Package 'IHW'

November 10, 2026

Title Independent Hypothesis Weighting

Version 1.41.0

Description Independent hypothesis weighting (IHW) is a multiple testing procedure that increases power compared to the method of Benjamini and Hochberg by assigning data-driven weights to each hypothesis. The input to IHW is a two-column table of p-values and covariates. The covariate can be any continuous-valued or categorical variable that is thought to be informative on the statistical properties of each hypothesis test, while it is independent of the p-value under the null hypothesis.

Depends R (>= 3.3.0)

License Artistic-2.0

LazyData true

Imports methods, slam, lpsymphony, fdrtool, BiocGenerics

Suggests ggplot2, dplyr, gridExtra, scales, DESeq2, airway, testthat, Matrix, BiocStyle, knitr, rmarkdown, devtools

VignetteBuilder knitr

biocViews ImmunoOncology, MultipleComparison, RNASeq

RoxygenNote 7.1.1

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

git_branch devel

git_last_commit ea4f335

git_last_commit_date 2026-04-28

Repository Bioconductor 3.24

Date/Publication 2026-11-09

Author Nikos Ignatiadis [aut, cre], Wolfgang Huber [aut]

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

Contents

get_bh_threshold 2
groups_by_filter 2
ihw.default 3
ihw.DESeqResults 5
ihwResult-class 6
plot,ihwResult-method 10

Index 12

`get_bh_threshold` *Data-driven threshold of Benjamini Hochberg Procedure*

Description

Given pvalues and a nominal significance level alpha, this function returns the rejection threshold of the Benjamini-Hochberg procedure, i.e. a value t_BH such that p-values with P_i <= t_BH get rejected by the procedure.

Usage

```
get_bh_threshold(pvals, alpha, mtests = length(pvals))
```

Arguments

`pvals`	Numeric, vector of p-values
`alpha`	Numeric in [0,1], significance level of the multiple testing procedure
`mtests`	Integer, total number of hypothesis tests; only set this (to non-default) when you know what you are doing!

Value

A numeric in [0,1], threshold of the BH procedure

Examples

```
pvalues <- c(runif(1000), rbeta(1000,0.5,7)) # generate some p-values
adj_pvalues <- p.adjust(pvalues, method="BH") # calculate adjusted p-values
t_BH  <- get_bh_threshold(pvalues, 0.1) #get rejection threshold at alpha=0.1
all((pvalues <= t_BH) == (adj_pvalues <= 0.1)) #equivalence of two formulations
```

`groups_by_filter` *Stratify hypotheses based on increasing value of the covariate*

Description

Hypotheses are stratified into nbins different strata of (approximately) equal size based on increasing value of the covariate

Usage

```
groups_by_filter(covariate, nbins, ties.method = "random", seed = NULL)
```

Arguments

covariate	Numeric vector of ordinal covariates based on which the stratification will be done.
nbins	Integer, number of groups/strata into which p-values will be split based on covariate.
ties.method	Character specifying how ties are treated, see rank function.
seed	Integer, specifies random seed to be used when ties.method=="random".

Value

A factor with nbins different levels, each entry corresponds to the stratum the i-th hypothesis was assigned to.

Examples

```
covariates <- runif(100)
groups <- groups_by_filter(covariates,10)
table(groups)
```

---

ihw.default *ihw: Main function for Independent Hypothesis Weighting*

Description

Given a vector of p-values, a vector of covariates which are independent of the p-values under the null hypothesis and a nominal significance level alpha, IHW learns multiple testing weights and then applies the weighted Benjamini Hochberg (or Bonferroni) procedure.

Usage

```
## Default S3 method:
ihw(
  pvalues,
  covariates,
  alpha,
  covariate_type = "ordinal",
  nbins = "auto",
  m_groups = NULL,
  folds = NULL,
  quiet = TRUE,
  nfolds = 5L,
  nfolds_internal = 5L,
  nsplits_internal = 1L,
  lambdas = "auto",
  seed = 1L,
  distrib_estimator = "grenander",
  lp_solver = "lpsymphony",
  adjustment_type = "BH",
  null_proportion = FALSE,
  null_proportion_level = 0.5,
```

```
  return_internal = FALSE,
  ...
)

## S3 method for class 'formula'
ihw(formula, data = parent.frame(), ...)
```

Arguments

pvalues	Numeric vector of unadjusted p-values.
covariates	Vector which contains the one-dimensional covariates (independent under the H0 of the p-value) for each test. Can be numeric or a factor. (If numeric it will be converted into factor by binning.)
alpha	Numeric, sets the nominal level for FDR control.
covariate_type	"ordinal" or "nominal" (i.e. whether covariates can be sorted in increasing order or not)
nbins	Integer, number of groups into which p-values will be split based on covariate. Use "auto" for automatic selection of the number of bins. Only applicable when covariates is not a factor.
m_groups	Integer vector of length equal to the number of levels of the covariates (only to be specified when the latter is a factor/categorical). Each entry corresponds to the number of hypotheses to be tested in each group (stratum). This argument needs to be given when the complete vector of p-values is not available, but only p-values below a given threshold, for example because of memory reasons. See the vignette for additional details and an example of how this principle can be applied with numerical covariates.
folds	Integer vector or NULL. Pre-specify assignment of hypotheses into folds.
quiet	Boolean, if False a lot of messages are printed during the fitting stages.
nfolds	Number of folds into which the p-values will be split for the pre-validation procedure
nfolds_internal	Within each fold, a second (nested) layer of cross-validation can be conducted to choose a good regularization parameter. This parameter controls the number of nested folds.
nsplits_internal	Integer, how many times to repeat the nfolds_internal splitting. Can lead to better regularization parameter selection but makes ihw a lot slower.
lambdas	Numeric vector which defines the grid of possible regularization parameters. Use "auto" for automatic selection.
seed	Integer or NULL. Split of hypotheses into folds is done randomly. To have the output of the function be reproducible, the seed of the random number generator is set to this value at the start of the function. Use NULL if you don't want to set the seed.
distrib_estimator	Character ("grenander" or "ECDF"). Only use this if you know what you are doing. ECDF with nfolds > 1 or lp_solver == "lpsymphony" will in general be excessively slow, except for very small problems.
lp_solver	Character ("lpsymphony" or "gurobi"). Internally, IHW solves a sequence of linear programs, which can be solved with either of these solvers.

`adjustment_type`
: Character ("BH" or "bonferroni") depending on whether you want to control FDR or FWER.

`null_proportion`
: Boolean, if True (default is False), a modified version of Storey's estimator is used within each bin to estimate the proportion of null hypotheses.

`null_proportion_level`
: Numeric, threshold for Storey's pi0 estimation procedure, defaults to 0.5

`return_internal`
: Returns a lower level representation of the output (only useful for debugging purposes).

`...`
: Arguments passed to internal functions.

`formula`
: formula, specified in the form pvalue~covariate (only 1D covariate supported)

`data`
: data.frame from which the variables in formula should be taken

Value

A ihwResult object.

See Also

ihwResult, plot,ihwResult-method, ihw.DESeqResults

Examples

```
save.seed <- .Random.seed; set.seed(1)
X   <- runif(20000, min=0, max=2.5)   # covariate
H   <- rbinom(20000,1,0.1)            # hypothesis true or false
Z   <- rnorm(20000, H*X)              # Z-score
.Random.seed <- save.seed
pvalue <- 1-pnorm(Z)                  # pvalue

ihw_fdr <- ihw(pvalue, X, .1)        # Standard IHW for FDR control
ihw_fwer <- ihw(pvalue, X, .1, adjustment_type = "bonferroni")    # FWER control
table(H[adj_pvalues(ihw_fdr) <= 0.1] == 0) #how many false rejections?
table(H[adj_pvalues(ihw_fwer) <= 0.1] == 0)
```

`ihw.DESeqResults` *ihw.DESeqResults: IHW method dispatching on DESeqResults objects*

Description

ihw.DESeqResults: IHW method dispatching on DESeqResults objects

Usage

```
## S3 method for class 'DESeqResults'
ihw(deseq_res, filter = "baseMean", alpha = 0.1, adjustment_type = "BH", ...)
```

Arguments

deseq_res	"DESeqResults" object
filter	Vector of length equal to number of rows of deseq_res object. This is used for the covariates in the call to ihw. Can also be a character, in which case deseq_res[[filter]] is used as the covariate
alpha	Numeric, sets the nominal level for FDR control.
adjustment_type	Character ("BH" or "bonferroni") depending on whether you want to control FDR or FWER.
...	Other optional keyword arguments passed to ihw.

Value

A "DESeqResults" object, which includes weights and adjusted p-values returned by IHW. In addition, includes a metadata slot with an "ihwResult" object.

See Also

ihw, ihwResult

Examples

```
## Not run:
   library("DESeq2")
   library("airway")
   data("airway")
   dds <- DESeqDataSet(se = airway, design = ~ cell + dex)
   dds <- DESeq(dds)
   deseq_res <- results(dds)
   deseq_res <- ihw(deseq_res, alpha=0.1)
   #equivalent: deseq_res2 <- results(dds, filterFun = ihw)

## End(Not run)
```

---

ihwResult-class *An S4 class to represent the ihw output.*

Description

An S4 class to represent the ihw output.

Usage

```
adj_pvalues(object)

## S4 method for signature 'ihwResult'
adj_pvalues(object)

## S4 method for signature 'ihwResult'
weights(object, levels_only = FALSE)
```

```
thresholds(object, ...)

## S4 method for signature 'ihwResult'
thresholds(object, levels_only = FALSE)

pvalues(object)

## S4 method for signature 'ihwResult'
pvalues(object)

weighted_pvalues(object)

## S4 method for signature 'ihwResult'
weighted_pvalues(object)

covariates(object)

## S4 method for signature 'ihwResult'
covariates(object)

covariate_type(object)

## S4 method for signature 'ihwResult'
covariate_type(object)

groups_factor(object)

## S4 method for signature 'ihwResult'
groups_factor(object)

nfolds(object)

## S4 method for signature 'ihwResult'
nfolds(object)

nbins(object)

## S4 method for signature 'ihwResult'
nbins(object)

alpha(object)

## S4 method for signature 'ihwResult'
alpha(object)

rejections(object, ...)

## S4 method for signature 'ihwResult'
rejections(object)

rejected_hypotheses(object, ...)
```

```
## S4 method for signature 'ihwResult'
rejected_hypotheses(object)

regularization_term(object)

## S4 method for signature 'ihwResult'
regularization_term(object)

m_groups(object)

## S4 method for signature 'ihwResult'
m_groups(object)

as.data.frame_ihwResult(x, row.names = NULL, optional = FALSE, ...)

## S4 method for signature 'ihwResult'
as.data.frame(x, row.names = NULL, optional = FALSE, ...)

## S4 method for signature 'ihwResult'
nrow(x)

## S4 method for signature 'ihwResult'
show(object)
```

Arguments

object, x	A ihwResult object as returned by a call to ihw(...)
levels_only	Logical, if FALSE, return a vector of weights (thresholds) with one weight (threshold) for each hypothesis, otherwise return a nfolds x nbins matrix of weights (thresholds)
...	Parameters passed in to individual methods
row.names, optional	See ?base::as.data.frame for a description of these arguments.

Value

The different methods applied to an ihwResult object can return the following:

1) A vector of length equal to the number of hypotheses tested (e.g. the adjusted p-value or the weight of each hypothesis).

2) A matrix of dimension equal to nfolds x nbins (e.g. the weight of each stratum, fold combination, set by specifying levels_only=TRUE).

3) A vector of length 1 (usually a parameter of the ihwResult object such as nfolds or the total number of rejections).

4) A data.frame (as.data.frame) or just console output (show) for the extended Base generics.

See section below for the individual methods.

Methods (by generic)

- adj_pvalues: Extract adjusted pvalues
- weights: Extract weights

- thresholds: Calculate ihw thresholds
- pvalues: Extract pvalues
- weighted_pvalues: Extract weighted pvalues
- covariates: Extract covariates
- covariate_type: Extract type of covariate ("ordinal" or "nominal")
- groups_factor: Extract factor of stratification (grouping) variable
- nfolds: Extract number of folds
- nbins: Extract number of bins
- alpha: Extract nominal significance (alpha) level
- rejections: Total number of rejected hypotheses by ihw procedure
- rejected_hypotheses: Get a boolean vector of the rejected hypotheses
- regularization_term: Extract vector of regularization parameters used for each stratum
- m_groups: Extract total number of hypotheses within each stratum
- as.data.frame: Coerce ihwResult to data frame
- nrow: Return number of p-values
- show: Convenience method to show ihwResult object

Slots

`df` A data.frame that collects the input data, including the vector of p values and the covariate, the group assignment, as well as outputs (weighted p-values, adjusted p-values)

`weights` A (nbins X nfolds) matrix of the weight assigned to each stratum

`alpha` Numeric, the nominal significance level at which the FDR is to be controlled

`nbins` Integer, number of distinct levels into which the hypotheses were stratified

`nfolds` Integer, number of folds for pre-validation procedure

`regularization_term` Numeric vector, the final value of the regularization parameter within each fold

`m_groups` Integer vector, number of hypotheses tested in each stratum

`penalty` Character, "uniform deviation" or "total variation"

`covariate_type` Character, "ordinal" or "nominal"

`adjustment_type` Character, "BH" or "bonferroni"

`reg_path_information` A data.frame, information about the whole regularization path. (Currently not used, thus empty)

`solver_information` A list, solver specific output, e.g. were all subproblems solved to optimality? (Currently empty list)

See Also

ihw, plot,ihwResult-method

Examples

```
save.seed <- .Random.seed; set.seed(1)
X   <- runif(n = 20000, min = 0.5, max = 4.5)       # Covariate
# Is the null hypothesis (mean=0) true or false ?
H   <- rbinom(n = length(X), size = 1, prob = 0.1)
Z   <- rnorm(n = length(X), mean = H * X)           # Z-score
.Random.seed <- save.seed

pvalue <- 1 - pnorm(Z)                              # pvalue
ihw_res <- ihw(pvalue, covariates = X, alpha = 0.1)
rejections(ihw_res)
colnames(as.data.frame(ihw_res))
```

---

plot,ihwResult-method *Plot functions for IHW*

Description

See the vignette for usage examples.

Usage

```
## S4 method for signature 'ihwResult'
plot(
  x,
  x_axis = c(weights = "group", decisionboundary = "covariate")[what],
  what = "weights",
  scale = covariate_type(x)
)
```

Arguments

x	Object of class ihwResult
x_axis	Character: "group" or "covariate". Default is "group" if "what" is "weights", and "covariate" if "what" is "decisionboundary".
what	Character: "weights" or "decisionboundary"
scale	Character: "ordinal" or "nominal"

Value

A ggplot2 object.

Examples

```
save.seed <- .Random.seed; set.seed(1)
X   <- runif(20000, min = 0.5, max = 4.5) # covariate
H   <- rbinom(20000, 1, 0.1)              # hypothesis true or false
Z   <- rnorm(20000, H*X)                  # z-score
.Random.seed <- save.seed
pvalue <- 1-pnorm(Z)                      #pvalue
ihw_res <- ihw(pvalue, X, .1)
```

```
plot(ihw_res)
```

Index

adj_pvalues *(*ihwResult-class*)*, 6
adj_pvalues,ihwResult-method
*(*ihwResult-class*)*, 6
alpha *(*ihwResult-class*)*, 6
alpha,ihwResult-method
*(*ihwResult-class*)*, 6
as.data.frame,ihwResult-method
*(*ihwResult-class*)*, 6
as.data.frame_ihwResult
*(*ihwResult-class*)*, 6

covariate_type *(*ihwResult-class*)*, 6
covariate_type,ihwResult-method
*(*ihwResult-class*)*, 6
covariates *(*ihwResult-class*)*, 6
covariates,ihwResult-method
*(*ihwResult-class*)*, 6

formula, *5*

get_bh_threshold, 2
groups_by_filter, 2
groups_factor *(*ihwResult-class*)*, 6
groups_factor,ihwResult-method
*(*ihwResult-class*)*, 6

ihw *(*ihw.default*)*, 3
ihw.default, 3
ihw.DESeqResults, 5
ihwResult *(*ihwResult-class*)*, 6
ihwResult-class, 6

m_groups *(*ihwResult-class*)*, 6
m_groups,ihwResult-method
*(*ihwResult-class*)*, 6

nbins *(*ihwResult-class*)*, 6
nbins,ihwResult-method
*(*ihwResult-class*)*, 6
nfolds *(*ihwResult-class*)*, 6
nfolds,ihwResult-method
*(*ihwResult-class*)*, 6
nrow,ihwResult-method
*(*ihwResult-class*)*, 6

plot,ihwResult-method, 10
pvalues *(*ihwResult-class*)*, 6
pvalues,ihwResult-method
*(*ihwResult-class*)*, 6

rank, *3*
regularization_term *(*ihwResult-class*)*, 6
regularization_term,ihwResult-method
*(*ihwResult-class*)*, 6
rejected_hypotheses *(*ihwResult-class*)*, 6
rejected_hypotheses,ihwResult-method
*(*ihwResult-class*)*, 6
rejections *(*ihwResult-class*)*, 6
rejections,ihwResult-method
*(*ihwResult-class*)*, 6

show,ihwResult-method
*(*ihwResult-class*)*, 6

thresholds *(*ihwResult-class*)*, 6
thresholds,ihwResult-method
*(*ihwResult-class*)*, 6

weighted_pvalues *(*ihwResult-class*)*, 6
weighted_pvalues,ihwResult-method
*(*ihwResult-class*)*, 6
weights,ihwResult-method
*(*ihwResult-class*)*, 6