

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 DDHW

Usage

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

Arguments

dataset Character, name of dataset to be preprocessed, only 4 choices currently available
res (default TRUE): return result table, rather than DESeq2 object

Value

Preprocessed dataset

Examples

pasilla <- analyze_dataset("pasilla")
bh: Wrapper for Benjamini Hochberg

Description
bh: Wrapper for Benjamini Hochberg

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

bonf: Wrapper for Bonferroni

Description
bonf: Wrapper for Bonferroni

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

Description

clfdr: Cai’s local fdr based method

Usage

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

Arguments

unadj_p Numeric vector of unadjusted p-values.
groups Factor to which different hypotheses belong
alpha Significance level at which to apply method
lfdr_estimation Method used to estimate the local fdr, defaults to “fdrtool”

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))
```

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**

- **mt_method**: Multiple testing method (e.g. a function such as gbh or ddfh)
- **nbins**: Integer, number of equally sized bins into which to stratify hypotheses

**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))
```

---

**Description**

ddhf: Greedy independent filtering

**Usage**

`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 <- ddfh(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

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

`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)
```
**gbh**

*gbh: Grouped Benjamini Hochberg*

**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**

- `unadj_p`: Numeric vector of unadjusted p-values.
- `groups`: Factor to which different hypotheses belong
- `alpha`: Significance level at which to apply method
- `method`: What pi0 estimator should be used (available "TST","LSL")
- `pi0_global`: GBH requires also a pi0 estimate for the marginal p-value distribution. Can either apply pi0 estimation method to all p-values (pi0_global="global") or use a weighted average (pi0_global="weighted_average") 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

**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))
```
**ihw naive**

**IHW wrappers**

**Description**
IHW wrappers

**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)`

**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 functions, 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

**Examples**

```r
sim_df <- du_ttest_sim(20000, 0.95, 1.5)
obj <- ihw_5fold(sim_df$pvalue, sim_df$filterstat, .1)
sum(rejected_hypotheses(obj))
```
**lsli0_est**

LSL (Least-Slope) pi0 estimator

**Usage**

```r
lsli0_est(pvalue)
```

**Arguments**

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

**Value**

estimated proportion of null hypotheses (pi0)

**Examples**

```r
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

**Usage**

```r
null_sim(m, seed = NULL)
```

```r
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)

**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

```r
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:  

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))

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

Arguments
- sim_funs: List of simulation settings
- fdr_methods: List of FDR controlling methods to be benchmarked
- nreps: Integer, number of Monte Carlo replicates for the simulations
- alphas: Numeric, vector of nominal significance levels at which to apply FDR controlling methods
- ...: Additional arguments passed to sim_fun_eval

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

```r
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)
```

**Description**

**scott_fdrreg**: Wrapper for FDR regression (https://github.com/jgscott/FDRreg)

**Usage**

```r
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.

**Value**

FDRreg multiple testing object.

**References**

storey_qvalue

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

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

Usage

\[
\text{wasserman_normal_prds_sim}(m, \pi_0, \rho = 0, \text{latent_factors} = 1, \text{xi_min} = 0, \text{xi_max} = 2.5, \text{seed} = \text{NULL}) \\
\text{wasserman_normal_prds_sim_fun}(m, \pi_0, \rho = 0, \text{latent_factors} = 1, \text{xi_min} = 0, \text{xi_max} = 2.5)
\]

Arguments

- \(m\): Integer, total number of hypotheses
- \(\pi_0\): Numeric, proportion of null hypotheses
- \(\rho\): Numeric, correlation between z-scores of hypotheses driven by same latent factor
- \(\text{latent_factors}\): Integer, number of latent factors driving the correlations
- \(\text{xi_min}, \text{xi_max}\): Numeric, covariates are drawn as uniform on \(\text{xi_min}, \text{xi_max}\)
- \(\text{seed}\): 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_prds_sim_fun}: Creates a closure function for a given seed

Examples

\[
sim\_df <- \text{wasserman_normal_prds_sim}(20000, 0.9, \rho=0.1)
\]
wasserman_normal_sim

Normal simulation: Covariate is effect size under alternative

### Description

Normal simulation: Covariate is effect size under alternative

### Usage

```r
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
- `pi0` Numeric, proportion of null hypotheses
- `xi_min, xi_max` Numeric, covariates are drawn as uniform on `xi_min, xi_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

- `wasserman_normal_sim_fun`: Creates a closure function for a given seed

### Examples

```r
sim_df <- wasserman_normal_sim(20000, 0.9, 1, 5)
```
Index

analyze_dataset, 2
bh, 3
bonf, 3
c1fdr, 4
continuous_wrap, 4
ddhf, 5
du_ttest_sim, 6
du_ttest_sim_fun (du_ttest_sim), 6
gbh, 7
ihw_5fold (ihw_naive), 8
ihw_5fold_reg (ihw_naive), 8
ihw_bonf_5fold_reg (ihw_naive), 8
ihw_ecdf_5fold (ihw_naive), 8
ihw_naive, 8
lsl_gbh (gbh), 7
lsl_pi0_est, 9
null_sim, 9
null_sim_fun (null_sim), 9
pretty_legend, 10
run_evals, 11
scott_fdrreg, 12
storey_qvalue, 13
stratified_bh, 13
tst_gbh (gbh), 7
tst_pi0_est, 14
wasserman_normal_prds_sim, 15
wasserman_normal_prds_sim_fun
  (wasserman_normal_prds_sim), 15
wasserman_normal_sim, 16
wasserman_normal_sim_fun
  (wasserman_normal_sim), 16