Package ‘netgsa’

March 8, 2015

Type Package
Title Network-Based Gene Set Analysis
Version 2.0
Date 2015-03-07
Author Ali Shojaie and Jing Ma
Maintainer Jing Ma <mjing@umich.edu>
Description The netgsa-package contains 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.
Depends cvTools, corpcor, glasso, glmnet, igraph
Suggests MASS
License GPL (>= 2)
LazyLoad yes
URL http://arxiv.org/abs/1411.7919
NeedsCompilation no
Repository CRAN
Date/Publication 2015-03-08 07:18:14

R topics documented:

  netgsa-package .................................................. 2
covsel ................................................................. 2
cv.covsel ............................................................ 5
edgelist ............................................................... 6
edgelist2adj ......................................................... 7
NetGSA ................................................................. 8
netgsaex .............................................................. 10
nonedgelist ......................................................... 11

Index 12
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**: 2.0
- **Date**: 2015-03-07
- **License**: GPL (>=2)

**Author(s)**

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

**References**


**See Also**

- glasso, glmnet

---

covsel  

**Covariance selection**

**Description**

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

covsel(X, zero = NULL, one = NULL, lambda, rho=0.01, 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 regular-
ization 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.

rho
(Non-negative) numeric scalar representing the regularization parameter for es-
timating the weights in the inverse covariance matrix.

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

The function covsel 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 underly-
ing 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.

covsel 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.

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 covsel 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 conditions 1
and 2 are different, and data from both conditions are available, the user needs to run covsel twice
to obtain estimates of the adjacency matrices under each condition.
The algorithm used in cvsel is based on glmnet and glasso. Please refer to glmnet and glasso for computational details.

Value

A list with components

Adj The estimated adjacency matrix of dimension $p \times p$.

wAdj The estimated weighted adjacency matrix of dimension $p \times p$, with diagonal entries set to 0.

Author(s)

Jing Ma

References


See Also

edgelist2adj, cv.covsel, glmnet, glasso

Examples

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

## Generate the covariance matrix for the AR(1) process
phi <- 0.5
p <- 50
n <- 50
Sigma <- diag(rep(1,p))
Sigma <- phi^abs(row(Sigma)-col(Sigma))/(1-phi^2)

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

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

## Covariance selection without external information
fit <- cvsel(x, lambda = 0.2)

## Covariance selection with external information
##-Not run-
#oneMat = edgelist2adj(file="edgelist.txt", vertex.names=1:p, mode="undirected")
#zeroMat = edgelist2adj(file="nonedgelist.txt", vertex.names=1:p, mode="undirected")
#fit2 <- cvsel(x, zero=zeroMat, one=oneMat, lambda = 0.2)
```
cv.covsel

Cross-validation for covsel

Description

Performs k-fold cross validation for covsel.

Usage

cv.covsel(x, zero = NULL, one = NULL, lambda, nfolds = NULL, verbose = FALSE)

Arguments

x
The n × 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 × 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.

nfolds
Number of folds. Default is 5-fold cross validation.

verbose
Whether to print out information as estimation proceeds. Default=FALSE.

Details

The function splits the data x into nfolds and runs covsel nfolds times to compute the fit with each of the folds omitted. The error is accumulated and the average error over the folds is computed.

Value

A list with components

lambda
The values of lambda used.

cve
The mean cross-validated error(s), a vector of length(lambda).

Author(s)

Jing Ma

References

See Also

covsel

Examples

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

## Generate the covariance matrix for the AR(1) process
phi <- 0.5
p <- 50
n <- 50
Sigma <- diag(rep(1,p))
Sigma <- phi^(abs(row(Sigma)-col(Sigma))/(1-phi^2))

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

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

## Covariance selection with external information
cv.fit <- cv.covsel(x, lambda=seq(0.1,0.3,0.1), nfolds=5)

datasets 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**

```r
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**


**See Also**

`covsel`
Examples

```r
#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=1:50, mode="undirected")
zeroMat = edgelist2adj(file="nonedgelist.txt", vertex.names=1:50, mode="undirected")
```

---

**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**

```r
NetGSA(A1, A2, x, y, B, lklMethod = c("REML", "ML"), directed = FALSE, eta = 0.1, lim4kappa = 500)
```

**Arguments**

- **A1** The weighted adjacency matrix for condition 1.
- **A2** The weighted adjacency matrix for condition 2.
- **x** The $p \times n$ data matrix.
- **y** Vector of class indicators of length $n$.
- **B** The $npath \times p$ 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.
- **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 $\beta$ 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 $a_1$ and $a_R$, 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 ($a_1$ or $a_2$). A small value of $\text{eta}$ is used to make sure that the influence matrices are well-conditioned (i.e. their condition numbers are bounded by $\lim \kappa$).

The problem is then to test the null hypothesis $\ell \beta = 0$ vs. the alternative $\ell \beta \neq 0$, where $\ell$ is a contrast vector, optimally defined through the underlying networks. The test statistic $T$ for each gene set is then a function of $\beta$, variances of $\gamma$ and $\epsilon$, the contrast vector $\ell$ 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.

The fixed effects $\beta$ are estimated by generalized least squares, and the estimate depends on estimates of the variance components of $\gamma$ and $\epsilon$. The variance components ($\sigma_\epsilon^2$ and $\sigma_\gamma^2$) are estimated using Newton’s method based on the profiling out $\sigma_\epsilon$.

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 condition 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 $\epsilon$.
- **s2.gamma**: Variance of the random effects $\gamma$.

Author(s)

Ali Shojaie and Jing Ma
References


See Also

edgelist2adj, covsel

Examples

```r
set.seed(1)
library(igraph)
data(netgsaex)
A1 = netgsaex$a1
A2 = netgsaex$a2
B = netgsaex$b
x = netgsaex$x
y = netgsaex$y

## Visualize the networks
par(mar = c(0, 0.5, 3, 0.5))
plot(netgsaex$g.alt, vertex.size = 5, vertex.label = NA, main="Network - alt")

par(mar = c(0.5, 0.5, 3, 0.5))
plot(netgsaex$g.null, vertex.size = 5, vertex.label = NA, main="Network - null")

out = NetGSA(A1, A2, x, y, B, lklMethod = "REML")
out2 = NetGSA(A1, A2, x, y, B, lklMethod = "ML")
```

netgsaex

Toy example for using NetGSA on a two-class test

Description

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

Usage

data(netgsaex)
nonedgelist

Format

A list with components

A1 The weighted adjacency matrix for condition 1.
A2 The weighted adjacency matrix for condition 2.
x The p × n data matrix.
y The vector of class indicators of length n.
B The npath by p indicator matrix for pathways.
g.null, g.alt The networks under the null and alternative hypothesis, respectively. Node colors in the two graphs represent different mean values. Note the network structures between g.null and g.alt differ slightly.

References


Examples

data(netgsaex)

data(nonedgelist)

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

*Topic datasets
  edgelist, 6
  netgsaex, 10
  nonedgelist, 11
*Topic package
  netgsa-package, 2

covsel, 2, 6, 7, 10
cv.covsel, 4, 5

edgelist, 6
edgelist2adj, 4, 7, 10

glasso, 2, 4
glmnet, 2, 4

NetGSA, 8
netgsa-package, 2
netgsaex, 10
nonedgelist, 11