

# Package: IHWpaper (via r-universe)

October 10, 2026

**Title** Reproduce figures in IHW paper

**Version** 1.41.0

**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, ggplot2, 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](mailto:nikos.ignatiadis01@gmail.com)>

**Config/pak/sysreqs** coinor-symphony coinor-libsymphony-dev libicu-dev libpng-dev libxml2-dev libssl-dev zlib1g-dev

**Repository** <https://bioc.r-universe.dev>

**Date/Publication** 2026-04-28 12:29:51 UTC

**RemoteUrl** <https://github.com/bioc/IHWpaper>

**RemoteRef** HEAD

**RemoteSha** 4458da47a181a8f755b45d2dbd83aecded67724c

Contents

|                                     |           |
|-------------------------------------|-----------|
| analyze_dataset . . . . .           | 2         |
| bh . . . . .                        | 3         |
| bonf . . . . .                      | 3         |
| clfdr . . . . .                     | 4         |
| continuous_wrap . . . . .           | 5         |
| ddhf . . . . .                      | 5         |
| du_ttest_sim . . . . .              | 6         |
| gbh . . . . .                       | 7         |
| ihw_naive . . . . .                 | 8         |
| lsl_pi0_est . . . . .               | 10        |
| null_sim . . . . .                  | 10        |
| pretty_legend . . . . .             | 11        |
| run_evals . . . . .                 | 12        |
| scott_fdrreg . . . . .              | 13        |
| storey_qvalue . . . . .             | 14        |
| stratified_bh . . . . .             | 15        |
| tst_pi0_est . . . . .               | 16        |
| wasserman_normal_prds_sim . . . . . | 16        |
| wasserman_normal_sim . . . . .      | 17        |
| <b>Index</b>                        | <b>19</b> |

---

|                 |                                                                                                                                                                                                                |
|-----------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| analyze_dataset | <i>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</i> |
|-----------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|

---

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**

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

---

|    |                                           |
|----|-------------------------------------------|
| bh | <i>bh: Wrapper for Benjamini Hochberg</i> |
|----|-------------------------------------------|

---

**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 | <i>bonf: Wrapper for Bonferroni</i> |
|------|-------------------------------------|

---

**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

*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**

|                 |                                                              |
|-----------------|--------------------------------------------------------------|
| 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**

Cai, T. Tony, and Wenguang Sun. "Simultaneous testing of grouped hypotheses: Finding needles in multiple haystacks." *Journal of the American Statistical Association* 104.488 (2009).

**Examples**

```
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 | <i>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</i> |
|-----------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------|

---

### 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

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

### Arguments

|           |                                                                         |
|-----------|-------------------------------------------------------------------------|
| mt_method | Multiple testing method (e.g. a function such as gbh or ddhf)           |
| 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

```
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 | <i>ddhf: Greedy independent filtering</i> |
|------|-------------------------------------------|

---

### 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**

```
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**

```
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**

|                                   |                                                                                                                         |
|-----------------------------------|-------------------------------------------------------------------------------------------------------------------------|
| <code>m</code>                    | Integer, total number of hypotheses                                                                                     |
| <code>pi0</code>                  | Numeric, proportion of null hypotheses                                                                                  |
| <code>effect_size</code>          | Numeric, the alternative hypotheses will be                                                                             |
| <code>n_samples</code>            | Integer, number of samples for t-test, i.e. the comparison will be $n\_samples/2$ vs $n\_samples/2$                     |
| <code>uninformative_filter</code> | Boolean, if TRUE will generate uniformly distributed filter statistic Otherwise will use the pooled standard deviations |
| <code>seed</code>                 | 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**

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

---

|                  |                                        |
|------------------|----------------------------------------|
| <code>gbh</code> | <i>gbh: Grouped Benjamini Hochberg</i> |
|------------------|----------------------------------------|

---

**Description**

`gbh`: Grouped Benjamini Hochberg

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

**Usage**

```
gbh(unadj_p, groups, alpha, method = "TST", pi0_global = "weighted_average")
```

```
tst_gbh(unadj_p, groups, alpha, ...)
```

```
lsl_gbh(unadj_p, groups, alpha, ...)
```

**Arguments**

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

**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**

Hu, James X., Hongyu Zhao, and Harrison H. Zhou. "False discovery rate control with groups." *Journal of the American Statistical Association* 105.491 (2010).

**Examples**

```
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)

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 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
- `ihw_storey_5fold`: IHW (E1-E2) with 5 folds and Storey's  $\pi_0$  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 | <i>LSL (Least-Slope) pi0 estimator</i> |
|-------------|----------------------------------------|

---

**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 | <i>Null simulation: Generate uniformly distributed p-values and covariates</i> |
|----------|--------------------------------------------------------------------------------|

---

**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)

**Value**

A data frame containing all information about the simulation experiment

**Functions**

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

**Examples**

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

---

|                            |                                               |
|----------------------------|-----------------------------------------------|
| <code>pretty_legend</code> | <i>helper function to create nice legends</i> |
|----------------------------|-----------------------------------------------|

---

**Description**

helper function to create nice legends

**Usage**

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

**Arguments**

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

## Examples

```
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

```
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**

```
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 | <i>scott_fdrreg:</i>                                                                       | <i>Wrapper</i> | <i>for</i> | <i>FDR</i> | <i>regression</i> |
|              | <i>(<a href="https://github.com/jgscott/FDRreg">https://github.com/jgscott/FDRreg</a>)</i> |                |            |            |                   |

---

**Description**

`scott_fdrreg`: Wrapper for FDR regression (<https://github.com/jgscott/FDRreg>)

**Usage**

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

**Arguments**

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

**Value**

FDRreg multiple testing object

**References**

James G. Scott, Ryan C. Kelly, Matthew A. Smith, Pengcheng Zhou, and Robert E. Kass. "False discovery rate regression: application to neural synchrony detection in primary visual cortex." Journal of the American Statistical Association (2015).

---

|               |                                                           |
|---------------|-----------------------------------------------------------|
| storey_qvalue | <i>storey_qvalue: Wrapper for Storey's qvalue package</i> |
|---------------|-----------------------------------------------------------|

---

**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 | <i>stratified_bh: Stratified Benjamini Hochberg</i> |
|---------------|-----------------------------------------------------|

---

## 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

Sun, Lei, et al. "Stratified false discovery control for large-scale hypothesis testing with application to genome-wide association studies." Genetic epidemiology 30.6 (2006): 519-530.

Yoo, Yun J., et al. "Were genome-wide linkage studies a waste of time? Exploiting candidate regions within genome-wide association studies." Genetic epidemiology 34.2 (2010): 107-118.

## Examples

```
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 | <i>TST (Two-Step) pi0 estimator</i> |
|-------------|-------------------------------------|

---

**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**

```
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**

```
wasserman_normal_prds_sim(
  m,
  pi0,
  rho = 0,
  latent_factors = 1,
  xi_min = 0,
  xi_max = 2.5,
  seed = NULL)
```

```

)

wasserman_normal_prds_sim_fun(
  m,
  pi0,
  rho = 0,
  latent_factors = 1,
  xi_min = 0,
  xi_max = 2.5
)

```

### Arguments

|                |                                                                                                   |
|----------------|---------------------------------------------------------------------------------------------------|
| m              | Integer, total number of hypotheses                                                               |
| pi0            | Numeric, proportion of null hypotheses                                                            |
| rho            | Numeric, correlation between z-scores of hypotheses driven by same latent factor                  |
| latent_factors | Integer, number of latent factors driving the correlations                                        |
| 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\_prds\_sim\_fun: Creates a closure function for a given seed

### Examples

```
sim_df <- 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

```

wasserman_normal_sim(m, pi0, xi_min, xi_max, seed = NULL)

wasserman_normal_sim_fun(m, pi0, xi_min, xi_max)

```

**Arguments**

|                             |                                                                                                   |
|-----------------------------|---------------------------------------------------------------------------------------------------|
| <code>m</code>              | Integer, total number of hypotheses                                                               |
| <code>pi0</code>            | Numeric, proportion of null hypotheses                                                            |
| <code>xi_min, xi_max</code> | Numeric, covariates are drawn as uniform on <code>xi_min</code> , <code>xi_max</code>             |
| <code>seed</code>           | 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**

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

# Index

analyze\_dataset, 2

bh, 3

bonf, 3

clfdr, 4

continuous\_wrap, 5

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

ihw\_storey\_5fold (ihw\_naive), 8

lsl\_gbh (gbh), 7

lsl\_pi0\_est, 10

null\_sim, 10

null\_sim\_fun (null\_sim), 10

pretty\_legend, 11

run\_evals, 12

scott\_fdrreg, 13

storey\_qvalue, 14

stratified\_bh, 15

tst\_gbh (gbh), 7

tst\_pi0\_est, 16

wasserman\_normal\_prds\_sim, 16

wasserman\_normal\_prds\_sim\_fun  
    (wasserman\_normal\_prds\_sim), 16

wasserman\_normal\_sim, 17

wasserman\_normal\_sim\_fun  
    (wasserman\_normal\_sim), 17