# Package 'netgsa'

December 7, 2021

**Type** Package

**Title** Network-Based Gene Set Analysis

**Version** 4.0.3

**Date** 2021-12-07

**Description** Carry out network-based gene set analysis by incorporating external information about interactions among genes, as well as novel interactions learned from data. Implements methods described in Shojaie A, Michailidis G (2010) <doi:10.1093/biomet/asq038>, Shojaie A, Michailidis G (2009) <doi:10.1089/cmb.2008.0081>, and Ma J, Shojaie A, Michailidis G (2016) <doi:10.1093/bioinformatics/btw410>.

**Depends** R (>= 3.5.0)

**biocViews**

**Imports** AnnotationDbi, corpcor, data.table, dplyr, genefilter, graph, graphite, glmnet, glassoFast, httr, igraph, jsonlite, magrittr, Matrix, msigdbr, org.Hs.eg.db, quadprog, RCy3, reshape2, rlang, Rcpp (>= 1.0.2)

**Suggests** knitr, MASS, ndexr, rmarkdown

**License** GPL (>= 3)

**LinkingTo** Rcpp, RcppEigen

**LazyLoad** yes

**LazyData** true

**VignetteBuilder** knitr

**URL** https://github.com/mikehellstern/netgsa

**RoxygenNote** 7.1.1

**NeedsCompilation** yes

**Author** Michael Hellstern [aut, cre],
Ali Shojaie [aut],
Jing Ma [aut],
Kun Yue [aut]

**Maintainer** Michael Hellstern <mikeh1@uw.edu>

**Repository** CRAN

**Date/Publication** 2021-12-07 19:40:02 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. doi: 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. https://academic.oup.com/biomet/article-abstract/97/3/519/243918

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

Shojaie, A., & Michailidis, G. (2009). Analysis of gene sets based on the underlying regulatory network. Journal of Computational Biology, 16(3), 407-426. https://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 $\lambda$, 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. doi: 10.1093/bioinformatics/btw410

## See Also

netEst.undir

---

`breastcancer2012_subset`

*Breast cancer data from TCGA (2012). This is a 750 gene subset*

---

## Description

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

## Usage

```
data(breastcancer2012_subset)
```

## 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.

`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.

`pathways_mat` Matrix with pathway indicators

## References

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

## Examples

```
data("breastcancer2012_subset")
```

## 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 19 rows and 4 columns.

## formatPathways *Format cytoscape nested networks*

### Description

Format cytoscape nested networks using preset NetGSA format

### Usage

```
formatPathways(x, pways, graph_layout = NULL)
```

### Arguments

| | |
|---|---|
| `x` | A NetGSA object returned from calling `NetGSA()` |
| `pways` | Character vector of pathways to format |
| `graph_layout` | (Optional) Layout to pass to plots. Must be a string for Cytoscape which will be passed to `RCy3::layoutNetwork`. |

### Details

Loads gene testing data into each pathway. Genes are tested using an F-test if there are 2 or more conditions or a two-sided one-class t-test against the null hypothesis of mean = 0 if there is only one condition. FDR corrected q-values are mapped to the color of the node. The scale ranges from 0 to 1 with red represents q-values of 0 and white representing q-values of 1. Loaded data includes: p-value from the F-test/t-test (pval), FDR corrected q-value (pFdr), test statistic from the F-test/t-test (teststat).

Custom formatting can be applied using the cytoscape GUI or the RCy3 pacakge.

### Value

No return value, called for side effects

## Author(s)

Michael Hellstern

## References

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

## See Also

plot.NetGSA

## Examples

```
## Not run:
## load the data
data("breastcancer2012_subset")

## consider genes from just 2 pathways
genenames    <- unique(c(pathways[["Adipocytokine signaling pathway"]],
                         pathways[["Adrenergic signaling in cardiomyocytes"]]))
sx           <- x[match(rownames(x), genenames, nomatch = 0L) > 0L,]

db_edges       <- obtainEdgeList(rownames(sx), databases = c("kegg", "reactome"))
adj_cluster    <- prepareAdjMat(sx, group, databases = db_edges, cluster = TRUE)
out_cluster    <- NetGSA(adj_cluster[["Adj"]], sx, group,
                  pathways_mat[c(1,2), rownames(sx)], lklMethod = "REHE", sampling = FALSE)
plot(out_cluster)
formatPathways(out_netgsa, "Adipocytokine signaling pathway")

## End(Not run)
```

---

# 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 $\alpha$ 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.

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. https://academic.oup.com/biomet/article-abstract/97/3/519/243918

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

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

## See Also

prepareAdjMat, glmnet

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,
             penalize_diag = TRUE, 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 weighted adjacency 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 weighted adjacency 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 vector representing the regularization parameters for lasso. Can choose best based on BIC using bic.netEst.undir |
| rho | (Non-negative) numeric scalar or symmetric $p \times p$ matrix 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. |
| penalize_diag | Logical. Whether or not to penalize diagonal entries when estimating weighted adjacency matrix. If TRUE a small penalty is used, otherwise no penalty is used. |
| 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.

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 | List of weighted adjacency matrices (partial correlations) of dimension $p \times p$, with diagonal entries set to 0. Each element in the list is the weighted adjacency matric corresponding to each value in lambda. Each element is a matrix that will be used in NetGSA. |
| invcov | List of estimated inverse covariance matrix of dimension $p \times p$. |
| lambda | List of values of tuning parameters used. |

## Author(s)

Jing Ma & Michael Hellstern

## References

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

## See Also

prepareAdjMat, bic.netEst.undir, glmnet

## Examples

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

set.seed(1)

## load the data
data(breastcancer2012_subset)

## consider genes from the "Estrogen signaling pathway" and "Jak-STAT signaling pathway"
genenames <- unique(c(pathways[[25]], pathways[[52]]))
sx <- x[match(genenames, rownames(x)),]
if (sum(is.na(rownames(sx)))>0){
  sx <- sx[-which(is.na(rownames(sx))),]
}
p <- length(genenames)

## zero/one matrices should be based on known non-edges/known edges. Random used as an example
one <- matrix(sample(c(0,1), length(rownames(sx))**2,
              replace = TRUE, prob = c(0.9, 0.1)), length(rownames(sx)),
              dimnames = list(rownames(sx), rownames(sx)))

ncond <- length(unique(group))
Amat <- vector("list",ncond)
for (k in 1:ncond){
    data_c <- sx[,(group==k)]
    fitBIC <- bic.netEst.undir(data_c,one=one,
                               lambda=seq(1,10)*sqrt(log(p)/ncol(data_c)),eta=0.1)
    fit <- netEst.undir(data_c,one=one,
                        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 = "REHE",
       sampling=FALSE, sample_n = NULL, sample_p = NULL, minsize=5,
       eta = 0.1, lim4kappa = 500)
```

## Arguments

| | |
|---|---|
| A | A list of weighted adjacency matrices. Typically returned from prepareAdjMat |
| 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 $\beta$ is the vector of fixed effects, $\gamma$ and $\epsilon$ 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 $\Psi$ and $\Pi$ 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 $\ell$ 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 $\gamma$ and $\epsilon$. The fixed effects $\beta$ are estimated by generalized least squares, and the estimate depends on estimated variance components of $\gamma$ and $\epsilon$.

Estimation of the variance components ($\sigma^2_\epsilon$ and $\sigma^2_\gamma$) can be done in several different ways after profiling out $\sigma^2_\epsilon$, 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 $Var(X) = \sigma^2_\gamma \Pi * \Pi' + \sigma^2_\epsilon I$, 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 `prepareAdjMat` 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, and test statistic for all pathways. |
| `beta` | Vector of fixed effects of length $kp$, the first k elements corresponds to condition 1, the second k to condition 2, etc |
| `s2.epsilon` | Variance of the random errors $\epsilon$. |
| `s2.gamma` | Variance of the random effects $\gamma$. |
| `graph` | List of components needed in `plot.NetGSA`. |

## 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. doi: 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. https://pubmed.ncbi.nlm.nih.gov/20597848/.

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

## See Also

prepareAdjMat, netEst.dir, netEst.undir

## Examples

```
## load the data
data("breastcancer2012_subset")

## consider genes from just 2 pathways
genenames    <- unique(c(pathways[["Adipocytokine signaling pathway"]],
                         pathways[["Adrenergic signaling in cardiomyocytes"]]))
sx           <- x[match(rownames(x), genenames, nomatch = 0L) > 0L,]

db_edges       <- obtainEdgeList(rownames(sx), databases = c("kegg", "reactome"))
adj_cluster    <- prepareAdjMat(sx, group, databases = db_edges, cluster = TRUE)
out_cluster    <- NetGSA(adj_cluster[["Adj"]], sx, group,
                  pathways_mat[c(1,2), rownames(sx)], lklMethod = "REHE", sampling = FALSE)
```

---

NetGSAq *"Quick" Network-based Gene Set Analysis*

---

## Description

Quick version of NetGSA

## Usage

```
NetGSAq(x, group, pathways, lambda_c = 1, file_e = NULL, file_ne = NULL,
    lklMethod="REHE", cluster = TRUE, sampling = TRUE, sample_n = NULL,
    sample_p = NULL, minsize=5, eta=0.1, lim4kappa=500)
```

## Arguments

| | |
|---|---|
| x | See x argument in NetGSA |
| group | See group argument in NetGSA |
| pathways | See pathways argument in NetGSA |
| lambda_c | See lambda_c argument in prepareAdjMat |

| | |
|---|---|
| file_e | See file_e argument in prepareAdjMat |
| file_ne | See file_ne argument in prepareAdjMat |
| lklMethod | See lklMethod argument in NetGSA |
| cluster | See cluster argument in prepareAdjMat |
| sampling | See sampling argument in NetGSA |
| sample_n | See sample_n argument in NetGSA |
| sample_p | See sample_p argument in NetGSA |
| minsize | See minsize argument in NetGSA |
| eta | See eta argument in NetGSA |
| lim4kappa | See lim4kappa argument in NetGSA |

## Details

This is a wrapper function to perform weighted adjacency matrix estimation and pathway enrichment in one step. For more details see ?prepareAdjMat and ?NetGSA.

## Value

A list with components

| | |
|---|---|
| results | A data frame with pathway names, pathway sizes, p-values and false discovery rate corrected q-values, and test statistic for all pathways. |
| beta | Vector of fixed effects of length $kp$, the first k elements corresponds to condition 1, the second k to condition 2, etc. |
| s2.epsilon | Variance of the random errors $\epsilon$. |
| s2.gamma | Variance of the random effects $\gamma$. |
| graph | List of components needed in plot.NetGSA. |

## Author(s)

Michael Hellstern

## References

Ma, J., Shojaie, A. & Michailidis, G. (2016) Network-based pathway enrichment analysis with incomplete network information. Bioinformatics 32(20):165–3174. doi: 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. https://pubmed.ncbi.nlm.nih.gov/20597848/.

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

## See Also

prepareAdjMat, netEst.dir, netEst.undir

## Examples

```
# Example takes ~3 minutes to run depending on computer
## load the data
data("breastcancer2012_subset")

## consider genes from just 2 pathways
genenames    <- unique(c(pathways[["Adipocytokine signaling pathway"]],
                         pathways[["Adrenergic signaling in cardiomyocytes"]]))
sx           <- x[match(rownames(x), genenames, nomatch = 0L) > 0L,]

out_clusterq <- NetGSAq(sx, group, pathways_mat[c(1, 2), rownames(sx)])
```

---

# 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 1 rows and 4 columns.

---

# obtainClusters *Estimate optimal gene clustering structure*

---

## Description

Tries six different clustering methods and chooses the one with the best results. This is a helper function in `prepareAdjMat` and should not be called by the user.

## Usage

```
obtainClusters(A, order, cluster)
```

### Arguments

| | |
|---|---|
| A | A 0-1 adjacency matrix |
| order | Final ordering of genes/metabs to be consistent with order you passed data in |
| cluster | Whether or not to cluster (TRUE/FALSE). We always cluster connected components, but if cluster = TRUE we cluster further |

### Details

This function tries the six different clustering methods in igraph and chooses the best one. As stated in prepareAdjMat the six methods evaluated are: cluster_walktrap, cluster_leading_eigen, cluster_fast_greedy, cluster_label_prop, cluster_infomap, and cluster_louvain. See prepareAdjMat for how the best is chosen. Even if cluster = FALSE, connected components of the 0-1 adjacency matrix are used as clusters.

It is essential that the order of the returned named numeric vector must be in the same order as the rows of the data matrix.

### Value

Named numeric vector of membership. The name of each element is the corresponding gene and the value is the cluster it belongs to.

### Author(s)

Michael Hellstern

### References

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

### See Also

prepareAdjMat

---

## obtainEdgeList *Obtain edgelist from graphite databases. To be used within prepareAdjMat*

---

### Description

Find all edges between genes in the specified graphite databases.

### Usage

```
obtainEdgeList(genes, databases)
```

## Arguments

| | |
|---|---|
| genes | Character vector of gene ID and gene value. The ID and gene value should be separated by a colon. E.g. "ENTREZID:127550". It is very important to have these separated by a colon since obtainEdgeList uses regular expressions to split this into gene value and gene ID. |
| databases | Character vector of graphite databases you wish to search for edges. Options are: biocarta, kegg, nci, panther, pathbank, pharmgkb, reactome, smpdb, ndex. Note NDEx is recommended for expert users and is only available for the development version of netgsa (https://github.com/mikehellstern/netgsa), see details. |

## Details

obtainEdgeList searches through the specified databases to find edges between genes in the genes argument. Since one can search in multiple databases with different identifiers, genes are converted using AnnotationDbi::select and metabolites are converted using graphite:::metabolites(). Databases are also used to specify non-edges. This function searches through graphite databases and also has the option to search NDEx (public databases only). However, since NDEx is open-source and does not contain curated edge information like graphite, NDEx database search is a beta function and is only recommended for expert users. When searching through NDEx, gene identifiers are not converted. Only, the gene identifiers passed to the genes argument are used to search through NDEx. NDEx contains some very large networks with millions of edges and extracting those of interest can be slow.

This function is particularly useful if the user wants to create an edgelist outside of prepareAdjMat. graphite and it's databases are constantly updated. Creating and storing an edgelist outside of prepareAdjMat may help reproducibility as this guarantees the same external information is used. It can also speed up computation since if only a character vector of databases is passed to prepareAdjMat, it calls obtainEdgeList each time and each call can take several minutes. The edges from obtainEdgeList are used to create the 0-1 adjacency matrices used in netEst.undir and netEst.dir.

Using obtainEdgeList to generate edge information is highly recommended as this performs all the searching and conversion of genes to common identifiers. Inclusion of additional edges, removal of edges, or other user modifications to edgelists should be through the file_e and file_ne arguments in prepareAdjMat.

## Value

A list of class obtainedEdgeList with components

| | |
|---|---|
| edgelist | A data.table listing the edges. One row per edge. Edges are assumed to be directed. So if an edge is undirected there will be two rows. |
| genes_not_in_dbs | A vector of genes specified, but were not found in the databases searched |

## Author(s)

Michael Hellstern

## See Also

prepareAdjMat, netEst.dir, netEst.undir

## Examples

```
genes <- paste0("ENTREZID:", c("10000", "10298", "106821730",
                               "10718", "1398", "1399", "145957",
                               "1839", "1950", "1956"))

out <- obtainEdgeList(genes, c("kegg", "reactome"))
```

---

# pathways *A list of KEGG pathways*

---

## Description

A list of KEGG pathways

## Usage

```
pathways
```

## Format

An object of class `list` of length 100.

---

# pathways_mat *Matrix with pathway indicators*

---

## Description

Matrix with pathway indicators

## Usage

```
pathways_mat
```

## Format

An object of class `matrix` (inherits from `array`) with 51 rows and 250 columns.

plot.NetGSA *Generates NetGSA plots*

## Description

Generates network plots in Cytoscape and igraph

## Usage

```
## S3 method for class 'NetGSA'
plot(x, graph_layout = NULL, rescale_node = c(2,10), rescale_label = c(0.5,0.6), ...)
```

## Arguments

| | |
|---|---|
| x | An object of class "NetGSA" returned from calling `NetGSA()` |
| graph_layout | (Optional) Layout to pass to plots. Either a function for igraph plots (when Cytoscape not open) or a string for Cytoscape. The igraph function should only take one parameter (an igraph object). See `igraph::layout_` for more details. For example one might create a custom layout by setting the spring.length and spring.constant with: `my_layout <-function(graph) layout_with_graphopt(graph = graph,spring.length = 1000,spring.constant = 0.00004)`. The string for Cytoscape will be passed to `RCy3::layoutNetwork`. |
| rescale_node | (Optional) Node size rescaling to pass to igraph plots. Must be a vector of length 2 with the first element being the minimum node size and the second being the maximum. |
| rescale_label | (Optional) Label size rescaling to pass to igraph plots. Must be a vector of length 2 with the first element being the minimum node size and the second being the maximum. |
| ... | Other arguments not used by this method |

## Details

One of two options can occur.

(1) If Cytoscape is open on the user's computer, a nested network will be created. The main network is the interactions between pathways. In this graph, there is one node for each pathway. An edge is drawn between pathways if there is at least one edge between genes of each pathway. That is if gene A is in pathway 1 and gene B is in pathway 2, pathway 1 and pathway 2 will have an edge if gene A and gene B have an edge. Note self-edges are not drawn. The value of the test statistic is mapped to node color. Large negative values of the test statistic are orange, values around 0 are white and large positive values are blue. FDR corrected q-values are mapped to the border color of the node. The scale ranges from 0 to 1 with red representing q-values of 0 and white representing q-values of 1. Pathway size is mapped to node size so pathways with more genes are larger. Each pathway node is also linked to its network of genes so the user can see individual gene interactions within a pathway. These can be accessed by right clicking the node -> Nested Networks -> Go To Nested Network. Alternatively, the corresponding nested network has the same name as the pathway so the user can click on the network directly in the Control Panel/Network menu. It is important to

note that `plot.NetGSA` generates default plots and loads in data into Cytoscape, but the user can customize the plots however they like using RCy3 or the Cytoscape GUI directly.

To save time, the nested networks are not formatted. One can apply NetGSA's formatting using `formatPathways`

For custom formatting, the node data that is loaded into Cytoscape includes the pathway results from NetGSA: Pathway size (pSize), p-value (pval), FDR corrected q-value (pFDR), test statistic (teststat) and pathway name. The edge data loaded into Cytoscape is: total number of edges between two pathways (weight). For example weight of 10 between pathway 1 and pathway 2 means there are 10 edges between the genes of pathway 1 and the genes of pathway 2.

There are two R plots also generated. The first is the legend for Cytoscape. The legend shows the mapping for node color (test statistic) and node border color (FDR corrected q-value). This is generated in R because there does not seem to be a reliable way to plot the legend for the main network (interactions between pathways). The second plot is a plot of the main network created in igraph. It mimics the Cytoscape plot as closely as possible. NetGSA exports the x and y coordinates of the nodes in the Cytoscape layout and uses them in the igraph layout. Custom layouts can be passed to this using the `graph_layout` argument. The user can also zoom-in on individual pathways in igraph using the `zoomPathway` function.

(2) If Cytoscape is not open, the `igraph::rglplot` function is used to plot the main network (interactions between pathways). The default layout used is `layout_on_sphere`, but custom layouts can be specified with the `graph_layout` argument. The other plot generated is the legend since it is difficult to plot on `rglplot`.

## Value

No return value, called for plotting

## Author(s)

Michael Hellstern

## 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

## Examples

```
## Not run:
## load the data
data("breastcancer2012_subset")

## consider genes from just 2 pathways
genenames    <- unique(c(pathways[["Adipocytokine signaling pathway"]],
                         pathways[["Adrenergic signaling in cardiomyocytes"]]))
sx           <- x[match(rownames(x), genenames, nomatch = 0L) > 0L,]
```

```
db_edges       <- obtainEdgeList(rownames(sx), databases = c("kegg", "reactome"))
adj_cluster    <- prepareAdjMat(sx, group, databases = db_edges, cluster = TRUE)
out_cluster    <- NetGSA(adj_cluster[["Adj"]], sx, group,
                  pathways_mat[c(1,2), rownames(sx)], lklMethod = "REHE", sampling = FALSE)
plot(out_cluster)

## End(Not run)
```

---

`prepareAdjMat` *Construct adjacency matrices from graphite databases and/or user provided network information*

---

## Description

Read the network information from any of the graphite databases specified by the user and construct the adjacency matrices needed for NetGSA. This function also allows for clustering. See details for more information

## Usage

```
prepareAdjMat(x, group, databases = NULL, cluster = TRUE,
       file_e=NULL, file_ne=NULL, lambda_c=1, penalize_diag=TRUE, eta=0.5)
```

## Arguments

| | |
|---|---|
| `x` | The $p \times n$ data matrix with rows referring to genes and columns to samples. Row names should be unique and have gene ID types appended to them. The id and gene number must be separated by a colon. E.g. "ENTREZID:127550" |
| `group` | Vector of class indicators of length $n$. Identifies the condition for each of the $n$ samples |
| `databases` | (Optional) Either (1) the result of a call to `obtainEdgeList` or (2) a character vector of graphite databases you wish to search for edges. Since one can search in multiple databases with different identifiers, converts genes using `AnnotationDbi::select` and convert metabolites using `graphite:::metabolites()`. Databases are also used to specify non-edges. If `NULL` no external database information will be used. See Details for more information |
| `cluster` | (Optional) Logical indicating whether or not to cluster genes to estimate adjacency matrix. If not specified, set to TRUE if there are > 2,500 genes (p > 2,500). The main use of clustering is to speed up calculation time. If the dimension of the problem, or equivalently the total number of unique genes across all pathways, is large, `prepareAdjMat` may be slow.<br>If clustering is set to TRUE, the 0-1 adjacency matrix is used to detect clusters of genes within the connected components. Once gene clusterings are chosen, the weighted adjacency matrices are estimated for each cluster separately using `netEst.undir` or `netEst.dir`. Thus, the adjacency matrix for the full network is block diagonal with the blocks being the adjacency matrices from the clusters. |

Any edges between clusters are set to 0, so this can be thought of as an approximate weighted adjacency matrix. Six clustering algorithms from the `igraph` package are considered: `cluster_walktrap`, `cluster_leading_eigen`, `cluster_fast_greedy`, `cluster_label_prop`, `cluster_infomap`, and `cluster_louvain`. Clustering is performed on each connected component of size >1,000 genes. To ensure increases in speed, algorithms which produce a maximum cluster size of < 1,000 genes are considered first. Among those, the algorithm with the smallest edge loss is chosen. If all algorithms have a maximum cluster size > 1,000 genes the one with the smallest maximum cluster size is chosen. Edge loss is defined as the number of edges between genes of different clusters. These edges are "lost" since they are set to 0 in the block diagonal adjacency matrix.

If clustering is set to FALSE, the 0-1 adjacency matrix is used to detect connected components and the weighted adjacency matrices are estimated for each connected component.

Singleton clusters are combined into one cluster. This should not affect performance much since the gene in a singleton cluster should not have any edges to other genes.

`file_e` (Optional) The name of the file which the list of edges is to read from. This file is read in with `data.table::fread`. Must have 4 columns in the following order. The columns do not necessarily need to be named, but they must be in this specific order:

- 1st column - Source gene (base_gene_src), e.g. "7534""
- 2nd column - Gene identifier of the source gene (base_id_src), e.g. "ENTREZID"
- 3rd column - Destination gene (base_gene_dest), e.g. "8607"
- 4th column - Gene identifier of the destination gene (base_id_dest) e.g. "UNIPROT"

This information cannot conflict with the user specified non-edges. That is, one cannot have the same edge in `file_e` and `file_ne`. In the case where the graph is undirected everything will be converted to an undirected edge or non-edge. Thus if the user specifies A->B as a directed non-edge it will be changed to an undirected non-edge if the graph is undirected. See Details for more information.

`file_ne` (Optional) The name of the file which the list of non-edges is to read from. This file is read in with `data.table::fread`. The edges in this file are negative in the sense that the corresponding vertices are not connected. Format of the file must be the same as `file_e`. Again, each observation is assumed to be a directed edge. Thus for a negative undirected edge, input two separate negative edges.

In the case of conflicting information between `file_ne` and edges identified in a database, user non-edges are used. That is if the user specifies A->B in `file_ne`, but there is an edge between A->B in KEGG, the information in KEGG will be ignored and A->B will be treated as a non-edge. In the case where the graph is undirected everything will be converted to an undirected edge or non-edge. Thus if the user specifies A->B as a directed non-edge it will be changed to an undirected non-edge if the graph is undirected. See Details for more information.

`lambda_c` (Non-negative) a vector or constant. `lambda_c` is multiplied by a constant depending on the data to determine the actual tuning parameter, `lambda`, used in estimating the network. If `lambda_c` is a vector, the optimal `lambda` will be chosen from this vector using `bic.netEst.undir`. Note that `lambda` is only used if the network is undirected. If the network is directed, the default value in `netEst.dir` is used instead . By default, `lambda_c` is set to 1. See `netEst.undir` and `netEst.dir` for more details.

`penalize_diag` Logical. Whether or not to penalize diagonal entries when estimating weighted adjacency matrix. If TRUE a small penalty is used, otherwise no penalty is used.

`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.

## Details

The function `prepareAdjMat` accepts both network information from user specified sources as well as a list of graphite databases to search for edges in. `prepareAdjMat` calculates the 0-1 adjacency matrices and runs `netEst.undir` or `netEst.dir` if the graph is undirected or directed.

When searching for network information, `prepareAdjMat` makes some important assumptions about edges and non-edges. As already stated, the first is that in the case of conflicting information, user specified non-edges are given precedence.

`prepareAdjMat` uses `obtainEdgeList` to standardize and search the `graphite` databases for edges. For more information see `?obtainEdgeList`. `prepareAdjMat` also uses database information to identify non-edges. If two genes are identified in the `databases` edges but there is no edge between them this will be coded as a non-edge. The rationale is that if there was an edge between these two genes it would be present.

`prepareAdjMat` assumes no information about genes not identified in `databases` edgelists. That is, if the user passes gene A, but gene A is not found in any of the edges in `databases` no information about Gene A is assumed. Gene A will have neither edges nor non-edges.

Once all the network and clustering information has been compiled, `prepareAdjMat` estimates the network. `prepareAdjMat` will automatically detect directed graphs, rearrange them to the correct order and use `netEst.dir` to estimate the network. When the graph is undirected `netEst.undir` will be used. For more information on these methods see `?netEst.dir` and `?netEst.undir`.

Importantly, `prepareAdjMat` returns the list of weighted adjacency matrices to be used as an input in `NetGSA`.

## Value

A list with components

`Adj` A list of weighted adjacency matrices estimated from either `netEst.undir` or `netEst.dir`. That is `length(Adj) = length(unique(group))`. One list of weighted adjacency matrix will be returned for each condition in group. If cluster = TRUE is specified, the length of the list of adjacency matrices for each condition will be the same length as the number of clusters. The structure of Adj is Adj[[condition_number]][[cluster_adj_matrix]]. Note that even when `cluster = FALSE` the connected components are used as clusters. The last element which is needed for plotting and is passed through to the output of `NetGSA` is `edgelist`.

| | |
|---|---|
| invcov | A list of inverse covariance matrices estimated from either netEst.undir or netEst.dir. That is length(invcov) = length(unique(group)). One list of inverse covariance matrix will be returned for each condition in group. If cluster = TRUE is specified, the length of the list of inverse covariance matrices for each condition will be the same length as the number of clusters. The structure of invcov is invcov[[condition_number]][[cluster_adj_matrix]] |
| lambda | A list of values of tuning parameters used for each condition in group. If cluster = TRUE is specified, the length of the list of tuning parameters for each condition will be the same length as the number of clusters. |

## Author(s)

Michael Hellstern

## 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

```
## load the data
data("breastcancer2012_subset")

## consider genes from just 2 pathways
genenames    <- unique(c(pathways[[1]], pathways[[2]]))
sx           <- x[match(rownames(x), genenames, nomatch = 0L) > 0L,]

adj_cluster    <- prepareAdjMat(sx, group,
                                databases = c("kegg", "reactome"),
                                cluster = TRUE)
adj_no_cluster <- prepareAdjMat(sx, group,
                                databases = c("kegg", "reactome"),
                                cluster = FALSE)
```

---

# stackDatabases *Combine edges from databases into a data.table*

---

## Description

Retrieves edges from specified databases and stacks them into one data.table.This is a helper function in prepareAdjMat and should not be called by the user.

## Usage

```
stackDatabases(databases)
```

## Arguments

| | |
|---|---|
| databases | Character vector of databases to compile. Should be one of the options from hspaiens in `graphite::pathwayDatabases()` |

## Details

This function compiles all the edges from all databases specified into one data.table

## Value

A data.table with columns:

| | |
|---|---|
| database | Which database the edge comes from |
| src | Source gene |
| src_type | Source gene identifier type |
| dest | Destination gene |
| dest_type | Destination gene identifier type |
| direction | Direction of edge. Either Directed or Undirected |

## Author(s)

Michael Hellstern

## References

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

## See Also

obtainEdgeList

---

# x *Data matrix p by n*

---

## Description

Data matrix p by n

## Usage

```
x
```

## Format

An object of class `matrix` (inherits from `array`) with 250 rows and 520 columns.

---

`zoomPathway` *Zoom in on pathway in igraph*

---

## Description

Plots the gene to gene interactions for a given pathway in igraph.

## Usage

```
zoomPathway(x, pway, graph_layout = NULL)
```

## Arguments

| | |
|---|---|
| `x` | A NetGSA object returned from calling `NetGSA()` |
| `pway` | Name of pathway to plot |
| `graph_layout` | (Optional) Layout function to pass to igraph plots. This function should only take one parameter (an igraph object). For example one might create a custom layout by setting the spring.length and spring.constant with: `my_layout <-function(graph) layout_with_graphopt(graph = graph,spring.length = 1000,spring.constant = 0.00004)` |

## Details

Generates igraph plot for gene to gene interactions for a given pathway

## Value

No return value, called for side effects

## Author(s)

Michael Hellstern

## References

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

## See Also

`plot.NetGSA`

## Examples

```
## Not run:
## load the data
data("breastcancer2012_subset")

## consider genes from just 2 pathways
genenames    <- unique(c(pathways[["Adipocytokine signaling pathway"]],
                         pathways[["Adrenergic signaling in cardiomyocytes"]]))
sx           <- x[match(rownames(x), genenames, nomatch = 0L) > 0L,]

db_edges       <- obtainEdgeList(rownames(sx), databases = c("kegg", "reactome"))
adj_cluster    <- prepareAdjMat(sx, group, databases = db_edges, cluster = TRUE)
out_cluster    <- NetGSA(adj_cluster[["Adj"]], sx, group,
                  pathways_mat[c(1,2), rownames(sx)], lklMethod = "REHE", sampling = FALSE)
plot(out_cluster)
my_layout <- function(graph) layout_with_graphopt(graph = graph,
                                                  spring.length = 1000,
                                                  spring.constant = 0.00004)
zoomPathway(out_cluster, "Adipocytokine signaling pathway", my_layout)

## End(Not run)
```

# Index