Package 'SourceSet'

April 24, 2018

Type Package

Title A Graphical Model Approach to Identify Primary Genes in Perturbed Biological Pathways

Version 0.1.1

Date 2018-04-23

Maintainer Elisa Salviato <elisa.salviato.88@gmail.com>

Description The algorithm pursues the identification of the set of variables driving the differences in two different experimental conditions (i.e., the primary genes) within a graphical model context. It uses the idea of simultaneously looking for the differences between two multivariate normal distributions in all marginal and conditional distributions associated with a decomposable graph, which represents the pathway under exam. The implementation accommodates genomics specific issues (low sample size and multiple testing issues) and provides a number of functions offering numerical and visual summaries to help the user interpret the obtained results. In order to use the (optional) 'Cytoscape' functionalities, the suggested 'r2cytoscape' package must be installed from the 'GitHub' repository ('devtools::install_github('cytoscape/r2cytoscape')').

Depends R (>= 2.10)

Imports gRbase, progress, reshape2, graph, igraph, gtools, methods, plyr, scales

License AGPL-3

Encoding UTF-8

LazyData true

RoxygenNote 6.0.1

Suggests networkD3, ggplot2, grDevices, Rgraphviz, knitr, rmarkdown, r2cytoscape, BiocStyle, Biobase, graphite, hgu95av2.db, ALL, mvtnorm, org.Hs.eg.db

VignetteBuilder knitr

NeedsCompilation no

Author Elisa Salviato [aut, cre], Vera Djordjilovic [aut], Chiara Romualdi [aut], Monica Chiogna [aut]

Repository CRAN

Date/Publication 2018-04-24 08:00:41 UTC

R topics documented:

`easyLookSource` *Easy look results*

Description

The function `easyLookSourcee` allows to summarize the results obtained from the `sourceSet` function through a heatmap, using `ggplot` library.

Usage

```
easyLookSource(sourceObj, name.graphs = names(sourceObj),
  map.name.variable = NULL, label.variable = "Variable",
  label.graph = "Graph", subname.variable = 10, subname.graph = 20,
  maxnum.variable = 50, maxnum.graph = 30,
  title = "Source Set for each Pathway", subtitle = NULL,
  coord.equal = TRUE, coord.flip = FALSE, strsplit.variable = " ",
  strsplit.graph = " ", col.primary = "#324E7B",
  col.secondary = "#86A6DF")
```

Arguments

`sourceObj`	a `SourceSetObj` objects, i.e. the output of the `sourceSet` function
`name.graphs`	the graphs names to be visualized. Default value is `names(sourceObj)`

`map.name.variable`
: a list of customized labels to be associated with the names of the genes. Each list element must contain only one value (i.e. the new label), and the name of each element must be associated with the names of the genes given as input to the `sourceSet` function (column names of `data` input argument). If a label is not mapped, the original name is used

`label.variable`
: title of the variable axis

`label.graph`
: title of the graph axis

`subname.variable`
: number of characters of the variable names labels to show. The function cuts the name at the first `str.split` character that doesn't exceed `subname.variable`

`subname.graph`
: number of characters of the graph names labels to show. The function cuts the name at the first `str.split` character that doesn't exceed `subname.variable`

`maxnum.variable`
: maximal number of variables to include in the plot. The variables are sorted internally and only the first `maxnum.variable` are be plotted.

`maxnum.graph`
: maximal number of graphs to include in the plot. The graphs are sorted internally and only the first `maxnum.variable` are be plotted.

`title`
: overall title of the plot

`subtitle`
: subtitle of the plot

`coord.equal`
: if `TRUE`, forces the scale coordinate system to be equal for the y and x axis. See also `coord_fixed`

`coord.flip`
: if `TRUE`, flips cartesian coordinates so that horizontal becomes vertical, and vertical, horizontal. Default option sets to x axis the variables, and to the y axis the graphs. See also `coord_flip`

`strsplit.variable`
: character containing regular expression to use for cut varibale labels to be shown. More details in `subname.variable` and `subname.variable` argument descriptions

`strsplit.graph`
: character containing regular expression to use for cut graph labels to be shown. More details in `subname.variable` and `subname.variable` argument descriptions

`col.primary`
: cell color for the variables responsable of primary dysregulation

`col.secondary`
: cell color for the variables responsable of secondary dysregulation

Details

The plot is composed of a matrix whose rows represent pathways (i.e., graphs) and columns represent genes (i.e., variables). Each cell i, j can take one of the following configurations:

- `2`: blue color, if the i-th gene is in the primary set of the j-th pathway
- `1`: light blue color, if the i-th gene is in the secondary set of the j-th pathway
- `0`: gray, if the i-th gene belongs to the j-th pathway
- `NA`: white, if the i-th gene does not belong to the j-th pathway

In the plot, the pathways are vertically ordered - top to bottom - according to the numbers of nodes in the source set. The genes are horizontally ordered (from left to right) based on the number of times they appear in a source set.

See Also

sourceSet, sourceSankeyDiagram

Examples

```
## Load the SourceSetObj obtained from the source set analysis of ALL dataset

# see vignette for more details
print(load(file=system.file("extdata","ALLsourceresult.RData",package = "SourceSet")))
class(results.all)
n.primary<-length(lapply(results.all,function(x) x$primarySet))

# show only genes that appear in at least one of the source sets of the investigated pathways
easyLookSource(sourceObj=results.all, maxnum.variable = n.primary,
               label.variable = "Genes",label.graph = "Pathways")

# flip coordinates
easyLookSource(sourceObj = results.all,maxnum.variable = n.primary,coord.flip = TRUE)
```

---

getPermutations *Get random permutations of a set of elements*

Description

The function arranges, in an optimized way, all the elements of a set into a selected number of different sequences (i.e., permutations). If the number of possible orderings is less than the required number, the function returns the collection of all possible permutations.

Usage

```
getPermutations(n, nperms)
```

Arguments

n	number of elements
nperms	number of required permutations

Value

The function returns:

- perms: a matrix with nperms rows and n columns, containing the sequence of the ordered elements

- `all.perms.flag`: `1` if the perms array contains the entire collection of permutations, `0` otherwise. In the first case, the number of rows of `perms` matrix may be less than the number of requested permutations
- `nperms.act`: the number of permutations returned

Examples

```
sub.perm<-getPermutations(10,100)
all.perm<-getPermutations(3,100)
```

`infoSource` *Get summary statistics on graphs and variables*

Description

The `infoSource` function provides a summary of the results by focusing on either variables or graphs.

Usage

```
infoSource(sourceObj, map.name.variable = NULL, method = "fdr")
```

Arguments

`sourceObj`	a `SourceSetObj` object, i.e. the output of the `sourceSet` function
`map.name.variable`	a list of customized labels to be associated with the names of the genes. Each list element must contain only one value (i.e. the new label), and the name of each element must be associated with the names of the genes given as input to the `sourceSet` function (column names of `data` input argument). If a label is not mapped, the original name is used
`method`	correction method for p-values calculated on graphs. The adjustment methods allowed are: `fdr` (default), `holm`, `hochberg`, `hommel`, `bonferroni`, `BH`, `BY` or `none`. For more details refer to `p.adjust`.

Value

The function guides the user in identifying interesting variables returning two objects:

- `graph`: a dataframe that summirizes the results of the individual input graphs, composed as follows:
  - `n.primary`: number of genes belonging to the source set;
  - `n.secondary`: number of genes belonging to the secondary set;
  - `n.graph`: number of genes within the graph;
  - `n.cluster`: number of connected components of the graph;
  - `primary.impact`: relative size of the estimated source set. This index quantifies the proportion of the graph impacted by primary dysregulation;

  - total.impact: relative size of the set of genes impacted by dysregulation. This index quantifies the proportion of the graph impacted by either primary or secondary dysregulation;
  - adj.pvalue: multiplicity adjusted p-value for the hypothesis of equality of the two distributions associated to the given graph

- variable: a dataframe that summarized the results of the individual variables, composed as follows:
  - n.primary: number of input graphs in which the gene appears in the associated source set;
  - n.secondary: number of input graphs in which the gene appears in the associated secondary set;
  - n.graph: number of pathways in which the gene is annotated;
  - specificity: percentage of input graphs containing the given genes with respect to the total number of input graphs;
  - primary.impact: percentage of input graphs, such that the given gene belongs to their estimated source set, with respect to the total number of input graphs in which the gene appears;
  - total.impact: percentage of input graphs, such that the given gene is affected by some form of dysregulation in the considered graph, with respect to the total number of input graphs in which the gene appears;
  - relevance: percentage of the input graphs such that the given variable belongs to their estimated source set, with respect to the total number of input graphs. It is a general measure of the importance of the gene based on the chosen pathways;
  - score: a number ranging from 0 (low significance) to +Inf (maximal significance), computed as the combination of the p-values of all components (of all the input graphs) containing the given variable

Note

Ideally, variables of the primary dysregulation will be elements of the source set in all input graphs that contain them and will thus have high values of source.impact and score. However, if a given variable appears in a single graph, and belongs to its source set, these indices can be deceptive.

For this reason, relevance serves to identify variables that apart from being good candidates for primary genes, also appear frequently in the input graphs. Which index is to be preferred depends on the objective of the analysis: in case of exploratory analysis, we suggest to rely on relevance.

Examples

```
## Load the SourceSetObj obtained from the source set analysis of ALL dataset

# see vignette for more details
print(load(file=system.file("extdata","ALLsourceresult.RData",package = "SourceSet")))
class(results.all)

info.all<-infoSource(sourceObj = results.all)
## results of individual input graphs
info.all$graph
```

```
## results of individual variables
# ..that appear in more than one graph and with relevance>0
info.all.genes<-info.all$variable[info.all$variable$n.graph>1 & info.all$variable$relevance>0,]
# ..ordered by score
ind.ord<-order(info.all.genes$relevance,decreasing = TRUE)
info.all.genes[ind.ord,]
```

`parameters` *Estimation of parameters for test equality of two normal distributions*

Description

The function estimates the parameters of two normal distributions. Both maximum likelihood estimates and shrinkage estimate of covariance matrices are supplied.

Usage

```
parameters(data, classes, shrink = TRUE, shrink.function = shrinkTEGS,
  shrink.param = list(probs = 0.05, type = "min"))
```

Arguments

`data`	an expression matrix with colnames for variables and row names for samples
`classes`	a vector of length equal to the number of rows of data. It indicates the class (condition) of each statistical unit. Only two classes, labeled as 1 and 2, are allowed
`shrink`	boolean. if `FALSE` the maximum likelihood estimates are returned; if `TRUE` the shrinkage estimates are returned instead
`shrink.function`	function that implements the shrinkage method. It must return a list object with all the elements required as input arguments in `testMeanVariance`. Default is `shrinkTEGS` function.
`shrink.param`	additional parameters to pass as input arguments of the shrink function specified in the `shrink.function`.

See Also

`shrinkTEGS`, `testMeanVariance`

Examples

```
if(require(mvtnorm)){
  ## Generate two random samples of size 50 from two multivariate normal distributions
  # sample size
  n<-50
  # true parameters of class 1 and class 2
  param.class1<-simulation$condition1
```

```
  param.class2<-simulation$condition2$`5`$`2`
  # simulated dataset
  data.class1<-rmvnorm(n = n,mean =param.class1$mu ,sigma =param.class1$S)
  data.class2<-rmvnorm(n = n,mean =param.class2$mu ,sigma=param.class2$S)
  data<-rbind(data.class1,data.class2)
  classes<-c(rep(1,nrow(data.class1)),rep(2,nrow(data.class2)))

  ## estimated parameters: maximum likelihood estimate
  est.param<-parameters(data = data,classes =classes ,shrink = FALSE)

  ## estimated parameters: regularized estimate
  est.param.shrink<-parameters(data = data,classes =classes ,shrink = TRUE)
  # tuning values and other info on shrinkage estimate
  str(est.param.shrink$shrink.info)
}
```

`ripAllRootsClique` *All possible RIP orderings*

Description

The function identifies all possible clique orderings leading to distinct factorizations of the associated joint distribution.

Usage

```
ripAllRootsClique(graph)
```

Arguments

`graph`	a graph represented as a `graphNEL` object. If the input graph is not decomposable, the function will internally moralize and triangulate it.

Details

For each root clique, the function uses `rip` function to identify a sequence of the set of cliques that satisfies the running intersection property by first ordering variables by the maximum cardinality search algortithm. The `root` argument is used to check which clique will be the first to enter in the rip ordering.

Value

Given a graph, the function returns:

- `elements`: a list composed of four other lists:
  - `cliques`: (a list of character vectors) variables contained in each maximal clique of the moralized and triangulated input graph
  - `separators`: (a list of character vectors) unique separators, i.e., common variables among cliques

  - components: (a dataframe) unique "clique | separator" elements defined on the basis of all rip orderings. Each element represents a conditional distribution (see Djordjilovic and Chiogna)
  - variables: (a character vector) nodes of the graph
- indices: a list composed of two other lists:
  - all: (a list of character vectors) cliques and separators lists of variables
  - ordering: one dataframe for each identified ordering. Each data frame is a subset of size k (i.e., number of maximal cliques), of the components elements. The name of each list corresponds to the used root clique.
- graph: decomposable graph used in the identification of rip orderings. It may differ from the input graph. In fact, if the input graph is not decomposable, the function will internally moralize and triangulate it.

See Also

rip

Examples

```
if(require(gRbase)){

  ## decomposable graph
  ug.graph<-ug(~1:2:3+3:4+4:5:6:7)
  ug.rip.all<-ripAllRootsClique(ug.graph)
  # 7 variables
  length(ug.rip.all$elements$variables)
  # 3 max.cliques
  length(ug.rip.all$elements$cliques)
  # 7 unique components
  nrow(ug.rip.all$elements$components)
  # all rip orderings:
  ug.rip.all$indices$ordering

  ## directed graph
  dag.graph<-dag(~3:1+3:2+4:3)
  dag.rip.all<-ripAllRootsClique(dag.graph)
  # triangulated and morliazed graph
  dag.rip.all$graph
  # all rip orderings
  dag.rip.all$indices$ordering
}
```

---

shrinkTEGS *Default shrinkage estimation of covariance matrices*

Description

The function adds a small quantity to the diagonals of covariance matrix estimates to regularize them.

Usage

```
shrinkTEGS(s, s1, s2, param = list(probs = 0.05, type = "min"))
```

Arguments

s	covariance matrix estimte in the pooled sample
s1	sample covariance matrix estimate in class 1
s2	sample covariance matrix estimate in class 2
param	list of parameters:

- lamda: a vector of lambdas (to be supplied only if some custom lambdas are to be used)
- type: minimum (min), maximum (max) or optimal (opt)
- probs: the numeric value of probability with value in [0,1]

Details

To determine the quantity to add to the diagonals of covariance matrices, the function:

- finds the distributions of the sample variances of the p variables in the two classes and in the pooled sample
- computes the probs percentile of each of these distributions
- use the minimum, maximum or optimal (one for each matrix) (type)

Note

It should be stressed that the default parameters for TEGS shrink estimator allow to compare the log likelihood criterion among distributions if the testMeanVariance is performed.

References

Huang, Y.-T. and Lin, X. (2013). Gene set analysis using variance component tests. BMC Bioinformatics, 14(1), 210.

See Also

testMeanVariance, parameters

Examples

```
if(require(mvtnorm)){

  ## Generate two random samples of size 50 from two multivariate normal distributions
  # sample size
  n<-50
  # true parameters of class 1 and class 2
  param.class1<-simulation$condition1
  param.class2<-simulation$condition2$`5`$`2`
  # simulated dataset
```

```
  data.class1<-rmvnorm(n = n,mean =param.class1$mu ,sigma =param.class1$S)
  data.class2<-rmvnorm(n = n,mean =param.class2$mu ,sigma=param.class2$S)
  data<-rbind(data.class1,data.class2)
  classes<-c(rep(1,nrow(data.class1)),rep(2,nrow(data.class2)))

  ## estimated parameters: maximum likelihood estimate
  s<-cov(data)
  s1<-cov(data.class1)
  s2<-cov(data.class2)

  ## default parameters:
 # use the minimum of median variances distributions of the three supplied covariance matrices
  def.shrink<-shrinkTEGS(s,s1,s2)
  def.shrink$lambda

  ## use customize lamdas
  def.shrink<-shrinkTEGS(s,s1,s2,param = list(lambda=c(0.1,0.2,0.3)))
  def.shrink$lambda

  # use for each covariance matrix the 0.4 percentile of its variances distributions
  def.shrink<-shrinkTEGS(s,s1,s2,param = list(type="opt",probs=0.4))
  def.shrink$lambda
}
```

`simulation` *Simulated dataset*

Description

This data contains the parameters used in the study of the finite case behavior of source set algorithm, as described in of Salviato et al. (2018).

Usage

```
data(simulation)
```

Format

A list class that contains the true parameters (`mu`, vector of means and `S`, covariances matrix) of two multivariate normal distributions in two different experimental conditions (`condition1`, reference condition and `condition2`, perturbed condition) and the underlying graphical structure G (`graph`. Six different perturbations are considered, see below.).

The differences between the two conditions are driven by:

- a node that is a separator within the graph (`simulation$condition2$`5``)
- a node that is contained in only one clique of the graph (`simulation$condition2$`10``)

The intensity of the artificial perturbation is:

- mild (`simulation$condition2$`10`$`1.2``)

- moderate (`simulation$condition2$`10`$`1.6`)
- strong (`simulation$condition2$`10`$`2`)

Details

The starting parameters of the reference condition are obtained by randomly selecting a gene set of the same cardinality as the order of the graph G, from the Acute Lymphocytic Leukemia (`ALL`) dataset. These are then modified to represent the parameters of the perturbed condition. Formally, starting from the parameters related to the reference group, the procedure act on means and variances so that the conditional distribution of the variables on which it does not directly intervene remains unchanged under the two conditions. However, this action affects the entire global joint distribution, thus creating the propagation effect. See Salviato et al. (2016) for more details.

References

Chiaretti, S. et al. (2005). Gene expression profiles of b-lineage adult acute lymphocytic leukemia reveal genetic patterns that identify lineage derivation and distinct mechanisms of transformation. Clinical Cancer Research, 11(20), 7209–7219.

Salviato, E. et al. (2016). `simPATHy`: a new method for simulating data from perturbed biological pathways. Bioinformatics, 33(3), 456–457.

Salviato, E. et al. (2018). `SourceSet`: a graphical model approach to identify primary genes in perturbed biological pathways. Manuscript under submission.

See Also

`simPATHy`, `ALL`

`sourceCytoscape` *Visualize in Cytoscape a collection of graphs analyzed with the source set algorithm*

Description

The function, thanks to the connection with the Cytoscape software, allows the user to create a collection of graphs to be visualized in a unique session, while documenting interesting findings.

Usage

```
sourceCytoscape(sourceObj, name.graphs = names(sourceObj),
  collection.name = "SourceCollection", map.name.variable = NULL,
  method = "bonferroni")
```

Arguments

sourceObj	a SourceSetObj objects, i.e. the output of the sourceSet function.
name.graphs	the names of the graphs to be visualized. Default value is names(sourceObj). NOTE: even if a subset of graphs are selected in name.graphs, the returned statistics are always calculated on the entire collection in the sourceObj argument.
collection.name	name of the collection of graphs displayed in Cytoscape.
map.name.variable	a list of customized labels to be associated with the names of the genes. Each list element must contain only one value (i.e. the new label), and the name of each element must be associated with the names of the genes given as input to the sourceSet function (column names of data input argument). If a label is not mapped, the original name is used.
method	correction method for p-values calculated on graphs. The adjustment methods allowed are: fdr (default), holm, hochberg, hommel, bonferroni, BH, BY or none. For more details refer to p.adjust.

Details

The visual node attributes size and fill color are defined in a dynamic manner through a visual mapping based on the indices provided by the infoSource function (automatically uploaded in the bottom panel - right side).

A discrete mapper between source attribute and size is applied:

- big size: the variable belongs to the primary set (source=2);
- medium size: the variable belongs to the secondary set (source=1);
- small size: otherwise (source=0).

On the other hand, a color gradient mapper between fill node color and relevance is adopted: higher values are highlighted with darker blue color.

The default style can be changed manually either within Cytoscape (for further information see manual) or within an R package r2cytoscape through network SUID returned by the sourceCytoscape function (for further details see manual).

It is also possible to call the sourceCytoscape function multiple times, with all the graphs being visualized in a unique session within a collection specified by collection.name.

Note

The function use the r2cytoscape package to connect to Cytoscape from R using CyREST. r2cytoscape can be downloaded from:

- Bioconductor: biocLite("r2cytoscape");
- GitHub: install_github("cytoscape/r2cytoscape").

To enable the display function to work properly, three simple steps are required:

- Download Cytoscape (version 3.3 or later);

- Complete installation wizard;
- Launch Cytoscape (before calling the functions).

See Also

sourceSet, infoSource. sourceUnionCytoscape, r2cytoscape

Examples

```
## Load the SourceSetObj obtained from the source set analysis of ALL dataset

# see vignette for more details
print(load(file=system.file("extdata","ALLsourceresult.RData",package = "SourceSet")))
class(results.all)

## NB: Remember to launch cytoscape before running the following commands
# Create two collections of pathways to visualize the results
graph.signaling<-names(results.all)[grep("signaling",names(results.all))]
graph.other<-setdiff(names(results.all),graph.signaling)

## Signaling collection

cytoID.signaling<-sourceCytoscape(results.all,
    name.graphs = graph.signaling, collection.name ="SignalingPathway")


## Other collection

cytoID.other<-sourceCytoscape(results.all,
    name.graphs = graph.other, collection.name ="OtherPathway")
```

---

sourceSankeyDiagram *Create a D3 JavaScript Sankey diagram*

Description

The function sourceSankeyDiagram allows to summarize the results obtained from the sourceSet function through a Sankey diagram, highlighting the relationships among nodes, graphs, and source sets.

Usage

```
sourceSankeyDiagram(sourceObj, name.graphs = names(sourceObj),
  map.name.variable = NULL, cutoff = 50, cut.extra.module = TRUE,
  height = NULL, width = NULL)
```

Arguments

sourceObj	a SourceSetObj objects, i.e. the output of the sourceSet function
name.graphs	the names of the graphs to be visualized. Default value is names(sourceObj)
map.name.variable	a list of customized labels to be associated with the names of the genes. Each list element must contain only one value (i.e. the new label), and the name of each element must be associated with the names of the genes given as input to the sourceSet function (column names of data input argument). If a label is not mapped, the original name is used
cutoff	the maximum number of variables to include in the sankey graph. The final number of visualized variables could be higher than the cutoff number
cut.extra.module	if set to TRUE, modules consisting only of variables excluded by the cutoff are not displayed
height	numeric height (in pixels) for the network graph's frame area
width	numeric width (in pixels) for the network graph's frame area

Details

The layout is organized on three levels:

- the first level (left) shows nodes that appear in at least one source sets of the analyzed graphs;
- the second level (center) is made up of modules. A module is defined as a set of nodes belonging to a connected subgraph of one pathway, that is also contained in associated source set. A pathway can have multiple modules, and, at the same time, one module can be contained in multiple pathways;
- the third level (right) shows of pathways.

The three levels are to be read from left to right. A link between left element a and right element b must be interpret as "a is contained in b".

The implementation of the sourceSankeyDiagram function takes advantage of the D3 library (JavaScript), making the plot interactive. In fact, it is possible to vertically shift the displayed elements, and to view some useful information by positioning the cursor over items and links.

References

Allaire, J.J., Gandrud, G., Russell, K., and Yetman, C.J. (2017). networkD3: D3 JavaScript Network Graphs from R, r package version 0.4 edition.

Bostock, M., Ogievetsky, V., and Heer, J. (2011). D3 data-driven documents. IEEE Transactions on Visualization and Computer Graphics, 17(12):2301–2309.

See Also

sankeyNetwork, sourceSet, easyLookSource

Examples

```
## Load the SourceSetObj obtained from the source set analysis of ALL dataset

# see vignette for more details
print(load(file=system.file("extdata","ALLsourceresult.RData",package = "SourceSet")))
class(results.all)

sourceSankeyDiagram(sourceObj = results.all ,cut.extra.module = FALSE )

# shows the variable that appears most often in the source sets
sourceSankeyDiagram(sourceObj = results.all, cutoff = 1 ,cut.extra.module = FALSE )
# cut modules in which the variable is not contained
sourceSankeyDiagram(sourceObj = results.all, cutoff = 1 ,cut.extra.module = TRUE )
```

---

sourceSet *Source Set*

Description

Identify the sets of variables that are potential sources of differential behavior, (i.e., the primary genes) between two experimental conditions. The two experimental conditions are associated to a set of graphs, where each graph represents the topology of a biological pathway.

Usage

```
sourceSet(graphs, data, classes, seed = NULL, theta = 1, permute = TRUE,
  alpha = 0.05, shrink = FALSE, return.permutations = FALSE)
```

Arguments

graphs	a list of graphNEL objects representing the pathways to be analyzed.
data	a matrix of expression levels with column names for genes and row names for samples; gene names must be unique.
classes	a vector of length equal to the number of rows of data. It indicates the class (condition) of each statistical unit. Only two classes, labeled as 1 and 2, are allowed;
seed	integer value to get a reproducible random result. See Random.
theta	positive numeric value greater then 1, that defines the number of permutation. If permute=TRUE, (m/alpha x theta) permutations are used, where m is the number of unique conditional tests to be performed; otherwise, (1/alpha x theta) permutations are supplied.
permute	if TRUE permutation p-values are provided; if FALSE, asymptotic p-values are returned. NOTE: even if the argument permute is set to FALSE the function will permute the dataset; these permutations will be used to calculate the adjusted cut-off for the asymptotic p-values.

`alpha` the p-value threshold. Denotes the level at which FWER is controlled for each input graph.

`shrink` if `TRUE`, regularized estimation of the covariance matrices is performed; otherwise, maximum likelihood estimations is used.

`return.permutations` if `TRUE`, the function returns the matrix of test statistic values for the supplied (first row) and the permutated datasets.

Details

The `sourceSet` approach models the data of the same pathway in two different experimental conditions as realizations of two Gaussian graphical models sharing the same decomposable graph G. Here, G = (V,E) is obtained from the pathway topology conversion, where V and E represent genes and biochemical reactions, respectively.

We give full freedom to the user in providing the underlying graph G, requiring only a specific input format (i.e., a `graphNEL` object). So, the user can provide a list of manually curated pathways or use developed software to translate the bases of knowledge. To date, the most complete software available for this task is `graphite` R package (Sales et al. 2017).

The source set algorithm infers the set of primary genes (i.e., the source set) following - for each graph - five steps:

- decompose graph G in the set of the maximal cliques and the set of separators.
- identify the cliques orderings, and the associated separators, that satisfy the running intersection property, using each cliques as root. See `ripAllRootsClique`.
- a) calculate marginal test statistics for the cliques and the separators, for both the original and the permutated datasets; b) compute the conditional test statistics for the unique components, calculated as the difference between clique and separator marginal test statistics; c) control the FWER, using the test statistics matrix of the previous point.
- make the union of the sets of variables belonging to cliques that are associated to a significant test, within each decomposition.
- derive the source set, defined as the intersection of the set of variables obtained in step 4 across decompositions.

Although the interpretation of the source set for a single graph is intuitive, the interpretation of the collection of results associated to a set of pathways might be complex. For this reason, we propose a guideline for the meta-analysis providing descriptive statistics and predefined plots. See, `infoSource`, `easyLookSource`, `sourceSankeyDiagram`, `sourceCytoscape` and `sourceUnionCytoscape`.

Value

The output of the function is an object of the `sourceSetList` class. It contains as many lists as the input graphs, and each of them provides the following variables:

- `primarySet`: a character vector containing the names of the variables belonging to the estimated source set (primary dysregulation);
- `secondarySet`: a character vector containing the names of the variables belonging to the estimated secondary set (secondary dysregulation);

- `orderingSet`: a list of character vectors containing the names of the variables belonging to the estimated source set of each ordering; the union of these elements contains all genes affected by some form of perturbation;
- `Components`: a data frame that contains information about unique tests, including their associated p-values;
- `Decompositions`: a list of data frames, one for each identified ordering. Each data frame is a subset of size k (i.e., number of cliques), of the `Components` elements
- `Elements`: cliques and separators of the underlying decomposable graph. See `Graph`
- `Thresholds`: a list with information regarding the multiple testing correction:
  - `alpha`: the input (nominal) significance level;
  - `value`: the corrected threshold that ensures the control of FWER at level `alpha`;
  - `type`: the used procedure (minP or maxT);
  - `iterations`: the number of iterations for the step-down procedure;
  - `nperms`: the number of permutations.
- `Graph`: decomposable graph used in the analysis. It may differ from the input graph. In fact, if the input graph is not decomposable, the function will internally moralize and triangulate it.

Note

If `permute` and/or `shrink` parameters violate the conditions required for the existence of the full-rank maximum likelihood estimates, the algorithm reserves the possibility to change the user settings through internal controls.

Indeed, if the user wants to use the MLE of the covariance matrix (`shrink=FALSE`), all cliques - in all pathways - must satisfy the $n > p_i$ condition, where n is the number of samples for the smaller class and p_i is the cardinality of the largest clique in the i-th pathway. If even one clique does not satisfy this requirement, the regularized estimate must be used. When a regularized estimate is employed (`shrink=TRUE`), the analytical null distribution of the test statistics is no longer available, and we rely on permutation methods to obtain the associated p-values.

To address the multiple testing problem we use two versions of the method proposed by Westfall and Young (2017), which uses permutations to obtain the joint distribution of the p-values. More specifically, when the maximum likelihood estimates of the covariance matrices are used (`shrink=FALSE`), the asymptotic p-values and the maxT approach is adopted. While, if the regularized estimates are calculated (`shrink=TRUE`), asymptotic distribution is no longer valid and the min P version and the per-hypothesis permutation p-values to obtain the joint distribution of the p-values are needed. The number of permutations depends on the method, the alpa level chosen, and the number of hypotheses. A minimum number of 500 and a maximum number of 10.000 permutations are allowed.

References

Sales, G. et al. (2017). graphite: GRAPH Interaction from pathway Topological Environment, r package version 1.22.0 edition.

Westfall, P. and Young, S. (2017). Resampling-based multiple testing : examples and methods for p-value adjustment. Wiley.

Djordjilovic, Vera and Chiogna, Monica (2017) Searching for a Source of Difference: a Graphical Model Approach. [Working Paper] WORKING PAPER SERIES, 4/2017, PADOVA

Salviato et al. (2018). SourceSet: a graphical model approach to identify primary genes in perturbet biological pathways. Manuscript submitted for publication.

See Also

pathways, infoSource, easyLookSource, sourceSankeyDiagram, sourceCytoscape and sourceUnionCytoscape

Examples

```
#### Toy example: only one graph
if(require(mvtnorm)){
  # Generate two random samples of size 50 from two multivariate normal distributions
  n<-50
  # true parameters of class 1 and class 2
  param.class1<-simulation$condition1
  param.class2<-simulation$condition2$`10`$`2`

  # simulated dataset
  data.class1<-rmvnorm(n = n,mean =param.class1$mu ,sigma =param.class1$S)
  data.class2<-rmvnorm(n = n,mean =param.class2$mu ,sigma=param.class2$S)

  # Input arguments for the sourceSet function
  data<-rbind(data.class1,data.class2)
  classes<-c(rep(1,nrow(data.class1)),rep(2,nrow(data.class2)))
  graphs<-list("toy.graph"=simulation$graph)

  result<-sourceSet(graphs ,data ,classes ,seed = 123 ,permute =FALSE ,shrink =FALSE, alpha=0.05  )

  # source set: primary disregulation (toy.graph)
  result$toy.graph$primarySet
  # secondary disregulation (toy.graph)
  result$toy.graph$secondarySet
  # all affected variables
  unique(unlist(result$toy.graph$orderingSet))

  # summary statistics
  info<-infoSource(result)
  info$variable
  info$graph

  # visual summaries
  easyLookSource(result)
  sourceSankeyDiagram(result)
}

# launch cytoscape and run:

sourceCytoscape(result,name.graphs = "toy.graph",collection.name = "Example")
sourceUnionCytoscape(result ,collection.name = "Example")
```

```
### Real data:
# see vignette, section Getting deepening
vignette("SourceSet")
```

`sourceUnionCytoscape` *Visualize in Cytoscape the graphical union induced by the source sets of a collection of graphs*

Description

The function, thanks to the connection with the Cytoscape software, allows the user to create the graphical union induced by the source sets of a collection of graphs to be visualized in a unique session, while documenting interesting findings.

Usage

```
sourceUnionCytoscape(sourceObj, name.graphs = names(sourceObj),
  collection.name = "SourceSetUnion", network.name = "UnionSourceSetsGraph",
  map.name.variable = NULL, method = "bonferroni", complete.edges = TRUE,
  return.unionGraph = FALSE)
```

Arguments

`sourceObj`	a `SourceSetObj` objects, i.e. the output of the `sourceSet` function.
`name.graphs`	the names of the graphs to be visualized. Default value is `names(sourceObj)`. NOTE: even if a subset of graphs are selected in `name.graphs`, the returned statistics are always calculated on the entire collection in the `sourceObj` argument.
`collection.name`	name of the collection of graphs displayed in Cytoscape.
`network.name`	name of the resulting union graph.
`map.name.variable`	a list of customized labels to be associated with the names of the genes. Each list element must contain only one value (i.e. the new label), and the name of each element must be associated with the names of the genes given as input to the `sourceSet` function (column names of `data` input argument). If a label is not mapped, the original name is used.
`method`	correction method for p-values calculated on graphs. The adjustment methods allowed are: `fdr` (default), `holm`, `hochberg`, `hommel`, `bonferroni`, `BH`, `BY` or `none`. For more details refer to `p.adjust`.
`complete.edges`	if `TRUE`, the graphs selected in `name.graphs` are merged and the induced graph of the variables that appear at least in a source set is returned. if `FALSE`, the subgraph induced by the variables in the source set of each graph specified in `name.graphs` is found, and their union is returned.
`return.unionGraph`	if `TRUE`, the function returns the data frame of the edges of the resulting union graph, together with information about the variables obtained internally through the `infoSource` function.

Details

The visual node attributes size and fill color are defined in a dynamic manner through a visual mapping based on the indices provided by the `infoSource` function (automatically uploaded in the bottom panel - right side).

A continous mapper between `sub.n.source` attribute and size is applied: higher values are represented with bigger nodes. On the other hand, a color gradient mapper between fill node color and `relevance` is adopted: higher values are highlighted with darker blue color.

The edges connecting nodes belonging to the graph induced by the source set of each graph are represented by a solid line; while, the edges that connect two variables linked in the union of the graphs, but not within the same source set of a single graph, have dotted lines (supplied only if `complete.edges=TRUE`).

The default style can be changed manually either within Cytoscape (for further information see manual) or within an R package `r2cytoscape` through network SUID returned by the `sourceCytoscape` function (for further details see manual).

It is also possible to call the sourceCytoscape function multiple times, with all the graphs being visualized in a unique session within a collection specified by collection.name.

Note

The function use the `r2cytoscape` package to connect to Cytoscape from R using CyREST. `r2cytoscape` can be downloaded from:

- Bioconductor: `biocLite("r2cytoscape")`;
- GitHub: `install_github("cytoscape/r2cytoscape")`.

To enable the display function to work properly, three simple steps are required:

- Download Cytoscape (version 3.3 or later);
- Complete installation wizard;
- Launch Cytoscape (before calling the functions).

See Also

`sourceSet`, `sourceCytoscape`, `r2cytoscape`

Examples

```
## Load the SourceSetObj obtained from the source set analysis of ALL dataset

# see vignette for more details
print(load(file=system.file("extdata","ALLsourceresult.RData",package = "SourceSet")))
class(results.all)

## NB: Remember to launch cytoscape before running the following commands
# Create two collections of pathways to visualize the results
graph.signaling<-names(results.all)[grep("signaling",names(results.all))]
graph.other<-setdiff(names(results.all),graph.signaling)
```

```
## Signaling collection

cytoID.signaling.union<-sourceUnionCytoscape(results.all,
      name.graphs =graph.signaling ,collection.name ="SignalingPathway",
      network.name ="SignalingUnion")

## Other collection

cytoID.other.union<-sourceUnionCytoscape(results.all ,
      name.graphs =graph.other,collection.name ="OtherPathway" ,
      network.name ="OtherUnion")
```

`testMeanVariance` *Test the equality of two normal distributions*

Description

The function performs the test of equality of two multivariate normal distrbutions (class1 and class2).

Usage

```
testMeanVariance(S, S1, S2, n1, n2)
```

Arguments

`S`	estimated covariance matrix for pooled sample
`S1`	estimated covariance matrix in class 1
`S2`	estimated covariance matrix in class 2
`n1`	number of samples in class 1
`n2`	number of samples in class 2

Details

The criterion for testing the equality of two normal distributions is the following:

$$\Lambda_c = n_1 * log(|S|/|S^1|) + n_2 * log(|S|/|S^2|)$$

The asymptotic null distribution of the criterion, when the maximum likelihood estimates of the covariance matrices are used, is Chi square with $|\Gamma| * (|\Gamma| + 3)/2$ degrees of freedom, where G is the dimension of the underlying distributions.

Value

The function returns a list that contain the test statistic (`stat`) and the p-value test obtained of equality, using the asymptotic distribution (`alpha`).

Note

The asymptotic null distributions holds only when the maximum likelihood estimates of the covariance matrices are supplied.

See Also

parameters

Examples

```
if(require(mvtnorm)){

  ## Generate two random samples of size 50 from two multivariate normal distributions
  # sample size
  n<-50
  # true parameters of class 1 and class 2
  param.class1<-simulation$condition1
  param.class2<-simulation$condition2$`5`$`2`
  # simulated dataset
  data.class1<-rmvnorm(n = n,mean =param.class1$mu ,sigma =param.class1$S)
  data.class2<-rmvnorm(n = n,mean =param.class2$mu ,sigma=param.class2$S)
  data<-rbind(data.class1,data.class2)
  classes<-c(rep(1,nrow(data.class1)),rep(2,nrow(data.class2)))

  s<-cov(data)
  s1<-cov(data.class1)
  s2<-cov(data.class2)
  testMeanVariance(S = s,S1 =s1, S2 = s2, n1 = n, n2 = n)

  ## equivalently...
  # estimated parameters: maximum likelihood estimate
  est.param<-parameters(data = data,classes =classes ,shrink = FALSE)
  testMeanVariance(est.param$S,est.param$S1,est.param$S2,est.param$n1,est.param$n2)
}
```

Index