Package ‘hierinf’

May 29, 2024

<table>
<thead>
<tr>
<th>Type</th>
<th>Package</th>
</tr>
</thead>
<tbody>
<tr>
<td>Title</td>
<td>Hierarchical Inference</td>
</tr>
<tr>
<td>Version</td>
<td>1.22.0</td>
</tr>
<tr>
<td>Description</td>
<td>Tools to perform hierarchical inference for one or multiple studies / data sets based on high-dimensional multivariate (generalised) linear models. A possible application is to perform hierarchical inference for GWA studies to find significant groups or single SNPs (if the signal is strong) in a data-driven and automated procedure. The method is based on an efficient hierarchical multiple testing correction and controls the FWER. The functions can easily be run in parallel.</td>
</tr>
<tr>
<td>License</td>
<td>GPL-3</td>
</tr>
<tr>
<td>Encoding</td>
<td>UTF-8</td>
</tr>
<tr>
<td>LazyData</td>
<td>yes</td>
</tr>
<tr>
<td>Depends</td>
<td>R (&gt;= 3.6.0)</td>
</tr>
<tr>
<td>Imports</td>
<td>fmsb, glmnet, methods, parallel, stats</td>
</tr>
<tr>
<td>Suggests</td>
<td>knitr, MASS, testthat</td>
</tr>
<tr>
<td>biocViews</td>
<td>Clustering, GenomeWideAssociation, LinkageDisequilibrium, Regression, SNP</td>
</tr>
<tr>
<td>VignetteBuilder</td>
<td>knitr</td>
</tr>
<tr>
<td>RoxygenNote</td>
<td>6.1.1</td>
</tr>
<tr>
<td>git_url</td>
<td><a href="https://git.bioconductor.org/packages/hierinf">https://git.bioconductor.org/packages/hierinf</a></td>
</tr>
<tr>
<td>git_branch</td>
<td>RELEASE_3_19</td>
</tr>
<tr>
<td>git_last_commit</td>
<td>76f3d9b</td>
</tr>
<tr>
<td>git_last_commit_date</td>
<td>2024-04-30</td>
</tr>
<tr>
<td>Repository</td>
<td>Bioconductor 3.19</td>
</tr>
<tr>
<td>Date/Publication</td>
<td>2024-05-29</td>
</tr>
<tr>
<td>Author</td>
<td>Claude Renaux [aut, cre], Laura Buzdugan [aut], Markus Kalisch [aut], Peter Bühlmann [aut]</td>
</tr>
<tr>
<td>Maintainer</td>
<td>Claude Renaux <a href="mailto:renaux@stat.math.ethz.ch">renaux@stat.math.ethz.ch</a></td>
</tr>
</tbody>
</table>


class_cluster_position

## cluster_position

**Build Hierarchical Tree based on Position**

### Description

Build a hierarchical tree based on the position of the variables.

### Usage

```r
cluster_position(position, block = NULL, sort.parallel = TRUE, parallel = c("no", "multicore", "snow"), ncpus = 1L, cl = NULL)
```

### Arguments

- **position**: a data frame with two columns specifying the variable names and the corresponding position or a list of data frames for multiple data sets. The first column is required to contain the variable names and to be of type character. The second column is required to contain the position and to be of type numeric.

- **block**: a data frame or matrix specifying the second level of the hierarchical tree. The first column is required to contain the variable names and to be of type character. The second column is required to contain the group assignment and to be a vector of type character or numeric. If not supplied, the second level is built based on the data.

- **sort.parallel**: a logical indicating whether the values are sorted with respect to the size of the block. This can reduce the run time for parallel computation.

- **parallel**: type of parallel computation to be used. See the 'Details' section.

- **ncpus**: number of processes to be run in parallel.

- **cl**: an optional `parallel` or `snow` cluster used if `parallel = "snow"`. If not supplied, a cluster on the local machine is created.
cluster_position

Details

The hierarchical tree is built based on recursive binary partitioning of consecutive variables w.r.t.
their position. The partitioning consists of splitting a given node / cluster into two children of
about equal size based on the positions of the variables. If a node contains an odd number of
variables, then the variable in the middle w.r.t. position is assigned to the cluster containing the
closest neighbouring variable. Hence, clusters at a given depth of the binary hierarchical tree contain
about the same number of variables.

If the argument block is supplied, i.e. the second level of the hierarchical tree is given, the function
can be run in parallel across the different blocks by specifying the arguments parallel and ncpus.
There is an optional argument cl if parallel = "snow". There are three possibilities to set the arg-
ument parallel: parallel = "no" for serial evaluation (default), parallel = "multicore" for
parallel evaluation using forking, and parallel = "snow" for parallel evaluation using a parallel
socket cluster. It is recommended to select RNGkind("L'Ecuyer-CMRG") and set a seed to ensure
that the parallel computing of the package hierinf is reproducible. This way each processor gets a
different substream of the pseudo random number generator stream which makes the results repro-
ducible if the arguments (as sort.parallel and ncpus) remain unchanged. See the vignette or the
reference for more details.

Value

The returned value is an object of class "hierD", consisting of two elements, the argument "block"
and the hierarchical tree "res.tree".

The element "block" defines the second level of the hierarchical tree if supplied.

The element "res.tree" contains a dendrogram for each of the blocks defined in the argument
block. If the argument block is NULL (i.e. not supplied), the element contains only one dendrogram.

References

Renaux, C. et al. (2018), Hierarchical inference for genome-wide association studies: a view on
methodology with software. (arXiv:1805.02988)

See Also

cluster_var and test_hierarchy.

Examples

# The column names of the data frames position and block are optional.
position <- data.frame("var.name" = paste0("Var", 1:500),
  "position" = seq(from = 1, to = 1000, by = 2),
  stringsAsFactors = FALSE)
dendr1 <- cluster_position(position = position)

block <- data.frame("var.name" = paste0("Var", 1:500),
  "block" = rep(c(1, 2), each = 250),
  stringsAsFactors = FALSE)
dendr2 <- cluster_position(position = position, block = block)
cluster_var

Build Hierarchical Tree based on Hierarchical Clustering

Description

Build a hierarchical tree based on hierarchical clustering of the variables.

Usage

cluster_var(x = NULL, d = NULL, block = NULL, method = "average",
use = "pairwise.complete.obs", sort.parallel = TRUE,
parallel = c("no", "multicore", "snow"), ncpus = 1L, cl = NULL)

Arguments

x a matrix or list of matrices for multiple data sets. The matrix or matrices have to
be of type numeric and are required to have column names / variable names. The
rows and the columns represent the observations and the variables, respectively.
Either the argument x or d has to be specified.

d a dissimilarity matrix. This can be either a symmetric matrix of type numeric
with column and row names or an object of class dist with labels. Either the
argument x or d has to be specified.

block a data frame or matrix specifying the second level of the hierarchical tree. The
first column is required to contain the variable names and to be of type character.
The second column is required to contain the group assignment and to be a
vector of type character or numeric. If not supplied, the second level is built
based on the data.

method the agglomeration method to be used for the hierarchical clustering. See hclust
for details.

use the method to be used for computing covariances in the presence of missing
values. This is important for multiple data sets which do not measure exactly the
same variables. If data is specified using the argument x, the dissimilarity matrix
for the hierarchical clustering is calculated using correlation See the 'Details'
section and cor for all the options.

sort.parallel a logical indicating whether the values are sorted with respect to the size of the
block. This can reduce the run time for parallel computation.

parallel type of parallel computation to be used. See the 'Details' section.

ncpus number of processes to be run in parallel.

cl an optional parallel or snow cluster used if parallel = "snow". If not supplied,
a cluster on the local machine is created.
cluster_var

Details

The hierarchical tree is built by hierarchical clustering of the variables. Either the data (using the argument `x`) or a dissimilarity matrix (using the argument `d`) can be specified.

If one or multiple data sets are defined using the argument `x`, the dissimilarity matrix is calculated by one minus squared empirical correlation. In the case of multiple data sets, a single hierarchical tree is jointly estimated using hierarchical clustering. The argument `use` is important because missing values are introduced if the data sets do not measure exactly the same variables. The argument `use` determines how the empirical correlation is calculated.

Alternatively, it is possible to specify a user-defined dissimilarity matrix using the argument `d`.

If the argument `x` and `block` are supplied, i.e. the `block` defines the second level of the hierarchical tree, the function can be run in parallel across the different blocks by specifying the arguments `parallel` and `ncpus`. There is an optional argument `cl` if `parallel = "snow"`. There are three possibilities to set the argument `parallel`: `parallel = "no"` for serial evaluation (default), `parallel = "multicore"` for parallel evaluation using forking, and `parallel = "snow"` for parallel evaluation using a parallel socket cluster. It is recommended to select `RNGkind("L"Ecuyer-CMRG")` and set a seed to ensure that the parallel computing of the package `hierinf` is reproducible. This way each processor gets a different substream of the pseudo random number generator stream which makes the results reproducible if the arguments (as `sort.parallel` and `ncpus`) remain unchanged.

See the vignette or the reference for more details.

Value

The returned value is an object of class "hierD", consisting of two elements, the argument "block" and the hierarchical tree "res.tree".

The element "block" defines the second level of the hierarchical tree if supplied.

The element "res.tree" contains a `dendrogram` for each of the blocks defined in the argument `block`. If the argument `block` is `NULL` (i.e. not supplied), the element contains only one `dendrogram`.

References


See Also

`cluster_position` and `test_hierarchy`.

Examples

```r
library(MASS)
x <- mvrnorm(200, mu = rep(0, 500), Sigma = diag(500))
colnames(x) <- paste0("Var", 1:500)
dendr1 <- cluster_var(x = x)
# The column names of the data frame block are optional.
block <- data.frame("var.name" = paste0("Var", 1:500),
                    "block" = rep(1, 250),
                    stringsAsFactors = FALSE)
```

dendr2 <- cluster_var(x = x, block = block)

# The matrix x is first transposed because the function dist calculates
# distances between the rows.
d <- dist(t(x))
dendr3 <- cluster_var(d = d, method = "single")

---

**compute_r2**

**Compute R squared**

**Description**

Compute the R squared value for a given cluster or group of variables.

**Usage**

```r
compute_r2(x, y, res.test.hierarchy, clvar = NULL,
family = c("gaussian", "binomial"), colnames.cluster = NULL)
```

**Arguments**

- `x`: a matrix or list of matrices for multiple data sets. The matrix or matrices have to be of type numeric and are required to have column names / variable names. The rows and the columns represent the observations and the variables, respectively.
- `y`: a vector, a matrix with one column, or list of the aforementioned objects for multiple data sets. The vector, vectors, matrix, or matrices have to be of type numeric.
- `res.test.hierarchy`: the output of one of the functions `test_hierarchy`, `test_only_hierarchy`, or `multisplit`.
- `clvar`: a matrix or list of matrices of control variables.
- `family`: a character string naming a family of the error distribution; either "gaussian" or "binomial".
- `colnames.cluster`: The column names / variables names of the cluster of interest. If not supplied, the R squared value of the full model is computed.

**Details**

The R squared value is computed based on the output of the multi-sample splitting step. For each split, the intersection of the cluster / group (specified in `colnames.cluster`) and the selected variables is taken and R squared values are computed based on the second halves of observations. Finally, the R squared values are averaged over the B splits and over the different data sets if multiple data sets are supplied.
For a continuous response, the adjusted R squared values is calculated for a given cluster or group of variables. The Nagelkerke’s R squared values is computed for a binary response using the function \texttt{NagelkerkeR2}.

If \texttt{colnames.cluster} is not supplied, the R squared value of the full model is computed.

\textbf{Value}

The returned value is the R squared value.

\textbf{References}


\textbf{See Also}

\texttt{test_hierarchy}.

\textbf{Examples}

```r
n <- 200
p <- 500
library(MASS)
set.seed(3)
x <- mvrnorm(n, mu = rep(0, p), Sigma = diag(p))
colnames(x) <- paste0("Var", 1:p)
beta <- rep(0, p)
beta[c(5, 20, 46)] <- 1
y <- x %*% beta + rnorm(n)
dendr <- cluster_var(x = x)
set.seed(47)
sign.clusters <- test_hierarchy(x = x, y = y, dendr = dendr,
                               family = "gaussian")
compute_r2(x = x, y = y, res.test.hierarchy = sign.clusters,
           family = "gaussian",
           colnames.cluster = c("Var1", "Var5", "Var8"))
```

\textbf{Description}

The hierinf package provides the functions to perform hierarchical inference. The main workflow consists of two function calls.
Functions

The building of the hierarchical tree can be achieved by either of the functions `cluster_var` or `cluster_position`. The function `test_hierarchy` performs the hierarchical testing by going top down through the hierarchical tree and obviously requires the hierarchical tree as an input.

It is possible to calculate the R squared value of a given cluster using `compute_r2`. The hierarchical testing consists of two steps which can be evaluated separately if desired. Instead of calling `test_hierarchy`, the multi-sample splitting step is performed by `multisplit` and its output is used by the function `test_only_hierarchy` to test for significant clusters by going top down through the hierarchical tree.

---

**multisplit**

**Multi-sample splitting**

---

**Description**

The data is randomly split in two halves w.r.t. the observations and variable selection using Lasso is performed on one half. Whereas the second half and the selected variables are later used for testing by the function `test_only_hierarchy`. This is repeated multiple times.

**Usage**

```r
multisplit(x, y, clvar = NULL, B = 50, proportion.select = 1/6, 
standardize = FALSE, family = c("gaussian", "binomial"), 
parallel = c("no", "multicore", "snow"), ncpus = 1L, cl = NULL, 
check.input = TRUE)
```

**Arguments**

- **x**: a matrix or list of matrices for multiple data sets. The matrix or matrices have to be of type numeric and are required to have column names / variable names. The rows and the columns represent the observations and the variables, respectively.
- **y**: a vector, a matrix with one column, or list of the aforementioned objects for multiple data sets. The vector, vectors, matrix, or matrices have to be of type numeric. For `family = "binomial"`, the response is required to be a binary vector taking values 0 and 1.
- **clvar**: a matrix or list of matrices of control variables.
- **B**: number of sample splits.
- **proportion.select**: proportion of variables to be selected by Lasso in the multi-sample splitting step.
- **standardize**: a logical value indicating whether the variables should be standardized.
- **family**: a character string naming a family of the error distribution; either "gaussian" or "binomial".
- **parallel**: type of parallel computation to be used. See the 'Details' section.
- **ncpus**: number of processes to be run in parallel.
cl an optional parallel or snow cluster used if parallel = "snow". If not supplied, a cluster on the local machine is created.

check.input a logical value indicating whether the function should check the input. This argument is used to call multisplit within test_hierarchy.

Details

A given data with nobs is randomly split in two halves w.r.t. the observations and nobs * proportion.select variables are selected using Lasso (implemented in glmnet) on one half. Control variables are not penalized if supplied using the argument clvar. This is repeated B times for each data set if multiple data sets are supplied. Those splits (i.e. second halves of observations) and corresponding selected variables are used to perform hierarchical testing by the function test_only_hierarchy.

The multi-sample split step can be run in parallel across the different sample splits (B corresponds to number of sample splits) by specifying the arguments parallel and ncpus. There is an optional argument cl if parallel = "snow". There are three possibilities to set the argument parallel: parallel = "no" for serial evaluation (default), parallel = "multicore" for parallel evaluation using forking, and parallel = "snow" for parallel evaluation using a parallel socket cluster. It is recommended to select RNGkind("L'Ecuyer-CMRG") and set a seed to ensure that the parallel computing of the package hierinf is reproducible. This way each processor gets a different sub-stream of the pseudo random number generator stream which makes the results reproducible if the arguments (as sort.parallel and ncpus) remain unchanged. See the vignette or the reference for more details.

Value

The returned value is an object of class "hierM", consisting of a list with number of elements corresponding to the number of data sets. Each element (corresponding to a data set) contains the indices of the second half of variables (which were not used to select the variables). The second matrix contains the column names / variable names of the selected variables.

References


See Also

cluster_var, cluster_position, test_only_hierarchy, test_hierarchy, and compute_r2.

Examples

n <- 200
p <- 500
library(MASS)
set.seed(3)
x <- mvrnorm(n, mu = rep(0, p), Sigma = diag(p))
colnames(x) <- paste0("Var", 1:p)
beta <- rep(0, p)
beta[c(5, 20, 46)] <- 1
y <- x %*% beta + rnorm(n)

set.seed(84)
res.multisplit <- multisplit(x = x, y = y, family = "gaussian")

print.hierT

Description

Print significant clusters or groups of variables of an object of class hierT.

Usage

## S3 method for class 'hierT'
print(x, n.terms = 5L, digits = max(3,
   getOption("digits") - 3), right = FALSE, ...)

Arguments

x an object of class hierT
n.terms maximum number of column names or variables names to be printed per cluster or group of variables.
digits number of significant digits to be used.
right logical value indicating whether the values should or should not be right-aligned.
... additional arguments to print.data.frame

Details

The function prints the significant clusters or groups of variables of an object of class hierT. By default, it prints at most the first n.terms column or variable names per significant cluster and the number of omitted column names are printed in square brackets (if any).

Value

The returned values is a invisible copy of the object x.

References


See Also

invisible.
Examples

n <- 200
p <- 500
library(MASS)
set.seed(3)
x <- mvrnorm(n, mu = rep(0, p), Sigma = diag(p))
colnames(x) <- paste0("Var", 1:p)
beta <- rep(0, p)
beta[c(5, 20, 46)] <- 1
y <- x %*% beta + rnorm(n)
dendr <- cluster_var(x = x)
sign.clusters <- test_hierarchy(x = x, y = y, dendr = dendr,
                             family = "gaussian")

# The argument n.terms is useful if there is one or multiple
# significant groups containing many variables.
# print(sign.clusters, n.terms = 4)

print(sign.clusters, right = TRUE)
print(sign.clusters, digits = 4)

---

simGWAS Simulated GWAS data set

Description

The data set simGWAS was simulated using PLINK where the SNPs were binned into different allele frequency ranges. There are 250 controls and 250 cases, i.e. a binary response and 500 subjects. The variables age and sex are two additional control variables. The variables SNP.1 till SNP.990 were simulated to have no association with the response and the variables SNP.991 till SNP.1000 have a population odds ratio of 2.

Usage

data(simGWAS)

Format

A list with three elements:

  x  a matrix with 500 rows and 1000 columns where the rows and columns correspond to the subjects and variables, respectively. The variables are named SNP.1, ..., SNP.1000.
  y  binary response vector with 500 elements where the elements correspond to the subjects.
  clvar a matrix with 500 rows and 2 columns where the rows and columns correspond to the subjects and variables, respectively. The age of the subject is stored in the variable age. The variable sex takes the value 0 for men and 1 for women.
Source


Examples

data(simGWAS)

sim.geno <- simGWAS$x
sim.pheno <- simGWAS$y
sim.clvar <- simGWAS$clvar

dendr <- cluster_var(x = sim.geno)
set.seed(1234)
result <- test_hierarchy(x = sim.geno, y = sim.pheno,
                        dendr = dendr, clvar = sim.clvar,
                        family = "binomial")

---

test_hierarchy Hierarchical Testing

Description

Hierarchical testing based on multi-sample splitting.

Usage

test_hierarchy(x, y, dendr, clvar = NULL, family = c("gaussian", "binomial"), B = 50, proportion.select = 1/6, standardize = FALSE, alpha = 0.05, global.test = TRUE, agg.method = c("Tippett", "Stouffer"), verbose = FALSE, sort.parallel = TRUE, parallel = c("no", "multicore", "snow"), ncpus = 1L, cl = NULL)

Arguments

x a matrix or list of matrices for multiple data sets. The matrix or matrices have to be of type numeric and are required to have column names / variable names. The rows and the columns represent the observations and the variables, respectively.

y a vector, a matrix with one column, or list of the aforementioned objects for multiple data sets. The vector, vectors, matrix, or matrices have to be of type numeric. For family = "binomial", the response is required to be a binary vector taking values 0 and 1.

dendr the output of one of the functions cluster_var or cluster_position.
clvar a matrix or list of matrices of control variables.
family a character string naming a family of the error distribution; either "gaussian" or "binomial".
B number of sample splits.
proportion.select

proportion of variables to be selected by Lasso in the multi-sample splitting step.

standardize

a logical value indicating whether the variables should be standardized.

alpha

the significant level at which the FWER is controlled.

global.test

a logical value indicating whether the global test should be performed.

agg.method

a character string naming an aggregation method which aggregates the p-values over the different data sets for a given cluster; either "Tippett" (Tippett's rule) or "Stouffer" (Stouffer's rule). This argument is only relevant if multiple data sets are specified in the function call.

verbose

logical value indicating whether the progress of the computation should be printed in the console.

sort.parallel

a logical indicating whether the values are sorted with respect to the size of the block. This can reduce the run time for parallel computation.

parallel

type of parallel computation to be used. See the 'Details' section.

ncpus

number of processes to be run in parallel.

cl

an optional parallel or snow cluster used if parallel = "snow". If not supplied, a cluster on the local machine is created.

Details

The hierarchal testing requires the output of one of the functions cluster_var or cluster_position as an input (argument dendr).

The function first performs multi-sample splitting step. A given data with nobs is randomly split in two halves w.r.t. the observations and nobs * proportion.select variables are selected using Lasso (implemented in glmnet) on one half. Control variables are not penalized if supplied using the argument clvar. This is repeated B times for each data set if multiple data sets are supplied.

Those splits (i.e. second halves of observations) and corresponding selected variables are used to perform hierarchical testing by going top down through the hierarchical tree. Testing only continues if at least one child of a given cluster is significant.

The multi-sample splitting step can be run in parallel across the different sample splits where the argument B corresponds to number of sample splits. If the argument block was supplied for the building of the hierarchical tree (i.e. in the function call of either cluster_var or cluster_position), i.e. the second level of the hierarchical tree was given, the hierarchical testing step can be run in parallel across the different blocks by specifying the arguments parallel and ncpus. There is an optional argument cl if parallel = "snow". There are three possibilities to set the argument parallel: parallel = "no" for serial evaluation (default), parallel = "multicore" for parallel evaluation using forking, and parallel = "snow" for parallel evaluation using a parallel socket cluster. It is recommended to select RNGkind("L'Ecuyer-CMRG") and set a seed to ensure that the parallel computing of the package hierinf is reproducible. This way each processor gets a different substream of the pseudo random number generator stream which makes the results reproducible if the arguments (as sort.parallel and ncpus) remain unchanged. See the vignette or the reference for more details.

Note that if Tippett’s aggregation method is applied for multiple data sets, then very small p-values are set to machine precision. This is due to rounding in floating point arithmetic.
Value

The returned value is an object of class "hierT", consisting of two elements, the result of the multi-sample splitting step "res.multisplit" and the result of the hierarchical testing "res.hierarchy". The result of the multi-sample splitting step is a list with number of elements corresponding to the number of data sets. Each element (corresponding to a data set) contains a list with two matrices. The first matrix contains the indices of the second half of variables (which were not used to select the variables). The second matrix contains the column names / variable names of the selected variables.

The result of the hierarchical testing is a data frame of significant clusters with the following columns:

- `block`: NA or the name of the block if the significant cluster is a subcluster of the block or is the block itself.
- `p.value`: The p-value of the significant cluster.
- `significant.cluster`: The column names of the members of the significant cluster.

There is a `print` method for this class; see `print.hierT`.

References


See Also

`cluster_var`, `cluster_position`, and `compute_r2`.

Examples

```r
n <- 200
p <- 500
library(MASS)
set.seed(3)
x <- mvrnorm(n, mu = rep(0, p), Sigma = diag(p))
colnames(x) <- paste0("Var", 1:p)
beta <- rep(0, p)
beta[c(5, 20, 46)] <- 1
y <- x %*% beta + rnorm(n)

dendr1 <- cluster_var(x = x)
set.seed(68)
sign.clusters1 <- test_hierarchy(x = x, y = y, dendr = dendr1,
                               family = "gaussian")

## With block
# The column names of the data frame block are optional.
block <- data.frame("var.name" = paste0("Var", 1:p),
                 "block" = rep(c(1, 2), each = p/2),
                 stringsAsFactors = FALSE)
```
dendr2 <- cluster_var(x = x, block = block)
set.seed(23)
sign.clusters2 <- test_hierarchy(x = x, y = y, dendr = dendr2,
family = "gaussian")

# Access part of the object
sign.clusters2$res.hierarchy[, "block"]
sign.clusters2$res.hierarchy[, "p.value"]
# Column names or variable names of the significant cluster in the first row.
sign.clusters2$res.hierarchy[[1, "significant.cluster"]]

test_only_hierarchy Hierarchical Testing

Description
Hierarchical testing given the output of the function multisplit.

Usage
test_only_hierarchy(x, y, dendr, res.multisplit, clvar = NULL,
family = c("gaussian", "binomial"), alpha = 0.05,
global.test = TRUE, agg.method = c("Tippett", "Stouffer"),
verbose = FALSE, sort.parallel = TRUE, parallel = c("no",
"multicore", "snow"), ncpus = 1L, cl = NULL, check.input = TRUE,
unique.colnames.x = NULL)

Arguments

x a matrix or list of matrices for multiple data sets. The matrix or matrices have to be of type numeric and are required to have column names / variable names. The rows and the columns represent the observations and the variables, respectively.

y a vector, a matrix with one column, or list of the aforementioned objects for multiple data sets. The vector, vectors, matrix, or matrices have to be of type numeric. For family = "binomial", the response is required to be a binary vector taking values 0 and 1.

dendr the output of one of the functions cluster_var or cluster_position.
res.multisplit the output of the function multisplit.
clvar a matrix or list of matrices of control variables.
family a character string naming a family of the error distribution; either "gaussian" or "binomial".
alpha the significant level at which the FWER is controlled.
global.test a logical value indicating whether the global test should be performed.
a character string naming an aggregation method which aggregates the p-values over the different data sets for a given cluster; either "Tippett" (Tippett's rule) or "Stouffer" (Stouffer's rule). This argument is only relevant if multiple data sets are specified in the function call.

verbose

a logical value indicating whether the progress of the computation should be printed in the console.

sort.parallel

a logical indicating whether the values are sorted with respect to the size of the block. This can reduce the run time for parallel computation.

parallel
type of parallel computation to be used. See the 'Details' section.

ncpus

number of processes to be run in parallel.

c1

an optional parallel or snow cluster used if parallel = "snow". If not supplied, a cluster on the local machine is created.

check.input

a logical value indicating whether the function should check the input. This argument is used to call test_only_hierarchy within test_hierarchy.

unique.colnames.x

a character vector containing the unique column names of x. This argument is used to call test_only_hierarchy within test_hierarchy.

Details

The function test_only_hierarchy requires the output of one of the functions cluster_var or cluster_position as an input (argument dendr). Furthermore it requires the output of the function multisplit as an input (argument res.multisplit). Hierarchical testing is performed by going top down through the hierarchical tree. Testing only continues if at least one child of a given cluster is significant.

If the argument block was supplied for the building of the hierarchical tree (i.e. in the function call of either cluster_var or cluster_position), i.e. the second level of the hierarchical tree was given, the hierarchical testing step can be run in parallel across the different blocks by specifying the arguments parallel and ncpus. There is an optional argument cl if parallel = "snow". There are three possibilities to set the argument parallel: parallel = "no" for serial evaluation (default), parallel = "multicore" for parallel evaluation using forking, and parallel = "snow" for parallel evaluation using a parallel socket cluster. It is recommended to select RNGkind("L'Ecuyer-CMRG") and set a seed to ensure that the parallel computing of the package hierinf is reproducible. This way each processor gets a different substream of the pseudo random number generator stream which makes the results reproducible if the arguments (as sort.parallel and ncpus) remain unchanged. See the vignette or the reference for more details.

Note that if Tippett's aggregation method is applied for multiple data sets, then very small p-values are set to machine precision. This is due to rounding in floating point arithmetic.

Value

The returned value is an object of class "hierT", consisting of two elements, the result of the multi-sample splitting step "res.multisplit" and the result of the hierarchical testing "res.hierarchy".

The result of the multi-sample splitting step is a list with number of elements corresponding to the number of data sets. Each element (corresponding to a data set) contains a list with two matrices. The first matrix contains the indices of the second half of variables (which were not used to select
the variables). The second matrix contains the column names / variable names of the selected variables.

The result of the hierarchical testing is a data frame of significant clusters with the following columns:

- **block**: NA or the name of the block if the significant cluster is a subcluster of the block or is the block itself.
- **p.value**: The p-value of the significant cluster.
- **significant.cluster**: The column names of the members of the significant cluster.

There is a print method for this class; see `print.hierT`.

### References


### See Also

- `cluster_var`
- `cluster_position`
- `multisplit`
- `test_hierarchy`
- `compute_r2`

### Examples

```r
n <- 200
p <- 500
library(MASS)
set.seed(3)
x <- mvrnorm(n, mu = rep(0, p), Sigma = diag(p))
colnames(x) <- paste0("Var", 1:p)
beta <- rep(0, p)
beta[c(5, 20, 46)] <- 1
y <- x %*% beta + rnorm(n)
dendr1 <- cluster_var(x = x)
set.seed(76)
res.multisplit1 <- multisplit(x = x, y = y, family = "gaussian")
sign.clusters1 <- test_only_hierarchy(x = x, y = y, dendr = dendr1,
                                      res.multisplit = res.multisplit1,
                                      family = "gaussian")

# With block
block <- data.frame("var.name" = paste0("Var", 1:p),
                    "block" = rep(c(1, 2), each = p/2),
                    stringsAsFactors = FALSE)
dendr2 <- cluster_var(x = x, block = block)
# The output res.multisplit1 can be used since the multi-sample step is the same with or without blocks.
sign.clusters2 <- test_only_hierarchy(x = x, y = y, dendr = dendr2,
                                      res.multisplit = res.multisplit1,
                                      family = "gaussian")
```

family = "gaussian"

# Access part of the object
sign.clusters2$res.hierarchy[, "block"]
sign.clusters2$res.hierarchy[, "p.value"]

# Column names or variable names of the significant cluster in the first row.
sign.clusters2$res.hierarchy[[1, "significant.cluster"]]

**Index**

* datasets
  simGWAS, 11

cluster_position, 2, 5, 8, 9, 12–17
cluster_var, 3, 4, 8, 9, 12–17
compute_r2, 6, 8, 9, 14, 17
cor, 4
dendrogram, 3, 5
dist, 4
glmnet, 9, 13
hclust, 4
hierinf, 7
hierinf-package (hierinf), 7
invisible, 10
multisplit, 6, 8, 8, 9, 15–17
NagelkerkeR2, 7
print.data.frame, 10
print.hierT, 10, 14, 17
RNGkind, 3, 5, 9, 13, 16
simGWAS, 11
test_hierarchy, 3, 5–9, 12, 16, 17
test_only_hierarchy, 6, 8, 9, 15, 16