Package 'GLSUP'

April 21, 2026

Type Package

Title Generalised Linear Step-Up Procedure for Multiple Hypothesis Testing

Version 1.0.1

Maintainer Toby Kenney <tkenney@mathstat.dal.ca>

Description Performs the Generalised Linear Step-up Procedure (GLSUP) with a flexible user-defined sizing function. Functions are also available for creating common sizing functions.

License GPL-3

Encoding UTF-8

Imports graphics

NeedsCompilation no

Author Toby Kenney [aut, cre]

Repository CRAN

Date/Publication 2026-04-20 22:40:02 UTC

Contents

GLSUP *Generalised Linear Step-Up Procedure (GLSUP)*

Description

Performs variable selection, allowing the selection of sets of surrogates using the GLSUP method for FDR control. This allows selection of sets of variables from a hierarchical clustering of the predictors.

Usage

```
GLSUP(pvals,sizing,threshold)

## S3 method for class 'GLSUP'
print(x,...)
## S3 method for class 'GLSUP'
plot(x,...)
```

Arguments

pvals	The p-values of the hypothesis tests
sizing	The sizing function. Given a permutation of the hypotheses, this should calculate the sizing of every initial subset of the permutation.
threshold	The GLSUP rejects as many hypotheses as possible such that the p-values are all less than the sizing function of the rejected hypothesis divided by this threshold.
x	The GLSUP object to be printed or plotted.
...	Additional graphics or printing parameters. The graphics parameters are passed to other functions. For print.GLSUP, any additional parameters are ignored.

Details

The GLSUP method is a generalisation of the Benjamini-Hochberg (BH) or Benjamini-Yekutieli (BY) procedures, where the cardinality of a set of rejected hypotheses is replaced by a more general sizing function which quantifies the total discovery from rejecting a set of hypotheses, potentially incorporating different importance for different hypotheses, and overlap between the content of the hypotheses. GLSUP selects the largest possible subset of hypotheses such that the sizing function for the set of rejected hypotheses exceeds the largest p-value of any rejected hypothesis multiplied by the threshold.

Value

An object of class "GLSUP" which contains the following components "pv" The p-values sorted in ascending order "sizing" The sizing function applied to the smallest p-values "rejected" A logical vector indicating whether each hypothesis is rejected "pv.cut.off" The p-value cut-off below which hypotheses are rejected "cut.off.slope" The threshold parameter given to the function

Author(s)

Toby Kenney

References

Setwise Hierarchical Variable Selection and the Generalized Linear Step-Up Procedure for False Discovery Rate Control

Sarah Organ, Toby Kenney, Hong Gu

http://arxiv.org/abs/2603.02160

Examples

```
set.seed(1)
pv<-rbeta(31,1,5)
parents<-c(NA,rep(seq_len(15),each=2)) # perfect binary tree
weights<-2^-c(5,rep(4,2),rep(3,4),rep(2,8),rep(1,16))

sizing<-minimal.weights.forest(weights,parents)

ans<-GLSUP(pv,sizing,sum(weights)*20) # threshold under PRDS

print(ans)
plot(ans)
```

`Sizing Functions` *Cumulative sum-of-minimal-weights sizing function*

Description

Calculates the sum-of-minimal-weights sizing function for the initial elements of a hierarchical tree or a partially-ordered set.

Usage

```
minimal.weights.forest(weights,parents)
minimal.weights.poset(weights,poset)
sizing.BH(ord)
sizing.wBH(weights)
```

Arguments

`weights`	A vector of weights.
`parents`	A vector giving the index of the parent node for each node in the hierarchical tree or forest (NA for root nodes).
`poset`	An object of class poset, representing a partial order in the form of a list with two components: covers — list of lists of elements covered by each element. inc.order — a total order on the elements consistent with the partial order.
`ord`	A permutation

Details

Three of these functions generate sizing functions based on their parameters, for use with the GLSUP function. the sizing.BH function can be used directly as a sizing function, as it does not depend on any parameters. A sizing function is a function that "counts" the number of discoveries from rejecting any set of null hypotheses. The simplest sizing function is just the number of rejected hypotheses, as used by the Benjamini-Hochberg and Benjamini-Yekutieli methods. However, for some situations, this is not an appropriate way to count discoveries, and other sizing functions are preferable. The functions described provide the following sizing functions:

sizing.BH uses the cardinality as the sizing function.

sizing.wBH uses the sum of weights of selected elements as sizing function.

minimal.weights.poset uses the sum of weights of all minimal elements in the sub-partial order consisting of selected elements.

minimal.weights.forest is the same as minimal.weights.poset in the special case where the partial order is a hierarchical forest with root as top element in each tree. In this case, the sizing function can be made more computationally efficient.

Value

A sizing function that can be used with GLSUP. Sizing functions take a permutation as input and return the appropriate sizing function applied to the initial entries of the permutation.

Author(s)

Toby Kenney

References

Setwise Hierarchical Variable Selection and the Generalized Linear Step-Up Procedure for False Discovery Rate Control

Sarah Organ, Toby Kenney, Hong Gu

http://arxiv.org/abs/2603.02160

Examples

```
set.seed(1)
parents<-c(NA,rep(seq_len(15),each=2)) # perfect binary tree
weights<-2^-c(5,rep(4,2),rep(3,4),rep(2,8),rep(1,16)) #weighted by
                                                      #no. of leaves
sizing<-minimal.weights.forest(weights,parents)

sizing(seq_len(31)) # example of how to compute the sizing function.
### In practice, the sizing function is intended to be used with the
### GLSUP function.
```

Index