Package 'netgsa'

June 16, 2016

Type Package

Title Network-Based Gene Set Analysis

Version 3.0

Date 2016-06-15

Author Ali Shojaie and Jing Ma

Maintainer Jing Ma <jinma@upenn.edu>

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 corpcor, Matrix, glasso, glmnet, igraph

Suggests MASS

License GPL (>= 2)

LazyLoad yes

URL http://arxiv.org/abs/1411.7919

NeedsCompilation no

Repository CRAN

Date/Publication 2016-06-16 18:27:48

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:	3.0
Date:	2016-05-11
License:	GPL (>=2)

Author(s)

Ali Shojaie <ashojaie@uw.edu> and Jing Ma <mjing@umich.edu>

References

Ma, J., Shojaie, A. & Michailidis, G. (2014). Network-based pathway enrichment analysis with incomplete network information, submitted. http://arxiv.org/abs/1411.7919.

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

glasso, 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 $n \times p$ data matrix as in `covsel`.
`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.
`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. (2014). Network-based pathway enrichment analysis with incomplete network information, submitted. http://arxiv.org/abs/1411.7919.

See Also

netEst.undir

Examples

```
set.seed(1)
library(MASS)
library(glmnet)
library(glasso)

## Generate the covariance matrix for the AR(1) process
rho <- 0.5
p <- 100
n <- 100
Sigma <- diag(rep(1,p))
Sigma <- rho^(abs(row(Sigma)-col(Sigma)))/(1-rho^2)

## The inverse covariance matrix is sparse
Omega <- solve(Sigma)

## Generate multivariate normal data n by p
X <- mvrnorm(n, mu=rep(0, p), Sigma=Omega)

## Select the tuning parameters
score = bic.netEst.undir(X, lambda = seq(0.1,1,0.1))
```

edgelist *An example edge list*

Description

An example edge list corresponding to known connections in a network.

Usage

```
data(edgelist)
```

Format

A matrix with two columns, each row defining one edge.

Examples

```
data(edgelist)
```

edgelist2adj *Construct an adjacency matrix from an edge list*

Description

Read the edge list of a graph from a file and construct the adjacency matrix.

Usage

```
edgelist2adj(file, vertex.names, mode = c("directed", "undirected"))
```

Arguments

file	The connection to read from. This should be a .txt file, with one edge in a line, the two vertices separated by a delimiter.
vertex.names	The names of all vertices in the graph.
mode	Whether the graph to read is directed.

Details

The function edgelist2adj accepts a list of edges and constructs the 0-1 adjacency matrix corresponding to the graph. If file contains an incomplete list of edges, the function determines the actual size of the graph through the vector vertex.names.

Although named as edgelist2adj, the user can also construct a 0-1 matrix corresponding to non-edges, i.e. connections that do not exist between any two vertices.

When file contains incomplete information, the returned 0-1 adjacency matrix (for edges or non-edges) can be used as input in covsel to estimate the complete (and weighted) adjacency matrix.

Value

A 0-1 adjacency matrix of dimension $p \times p$ corresponding to the edges defined in `file`, where p is the length of the vector `vertex.names`.

Author(s)

Ali Shojaie and Jing Ma

References

Ma, J., Shojaie, A. & Michailidis, G. (2014). Network-based pathway enrichment analysis with incomplete network information, submitted. http://arxiv.org/abs/1411.7919.

See Also

netEst.dir,netEst.undir

Examples

```
#Read the data
data(edgelist)
data(nonedgelist)

#Generate the .txt files
write.table(edgelist, file="edgelist.txt", row.names=FALSE)
write.table(nonedgelist, file="nonedgelist.txt", row.names=FALSE)

#Read the edge/nonedge list from files
oneMat = edgelist2adj(file="edgelist.txt", vertex.names=paste0("gene", 1:100),
         mode="undirected")
zeroMat = edgelist2adj(file="nonedgelist.txt", vertex.names=paste0("gene", 1:100),
          mode="undirected")
```

---

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 $n \times p$ 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 frm available databases. Then the function edgelist2adj can be used to first construct one and/or zero.

In this function, it is assumed that the columns of X are ordered accoring 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 reisdual 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 (X, 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

edgelist2adj, glmnet

Examples

```
library(MASS)
library(glmnet)
set.seed(1)

p <- 100 # number of variables
s <- 0.1 # probability of having an edge in the adjacency matrix
rho <- 0.6 # edge weights

## form an adjacency matrix
A <- matrix(rbinom(p*p,1,s),p,p)
A[upper.tri(A)] = 0

I <- diag(rep(1,p))

## generate data (see Shojaie & Michaildis, 2010b)
n <- 100
```

```
X <- solve(I-rho*A)
X <- t(X)
X <- scale(X)

zeros <- matrix(0,p,p)
ones <- matrix(0,p,p)

zeros[5,1] <- zeros[10,1] <- 1
ones[4,1] <- ones[11,2] <- 1

fit.dir <- netEst.dir(X=X, zero=zeros, one=ones, lambda=0.1)
```

`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 $n \times p$ 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. 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.
`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 (concerntration) matrices using a lasso (L1) penalty, as described in Ma, Shojaie and Michailidis (2014). 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 concerntration matrix, such as a list of edges and/or non-edges learned frm available databases. Then the function edgelist2adj can be used to first construct one and/or zero.

netEst.undir estimates both the support (0-1 structure) of the concerntration matrix, or equivalently, the adjacency matrix of the corresponding Gaussian graphical model, for a given tuning parameter, lambda; and the concerntration 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. (2014). Network-based pathway enrichment analysis with incomplete network information, submitted. http://arxiv.org/abs/1411.7919.

See Also

edgelist2adj,bic.netEst.undir, glmnet, glasso

Examples

```
library(MASS)
library(glmnet)
library(glasso)
set.seed(1)

## Generate the covariance matrix for the AR(1) process
rho <- 0.5
p <- 100
n <- 100
Sigma <- diag(rep(1,p))
Sigma <- rho^(abs(row(Sigma)-col(Sigma)))/(1-rho^2)

## The inverse covariance matrix is sparse
Omega <- solve(Sigma)

## Generate multivariate normal data
X <- mvrnorm(n, mu=rep(0, p), Sigma=Omega)

## Estimate the network without external information
fit <- netEst.undir(X, lambda = 0.2)

## Estimate the network with external information
##-Not run-
ones = edgelist2adj(file="edgelist.txt", vertex.names=paste0("gene", 1:p),
mode="undirected")
zeros = edgelist2adj(file="nonedgelist.txt", vertex.names=paste0("gene", 1:p),
mode="undirected")

fit.undir <- netEst.undir(X, zero=zeros, one=ones, lambda = 0.5)

## Estimate the network when the known edges are not entirely reliable.
ones[1,10:11] = 1
ones[10:11,1] = 1
fit.undir <- netEst.undir(X, zero=zeros, one=ones, weight = 0.1, lambda = 0.2)
```

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 network.

Usage

```
NetGSA(A, x, y, B, lklMethod = c("REML", "ML"), directed = FALSE, eta = 0.1,
       lim4kappa = 500)
```

Arguments

A	A list of weighted adjacency matrices.
x	The $p \times n$ data matrix.
y	Vector of class indicators of length n.
B	The npath by p indicator matrix for pathways.
lklMethod	Method used for likelihood calculation: options are ML (maximum likelihood) or REML (restricted maximum likelihood).
directed	Whether the networks are directed. By default, directed=FALSE.
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 differs from Gene Set Analysis (Efron and Tibshirani, 2007) in that it incorporates the underlying biological networks.

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 A, 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 $\ell\beta = 0$ vs. the alterernative $\ell\beta \neq 0$, where ℓ is a contrast vector, optimally defined through the underlying networks. For a two-sample test, the test statistic T for each gene set is a function of β, variances of γ and ϵ, the constrast vector ℓ and the underlying biological network(s) in both conditions. Under the null hypothesis, T has approximately a t-distribution, 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 estimates of the variance components of γ and ϵ. The

variance components (σ^2_ϵ and σ^2_γ) are estimated using Newton's method based on the profiling out σ_ϵ.

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 the case of undirected networks, the user can still apply `NetGSA` if only partial information on the adjacency matrices is available. The function `covsel` provides one way to estimate the weighted adjacency matrices from data based on available network information.

Value

A list with components

`beta`	Vector of fixed effects of length $2p$, of which the first half is for condition 1 and the second half for conditin 2.
`teststat`	Test statistics for gene sets (pathways).
`df`	Degrees of freedom for the test statistics.
`p.value`	P-values for gene sets (pathways).
`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. (2014). Network-based pathway enrichment analysis with incomplete network information, submitted. http://arxiv.org/abs/1411.7919

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

`edgelist2adj`, `netEst.dir`, `netEst.undir`

Examples

```
set.seed(1)

## NetGSA with directed networks

## NetGSA with undirected networks
data(netgsaex2)
```

```
A = netgsaex2$A
B = netgsaex2$B
x = netgsaex2$x
y = netgsaex2$y

# -Not-run-
# fit = NetGSA(A, x, y, B, lklMethod="REML", directed=FALSE)
```

`netgsaex2` *Toy example for using* `NetGSA` *on a two-class test with undirected networks*

Description

This dataset contains an example for using Network-based Gene Set Analysis on a two-sample test with undirected networks.

Usage

```
data("netgsaex2")
```

Format

A list with components

`A` A list ot two weighted adjacency matrices.

`x` The $p \times n$ data matrix.

`y` The vector of class indicators of length n.

`B` The npath by p indicator matrix for pathways.

References

Ma, J., Shojaie, A. & Michailidis, G. (2014). Network-based pathway enrichment analysis with incomplete network information. http://arxiv.org/abs/1411.7919

Examples

```
data(netgsaex2)
A = netgsaex2$A
B = netgsaex2$B
x = netgsaex2$x
y = netgsaex2$y

# -Not-run-
# fit = NetGSA(A, x, y, B, lklMethod="REML", directed=FALSE)
```

netgsaex3 *Toy example for using* NetGSA *on a three-class test with undirected networks*

Description

This dataset contains an example for using Network-based Gene Set Analysis on a multi-sample test with undirected networks.

Usage

```
data("netgsaex3")
```

Format

A list with components

- A A list ot three weighted adjacency matrices.
- x The $p \times n$ data matrix.
- y The vector of class indicators of length n.
- B The npath by p indicator matrix for pathways.

References

Ma, J., Shojaie, A. & Michailidis, G. (2014). Network-based pathway enrichment analysis with incomplete network information. http://arxiv.org/abs/1411.7919.

Examples

```
data(netgsaex3)
A = netgsaex3$A
B = netgsaex3$B
x = netgsaex3$x
y = netgsaex3$y

# -Not-run-
# fit = NetGSA(A, x, y, B, lklMethod="REML", directed=FALSE)
```

netgsaexDAG *Toy example for using* NetGSA *on a two-class test with directed acyclic graphs*

Description

This dataset contains an example for using Network-based Gene Set Analysis on a two-sample test with directed acyclic graphs.

Usage

```
data("netgsaexDAG")
```

Format

A list with components

- A A list ot two weighted adjacency matrices.
- x The $p \times n$ data matrix.
- y The vector of class indicators of length n.
- B The npath by p indicator matrix for pathways.

References

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/

Examples

```
data(netgsaexDAG)
A = netgsaexDAG$A
B = netgsaexDAG$B
x = netgsaexDAG$x
y = netgsaexDAG$y
fitDAG = NetGSA(A, x, y, B, lklMethod="REML", directed=TRUE)
```

nonedgelist *An example for not-an-edge list*

Description

An example for not-an-edge list corresponding to the connections that do not exist in a network.

Usage

```
data(nonedgelist)
```

Format

A matrix with two columns, each row defining one pair of vertices that is not an edge in the network.

Examples

```
data(nonedgelist)
```

Index