Package ‘netgsa’

March 27, 2019

Type Package

Title Network-Based Gene Set Analysis

Version 3.1.0

Date 2019-03-12

Author Jing Ma, Ali Shojaie and Kun Yue

Maintainer Jing Ma <jingma@fredhutch.org>

Description Carry out Network-based Gene Set Analysis by incorporating external information about interactions among genes, as well as novel interactions learned from data.

Depends R (>= 3.2.1)

Imports graph, graphite, corpcor, dplyr, glmnet, glassoFast, igraph, Matrix, msigdbr, quadprog, rlang, magrittr

Suggests knitr, MASS, rmarkdown

License GPL (>= 2)

LazyLoad yes

LazyData true

VignetteBuilder knitr

URL https://github.com/drjingma/netgsa

RoxygenNote 6.1.1

NeedsCompilation no

Repository CRAN

Date/Publication 2019-03-27 10:00:10 UTC

R topics documented:

`netgsa-package` *Network-Based Gene Set Analysis*

Description

The netgsa-package provides functions for carrying out Network-based Gene Set Analysis by incorporating external information about interactions among genes, as well as novel interactions learned from data.

Details

Package:	netgsa
Type:	Package
Version:	3.1.0
Date:	2019-03-12
License:	GPL (>=2)

Author(s)

Ali Shojaie <ashojaie@uw.edu> and Jing Ma <jingma@fredhutch.org>

References

Ma, J., Shojaie, A. & Michailidis, G. (2016) Network-based pathway enrichment analysis with incomplete network information. Bioinformatics 32(20):165–3174. `https://doi.org/10.1093/bioinformatics/btw410`

Shojaie, A., & Michailidis, G. (2010a). Penalized likelihood methods for estimation of sparse high-dimensional directed acyclic graphs. Biometrika 97(3), 519-538. `http://biomet.oxfordjournals.org/content/97/3/519.short`

Shojaie, A., & Michailidis, G. (2010b). Network enrichment analysis in complex experiments. Statistical applications in genetics and molecular biology, 9(1), Article 22. `http://www.ncbi.nlm.nih.gov/pubmed/20597848`.

Shojaie, A., & Michailidis, G. (2009). Analysis of gene sets based on the underlying regulatory network. Journal of Computational Biology, 16(3), 407-426. http://www.ncbi.nlm.nih.gov/pmc/articles/PMC3131840/

See Also

`glmnet`

`bic.netEst.undir` *Bayesian information criterion to select the tuning parameters for* `netEst.undir`

Description

This function uses the Bayesian information criterion to select the optimal tuning parameters needed in `netEst.undir`.

Usage

```
bic.netEst.undir(x, zero = NULL, one = NULL, lambda, rho = NULL, weight = NULL,
                 eta = 0, verbose = FALSE, eps = 1e-08)
```

Arguments

`x`	The $p \times n$ data matrix as in `netEst.undir`.
`zero`	(Optional) indices of entries of the matrix to be constrained to be zero. The input should be a matrix of $p \times p$, with 1 at entries to be constrained to be zero and 0 elsewhere. The matrix must be symmetric.
`one`	(Optional) indices of entries of the matrix to be kept regardless of the regularization parameter for lasso. The input is similar to that of `zero` and needs to be symmetric.
`lambda`	(Non-negative) user-supplied lambda sequence.
`rho`	(Non-negative) numeric scalar representing the regularization parameter for estimating the weights in the inverse covariance matrix. This is the same as `rho` in the graphical lasso algorithm `glassoFast`.
`weight`	(Optional) whether to add penalty to known edges. If NULL (default), then the known edges are assumed to be true. If nonzero, then a penalty equal to `lambda * weight` is added to penalize the known edges to account for possible uncertainty. Only non-negative values are accepted for the weight parameter.
`eta`	(Non-negative) a small constant added to the diagonal of the empirical covariance matrix of `X` to ensure it is well conditioned. By default, `eta` is set to 0.
`verbose`	Whether to print out information as estimation proceeds. Default=`FALSE`.
`eps`	Numeric scalar $>=0$, indicating the tolerance level for differentiating zero and non-zero edges: entries $<$ eps will be set to 0.

Details

Let $\hat{\Sigma}$ represent the empirical covariance matrix of data `x`. For a given λ, denote the estimated inverse covariance matrix by $\hat{\Omega}_\lambda$. the Bayesian information criterion (BIC) is defined as

$$trace(\hat{\Sigma}\hat{\Omega}_\lambda) - \log\det(\hat{\Omega}_\lambda) + \frac{\log n}{n} \cdot df,$$

where df represents the degrees of freedom in the selected model and can be estimated via the number of edges in $\hat{\Omega}_\lambda$. The optimal tuning parameter is selected as the one that minimizes the BIC over the range of `lambda`.

Note when the penalty parameter `lambda` is too large, the estimated adjacency matrix may be zero. The function will thus return a warning message.

Value

`lambda`	The values of `lambda` used.
`weight`	The values of `weight` used.
`BIC`	If `weight=NULL`, then a numeric vector of the same length as `lambda` with the corresponding BIC. If `weight` is a vector, then a matrix of size `length(lambda)` by `length(weight)` with the corresponding BIC.
`df`	The degrees of freedom corresponding to each BIC.

Author(s)

Jing Ma

References

Ma, J., Shojaie, A. & Michailidis, G. (2016) Network-based pathway enrichment analysis with incomplete network information. Bioinformatics 32(20):165–3174. https://doi.org/10.1093/bioinformatics/btw410

See Also

`netEst.undir`

Examples

```
library(glassoFast)
library(igraph)

set.seed(1)

## load the data
data(breastcancer2012)

## consider genes from the "ErbB signaling pathway" and "Jak-STAT signaling pathway"
genenames <- unique(c(pathways[[24]], pathways[[52]]))
p <- length(genenames)
```

```
sx <- x[match(genenames, rownames(x)),]
if (sum(is.na(rownames(sx)))>0){
  sx <- sx[-which(is.na(rownames(sx))),]
}
file_e <- system.file("extdata", "edgelist.txt", package = "netgsa")
out <- prepareAdjacencyMatrix(sx, group, pathways, FALSE, file_e, NULL)
sx <- sx[match(colnames(out$B), rownames(sx)),]

ncond <- length(unique(group))
Amat <- vector("list",ncond)

## -- Not run --

# for (k in 1:ncond){
#   data_c <- sx[,(group==k)]
#   fitBIC <- bic.netEst.undir(data_c,one=out$Adj,
#                            lambda=seq(1,10)*sqrt(log(p)/ncol(data_c)),eta=0.1)
#   fit <- netEst.undir(data_c,one=out$Adj,
#                     lambda=which.min(fitBIC$BIC)*sqrt(log(p)/ncol(data_c)),eta=0.1)
#   Amat[[k]] <- fit$Adj
# }
```

breastcancer2012 *Breast cancer data from TCGA (2012).*

Description

An example data set consisting of RNA-seq gene expression data, KEGG pathways, edge list and non-edge list.

Usage

```
data(breastcancer2012)
```

Format

A list with components

- `x` The $p \times n$ data matrix.
- `group` The vector of class indicators of length n.
- `pathways` A list of KEGG pathways.
- `g` A directed acyclic graph corresponding to the Adrenergic signaling in cardiomyocytes pathway from KEGG.
- `edgelist` A data frame of edges, each row corresponding to one edge.
- `nonedgelist` A data frame of nonedges, each row corresponding to one negative edge.

References

Cancer Genome Atlas Network. (2012). Comprehensive molecular portraits of human breast tumours. Nature, 490(7418), 61.

Examples

```
data("breastcancer2012")
```

`edgelist` *A data frame of edges, each row corresponding to one edge*

Description

A data frame of edges, each row corresponding to one edge

Usage

```
edgelist
```

Format

An object of class `data.frame` with 2959 rows and 3 columns.

`g` *A directed acyclic graph corresponding to the Adrenergic signaling in cardiomyocytes pathway from KEGG*

Description

A directed acyclic graph corresponding to the Adrenergic signaling in cardiomyocytes pathway from KEGG

Usage

```
g
```

Format

An object of class `igraph` of length 10.

`group` *The vector of class indicators*

Description

The vector of class indicators

Usage

```
group
```

Format

An object of class `numeric` of length 520.

`netEst.dir` *Constrained estimation of directed networks*

Description

Estimates a directed network using a lasso (L1) penalty.

Usage

```
netEst.dir(x, zero = NULL, one = NULL, lambda, verbose = FALSE, eps = 1e-08)
```

Arguments

`x`	The $p \times n$ data matrix.
`zero`	(Optional) indices of entries of the matrix to be constrained to be zero. The input should be a matrix of $p \times p$, with 1 at entries to be constrained to be zero and 0 elsewhere.
`one`	(Optional) indices of entries of the matrix to be kept regardless of the regularization parameter for lasso. The input is similar to that of `zero`.
`lambda`	(Non-negative) numeric scalar or a vector of length $p-1$ representing the regularization parameters for nodewise lasso. If `lambda` is a scalar, the same penalty will be used for all $p-1$ lasso regressions. By default (`lambda=NULL`), the vector of `lambda` is defined as $$\lambda_j(\alpha) = 2n^{-1/2} Z^*_{\frac{\alpha}{2p(j-1)}}, \quad j = 2, \ldots, p.$$ Here Z^*_q represents the $(1-q)$-th quantile of the standard normal distribution and α is a positive constant between 0 and 1. See Shojaie and Michailidis (2010a) for details on the choice of tuning parameters.
`verbose`	Whether to print out information as estimation proceeds. Default = `FALSE`.
`eps`	(Non-negative) numeric scalar indicating the tolerance level for differentiating zero and non-zero edges: entries with magnitude $<$ `eps` will be set to 0.

Details

The function `netEst.dir` performs constrained estimation of a directed network using a lasso (L1) penalty, as described in Shojaie and Michailidis (2010a). Two sets of constraints determine subsets of entries of the weighted adjacency matrix that should be exactly zero (the option `zero` argument), or should take non-zero values (option `one` argument). The remaining entries will be estimated from data.

The arguments `one` and/or `zero` can come from external knowledge on the 0-1 structure of underlying network, such as a list of edges and/or non-edges learned from available databases. Then the function `prepareAdjacencyMatrix` can be used to first construct `one` and/or `zero`.

In this function, it is assumed that the columns of x are ordered according to a correct (Wald) causal order, such that no x_j is a parent of x_k ($k \leq j$). Given the causal ordering of nodes, the resulting adjacency matrix is lower triangular (see Shojaie & Michailidis, 2010b). Thus, only lower triangular parts of `zero` and `one` are used in this function. For this reason, it is important that both of these matrices are also ordered according to the causal order of the nodes in x. To estimate the network, first each node is regressed on the known edges (`one`). The residual obtained from this regression is then used to find the additional edges, among the nodes that could potentially interact with the given node (those not in `zero`).

This function is closely related to `NetGSA`, which requires the weighted adjacency matrix as input. When the user does not have complete information on the weighted adjacency matrix, but has data (not necessarily the same as the `x` in `NetGSA`) and external information (`one` and/or `zero`) on the adjacency matrix, then `netEst.dir` can be used to estimate the remaining interactions in the adjacency matrix using the data. Further, when it is anticipated that the adjacency matrices under different conditions are different, and data from different conditions are available, the user needs to run `netEst.dir` separately to obtain estimates of the adjacency matrices under each condition.

The algorithm used in `netEst.undir` is based on `glmnet`. Please refer to `glmnet` for computational details.

Value

A list with components

`Adj`	The weighted adjacency matrix of dimension $p \times p$. This is the matrix that will be used in `NetGSA`.
`infmat`	The influence matrix of dimension $p \times p$.
`lambda`	The values of tuning parameters used.

Author(s)

Ali Shojaie

References

Shojaie, A., & Michailidis, G. (2010a). Penalized likelihood methods for estimation of sparse high-dimensional directed acyclic graphs. Biometrika 97(3), 519-538. http://biomet.oxfordjournals.org/content/97/3/519.short

Shojaie, A., & Michailidis, G. (2010b). Network enrichment analysis in complex experiments. Statistical applications in genetics and molecular biology, 9(1), Article 22. http://www.ncbi.nlm.nih.gov/pubmed/20597848.

Shojaie, A., & Michailidis, G. (2009). Analysis of gene sets based on the underlying regulatory network. Journal of Computational Biology, 16(3), 407-426. http://www.ncbi.nlm.nih.gov/pmc/articles/PMC3131840/

See Also

prepareAdjacencyMatrix, glmnet

Examples

```
library(glmnet)
library(graphite)
library(igraph)

set.seed(1)

## load the data
data(breastcancer2012)

print(is_dag(g))

genenames <- V(g)$name
p <- length(genenames)

# reorder the variables and get the adjacency matrix
reOrder <- topo_sort(g,"in")
Adj <- as.matrix(get.adjacency(g))
Adj <- Adj[reOrder,reOrder]

B <- matrix(rep(1,p),nrow=1)
rownames(B) <- "Adrenergic signaling in cardiomyocytes"
colnames(B) <- rownames(Adj)
gx <- x[match(rownames(Adj), rownames(x)),]

Amat <- vector("list", 2)
for (k in 1:2){
  data_c <- gx[,which(group==k)]
  Amat[[k]] <- netEst.dir(data_c, one = Adj)$Adj
}
```

---

netEst.undir *Constrained estimation of undirected networks*

Description

Estimates a sparse inverse covariance matrix using a lasso (L1) penalty.

Usage

```
netEst.undir(x, zero = NULL, one = NULL, lambda, rho=NULL, weight = NULL,
             eta = 0, verbose = FALSE, eps = 1e-08)
```

Arguments

`x`	The $p \times n$ data matrix with rows referring to genes and columns to samples.
`zero`	(Optional) indices of entries of the matrix to be constrained to be zero. The input should be a matrix of $p \times p$, with 1 at entries to be constrained to be zero and 0 elsewhere. The matrix must be symmetric.
`one`	(Optional) indices of entries of the matrix to be kept regardless of the regularization parameter for lasso. The input is similar to that of `zero` and needs to be symmetric.
`lambda`	(Non-negative) numeric scalar representing the regularization parameter for lasso. This algorithm only accepts one `lambda` at a time.
`rho`	(Non-negative) numeric scalar representing the regularization parameter for estimating the weights in the inverse covariance matrix. This is the same as `rho` in the graphical lasso algorithm `glassoFast`.
`weight`	(Optional) whether to add penalty to known edges. If NULL (default), then the known edges are assumed to be true. If nonzero, then a penalty equal to lambda * weight is added to penalize the known edges to account for possible uncertainty. Only non-negative values are accepted for the weight parameter.
`eta`	(Non-negative) a small constant added to the diagonal of the empirical covariance matrix of `X` to ensure it is well conditioned. By default, `eta` is set to 0.
`verbose`	Whether to print out information as estimation proceeds. Default = `FALSE`.
`eps`	(Non-negative) numeric scalar indicating the tolerance level for differentiating zero and non-zero edges: entries with magnitude $<$ `eps` will be set to 0.

Details

The function `netEst.undir` performs constrained estimation of sparse inverse covariance (concentration) matrices using a lasso (L1) penalty, as described in Ma, Shojaie and Michailidis (2016). Two sets of constraints determine subsets of entries of the inverse covariance matrix that should be exactly zero (the option `zero` argument), or should take non-zero values (option `one` argument). The remaining entries will be estimated from data.

The arguments `one` and/or `zero` can come from external knowledge on the 0-1 structure of underlying concentration matrix, such as a list of edges and/or non-edges learned from available databases. Then the function `prepareAdjacencyMatrix` can be used to first construct `one` and/or `zero`.

`netEst.undir` estimates both the support (0-1 structure) of the concentration matrix, or equivalently, the adjacency matrix of the corresponding Gaussian graphical model, for a given tuning parameter, `lambda`; and the concentration matrix with diagonal entries set to 0, or equivalently, the weighted adjacency matrix. The weighted adjacency matrix is estimated using maximum likelihood based on the estimated support. The parameter `rho` controls the amount of regularization used in the maximum likelihood step. A small `rho` is recommended, as a large value of `rho` may result in too much regularization in the maximum likelihood estimation, thus further penalizing the support of the weighted adjacency matrix. Note this function is suitable only for estimating the adjacency matrix of a undirected graph. The `weight` parameter allows one to specify whether to penalize the known edges. If known edges obtained from external information contain uncertainty such that some of them are spurious, then it is recommended to use a small positive `weight` parameter to select the most probable edges from the collection of known ones.

This function is closely related to `NetGSA`, which requires the weighted adjacency matrix as input. When the user does not have complete information on the weighted adjacency matrix, but has data (`x`, not necessarily the same as the `x` in `NetGSA`) and external information (`one` and/or `zero`) on the adjacency matrix, then `netEst.undir` can be used to estimate the remaining interactions in the adjacency matrix using the data. Further, when it is anticipated that the adjacency matrices under different conditions are different, and data from different conditions are available, the user needs to run `netEst.undir` separately to obtain estimates of the adjacency matrices under each condition.

The algorithm used in `netEst.undir` is based on `glmnet` and `glasso`. Please refer to `glmnet` and `glasso` for computational details.

Value

A list with components

`Adj`	The weighted adjacency matrix (partial correlations) of dimension $p \times p$, with diagonal entries set to 0. This is the matrix that will be used in `NetGSA`.
`invcov`	The estimated inverse covariance matrix of dimension $p \times p$.
`lambda`	The values of tuning parameters used.

Author(s)

Jing Ma

References

Ma, J., Shojaie, A. & Michailidis, G. (2016) Network-based pathway enrichment analysis with incomplete network information. Bioinformatics 32(20):165–3174. https://doi.org/10.1093/bioinformatics/btw410

See Also

`prepareAdjacencyMatrix`, `bic.netEst.undir`, `glmnet`

Examples

```
library(glassoFast)
library(graphite)
library(igraph)

set.seed(1)

## load the data
data(breastcancer2012)
```

```
## consider genes from the "ErbB signaling pathway" and "Jak-STAT signaling pathway"
genenames <- unique(c(pathways[[24]], pathways[[52]]))
p <- length(genenames)
sx <- x[match(genenames, rownames(x)),]
if (sum(is.na(rownames(sx)))>0){
  sx <- sx[-which(is.na(rownames(sx))),]
}
file_e <- system.file("extdata", "edgelist.txt", package = "netgsa")
out <- prepareAdjacencyMatrix(sx, group, pathways, FALSE, file_e, NULL)
sx <- sx[match(colnames(out$B), rownames(sx)),]

ncond <- length(unique(group))
Amat <- vector("list",ncond)

## -- Not run --
# for (k in 1:ncond){
#   data_c <- sx[,(group==k)]
#   fitBIC <- bic.netEst.undir(data_c,one=out$Adj,
#                              lambda=seq(1,10)*sqrt(log(p)/ncol(data_c)),eta=0.1)
#   fit <- netEst.undir(data_c,one=out$Adj,
#                       lambda=which.min(fitBIC$BIC)*sqrt(log(p)/ncol(data_c)),eta=0.1)
#   Amat[[k]] <- fit$Adj
# }
```

NetGSA *Network-based Gene Set Analysis*

Description

Tests the significance of pre-defined sets of genes (pathways) with respect to an outcome variable, such as the condition indicator (e.g. cancer vs. normal, etc.), based on the underlying biological networks.

Usage

```
NetGSA(A, x, group, pathways, lklMethod = c("REML", "ML", "HE", "REHE"),
       sampling=FALSE, sample_n = NULL, sample_p = NULL, minsize=5,
       eta = 0.1, lim4kappa = 500)
```

Arguments

`A`	A list of weighted adjacency matrices.
`x`	The $p \times n$ data matrix with rows referring to genes and columns to samples. It is very important that the adjacency matrices `A` share the same rownames as the data matrix `x`.
`group`	Vector of class indicators of length n.
`pathways`	The npath by p indicator matrix for pathways.

lklMethod	Method used for variance component calculation: options are ML (maximum likelihood), REML (restricted maximum likelihood), HE (Haseman-Elston regression) or REHE (restricted Haseman-Elston regression). See details.
sampling	(Logical) whether to subsample the observations and/or variables. See details.
sample_n	The ratio for subsampling the observations if sampling=TRUE.
sample_p	The ratio for subsampling the variables if sampling=TRUE.
minsize	Minimum number of genes in pathways to be considered.
eta	Approximation limit for the Influence matrix. See 'Details'.
lim4kappa	Limit for condition number (used to adjust eta). See 'Details'.

Details

The function NetGSA carries out a Network-based Gene Set Analysis, using the method described in Shojaie and Michailidis (2009) and Shojaie and Michailidis (2010). It can be used for gene set (pathway) enrichment analysis where the data come from K heterogeneous conditions, where K, or more. NetGSA differs from Gene Set Analysis (Efron and Tibshirani, 2007) in that it incorporates the underlying biological networks. Therefore, when the networks encoded in A are empty, one should instead consider alternative approaches such as Gene Set Analysis (Efron and Tibshirani, 2007).

The NetGSA method is formulated in terms of a mixed linear model. Let X represent the rearrangement of data x into an $np \times 1$ column vector.

$$X = \Psi\beta + \Pi\gamma + \epsilon$$

where β is the vector of fixed effects, γ and ϵ are random effects and random errors, respectively. The underlying biological networks are encoded in the weighted adjacency matrices, which determine the influence matrix under each condition. The influence matrices further determine the design matrices Ψ and Π in the mixed linear model. Formally, the influence matrix under each condition represents the effect of each gene on all the other genes in the network and is calculated from the adjacency matrix (A[[k]] for the k-th condition). A small value of eta is used to make sure that the influence matrices are well-conditioned (i.e. their condition numbers are bounded by lim4kappa.)

The problem is then to test the null hypothesis $\ell\beta = 0$ against the alternative $\ell\beta \neq 0$, where ℓ is a contrast vector, optimally defined through the underlying networks. For a one-sample or two-sample test, the test statistic T for each gene set has approximately a t-distribution under the null, whose degrees of freedom are estimated using the Satterthwaite approximation method. When analyzing complex experiments involving multiple conditions, often multiple contrast vectors of interest are considered for a specific subnetwork. Alternatively, one can combine the contrast vectors into a contrast matrix L. A different test statistic F will be used. Under the null, F has an F-distribution, whose degrees of freedom are calculated based on the contrast matrix L as well as variances of γ and ϵ. The fixed effects β are estimated by generalized least squares, and the estimate depends on estimated variance components of γ and ϵ.

Estimation of the variance components (σ_ϵ^2 and σ_γ^2) can be done in several different ways after profiling out σ_ϵ^2, including REML/ML which uses Newton's method or HE/REHE which is based on the Haseman-Elston regression method. The latter notes the fact that $Var(X) = \sigma_\gamma^2 \Pi * \Pi' + \sigma_\epsilon^2 I$, and uses an ordinary least squares to solve for the unknown coefficients after vectorizing both sides. In particular, REHE uses nonnegative least squares for the regression and therefore ensures nonnegative estimate of the variance components. Due to the simple formulation, `HE/REHE` also allows subsampling with respect to both the samples and the variables, and is recommended especially when the problem is large (i.e. large p and/or large n).

The pathway membership information is stored in `pathways`, which should be a matrix of $npath$ x p. See `prepareAdjacencyMatrix` for details on how to prepare a suitable pathway membership object.

This function can deal with both directed and undirected networks, which are specified via the option `directed`. Note `NetGSA` uses slightly different procedures to calculate the influence matrices for directed and undirected networks. In either case, the user can still apply `NetGSA` if only partial information on the adjacency matrices is available. The functions `netEst.undir` and `netEst.dir` provide details on how to estimate the weighted adjacency matrices from data based on available network information.

Value

A list with components

`results`	A data frame with pathway names, pathway sizes, p-values and false discovery rate corrected q-values for all pathways.
`beta`	Vector of fixed effects of length $2p$, of which the first half is for condition 1 and the second half for condition 2.
`s2.epsilon`	Variance of the random errors ϵ.
`s2.gamma`	Variance of the random effects γ.

Author(s)

Ali Shojaie and Jing Ma

References

Ma, J., Shojaie, A. & Michailidis, G. (2016) Network-based pathway enrichment analysis with incomplete network information. Bioinformatics 32(20):165–3174. `https://doi.org/10.1093/bioinformatics/btw410`

Shojaie, A., & Michailidis, G. (2010). Network enrichment analysis in complex experiments. Statistical applications in genetics and molecular biology, 9(1), Article 22. `http://www.ncbi.nlm.nih.gov/pubmed/20597848`.

Shojaie, A., & Michailidis, G. (2009). Analysis of gene sets based on the underlying regulatory network. Journal of Computational Biology, 16(3), 407-426. `http://www.ncbi.nlm.nih.gov/pmc/articles/PMC3131840/`

See Also

`prepareAdjacencyMatrix`, `netEst.dir`, `netEst.undir`

Examples

```
library(glassoFast)
library(glmnet)
library(igraph)

set.seed(1)
data(breastcancer2012)


## -----------------Undirected networks-----------------
## consider genes from the "ErbB signaling pathway" and "Jak-STAT signaling pathway"
genenames <- unique(c(pathways[[24]], pathways[[52]]))
p <- length(genenames)
sx <- x[match(genenames, rownames(x)),]
if (sum(is.na(rownames(sx)))>0){
  sx <- sx[-which(is.na(rownames(sx))),]
}
file_e <- system.file("extdata", "edgelist.txt", package = "netgsa")
out <- prepareAdjacencyMatrix(sx, group, pathways, FALSE, file_e, NULL)
sx <- sx[match(colnames(out$B), rownames(sx)),]

ncond <- length(unique(group))
Amat <- vector("list",ncond)

## -- Not run --
# for (k in 1:ncond){
#  data_c <- sx[,(group==k)]
#  fitBIC <- bic.netEst.undir(data_c,one=out$Adj,
#                             lambda=seq(1,10)*sqrt(log(p)/ncol(data_c)),eta=0.1)
#  fit <- netEst.undir(data_c,one=out$Adj,
#                      lambda=which.min(fitBIC$BIC)*sqrt(log(p)/ncol(data_c)),eta=0.1)
#  Amat[[k]] <- fit$Adj
# }
# test <- NetGSA(Amat, sx, group, pathways = out$B, lklMethod = 'REHE')

## -------------------Directed networks------------------
## NetGSA also works for directed acyclic graphs (DAGs).
# e.g. the "Adrenergic signaling in cardiomyocytes" pathway from KEGG is a DAG.
print(is_dag(g))

genenames <- V(g)$name
p <- length(genenames)

# reorder the variables and get the adjacency matrix
reOrder <- topo_sort(g,"in")
Adj <- as.matrix(get.adjacency(g))
Adj <- Adj[reOrder,reOrder]

B <- matrix(rep(1,p),nrow=1)
rownames(B) <- "Adrenergic signaling in cardiomyocytes"
colnames(B) <- rownames(Adj)
gx <- x[match(rownames(Adj), rownames(x)),]
```

```
Amat <- vector("list", 2)
# for (k in 1:2){
#   data_c <- gx[,which(group==k)]
#   Amat[[k]] <- netEst.dir(data_c, one = Adj)$Adj
# }
# test <- NetGSA(Amat, gx, group, pathways = B, lklMethod = 'REHE')
```

`nonedgelist` *A data frame of nonedges, each row corresponding to one negative edge*

Description

A data frame of nonedges, each row corresponding to one negative edge

Usage

```
nonedgelist
```

Format

An object of class `data.frame` with 500 rows and 3 columns.

`pathways` *A list of KEGG pathways*

Description

A list of KEGG pathways

Usage

```
pathways
```

Format

An object of class `list` of length 100.

prepareAdjacencyMatrix

Construct adjacency matrices from existing databases or user provided network information

Description

Read the network information from KEGG or specified by the user and construct the adjacency matrices needed for NetGSA.

Usage

```
prepareAdjacencyMatrix(x, group, pways, import_from_kegg=FALSE,
       file_e=c(NA, file_e), file_ne=c(NULL, file_ne, NA),
       estimate_network=FALSE, lambda_c=1, eta=0.5, minsize=5, fileEncoding="")
```

Arguments

`x`	The $p \times n$ data matrix with rows referring to genes and columns to samples.
`group`	Vector of class indicators of length n.
`pways`	A list of pathways from `preparePathways`.
`import_from_kegg`	(Logical) whether to import network information from KEGG.
`file_e`	The name of the file which the list of edges is to read from. This should be a .txt file, with one edge in a line, the two vertices separated by a delimiter. The third column indicates the direction of the edge: directed or undirected.
`file_ne`	The name of the file which the list of negative edges is to read from. The edges in this file are negative in the sense that the corresponding vertices are not connected. This should be a .txt file, with one negative edge in a line, the two vertices separated by a delimiter. The third column indicates the direction of the negative edge: directed or undirected. If `NA`, the input edge list from `file_e` will be treated as complete network information and edge weights will be estimated only for the input edge list. If `NULL`, the input edge list from `file_e` will be treated as partial information and a network estimation procedure will be used to refit the network topology and edge weights. See details.
`estimate_network`	(Logical) whether to estimate the weighted adjacency matrices. Default is `FALSE`. Users are recommended to use the returned 0-1 adjacency matrix and the function `netEst.undir` and `netEst.dir` to estimate the weighted adjacency matrices.
`lambda_c`	(Non-negative) a constant multiplied to the tuning parameter `lambda` needed for estimating the edge weights. By default, `lambda_c` is set to 1. See `netEst.undir` and `netEst.dir` for more details.
`eta`	(Non-negative) a small constant needed for estimating the edge weights. By default, `eta` is set to 0.5. See `netEst.undir` for more details.

`minsize`	Minimum number of genes in pathways to be considered.
`fileEncoding`	Character string: if non-empty declares the encoding used on a file (not a connection) so the character data can be re-encoded. See the 'Encoding' section of the help for file, and 'R Data Import/Export Manual'.

Details

The function `prepareAdjacencyMatrix` accepts both network information from user specified sources and KEGG, and return 0-1 adjacency matrices needed for `netEst.undir` and `netEst.dir`. If `estimate_network=TRUE`, the function also returns a list of weighted adjacency matrices that can be directly used in `NetGSA`. Note if the dimension of the problem, or equivalently the total number of unique genes across all pathways, is large, `prepareAdjacencyMatrix` with `estimate_network=TRUE` may be a bit slow.

If `file_ne=NA`, the input edge list from `file_e` will be treated as complete network information and edge weights will be estimated only for the input edge list. If `file_ne=NULL`, the input edge list from `file_e` will be treated as partial information and a network estimation procedure will be used to refit the network topology and edge weights. When importing network information from KEGG, the network information can also be treated as complete if `file_ne=NA` or incomplete if `file_ne=NULL`. The information in `file_e` and `file_ne` should be compatible in the sense that the same edge should not appear in both files. An error will be reported if incompatibility occurs.

When `estimate_network=FALSE`, the returned 0-1 adjacency matrices (for edges or non-edges) can be used as input in `netEst.undir` or `netEst.dir` to estimate the complete (and weighted) adjacency matrices under a variety of tuning parameters. The user can choose the set of tuning parameters that achieve the best balance in model fit and model complexity. When `estimate_network=TRUE`, `prepareAdjacencyMatrix` also returns the weighted adjacency matrices suitable for input in `NetGSA` using a fixed set of tuning parameters, as specified via `lambda_c` and `eta`.

Value

A list with components

`Amat`	A list of weighted adjacency matrices.
`Adj`	A list of 0-1 adjacency matrices corresponding to the edges.
`Zero_Adj`	A list of 0-1 adjacency matrices corresponding to the negative edges.
`B`	The npath by p indicator matrix for pathways.

Author(s)

Jing Ma

References

Ma, J., Shojaie, A. & Michailidis, G. (2016) Network-based pathway enrichment analysis with incomplete network information. Bioinformatics 32(20):165–3174.

See Also

`NetGSA`, `netEst.dir`, `netEst.undir`

Examples

```
library(glassoFast)
library(igraph)

set.seed(1)

## load the data
data("breastcancer2012")

## consider genes from the "ErbB signaling pathway" and "Jak-STAT signaling pathway"
genenames <- unique(c(pathways[[24]], pathways[[52]]))
p <- length(genenames)
sx <- x[match(genenames, rownames(x)),]
if (sum(is.na(rownames(sx)))>0){
  sx <- sx[-which(is.na(rownames(sx))),]
}
# compare the resulting matrices when file_ne=NULL vs file_ne=NA
file_e <- system.file("extdata", "edgelist.txt", package = "netgsa")
out1 <- prepareAdjacencyMatrix(sx, group, pathways, FALSE, file_e, NULL)
out2 <- prepareAdjacencyMatrix(sx, group, pathways, FALSE, file_e, NA)
```

---

preparePathways *Prepare pathway dataset needed by NetGSA*

Description

Prepare pathway dataset needed by NetGSA. See NetGSA for more details.

Usage

```
preparePathways(db=c("kegg", "MSigDB"),
            type=c("H","C1","C2","C3","C4","C5","C6","C7"),
            genename= c("EntrezID", "symbol"))
```

Arguments

db	Database to build pathway from. Could be either 'kegg' or 'MSigDB'.
type	The type of pathways to choose from if db=='MSigDB'.
genename	Whether gene symbol or EntrezID is used.

Value

A list of pathways.

Author(s)

Jing Ma (jingma@fredhutch.org)

See Also

`NetGSA`, `prepareAdjacencyMatrix`

Examples

```
#library(graphite)

#pathwayList <- preparePathways('kegg')
#pathwayList[[1]]
```

`x` *Data matrix p by n*

Description

Data matrix p by n

Usage

```
x
```

Format

An object of class `matrix` with 2598 rows and 520 columns.

Index