Package ‘mapggm’

October 6, 2015

Type Package

Title Multi-attribute network construction and perturbation detection

Version 1.0

Date 2015-09-30

Author Paula J Griffin

Maintainer Paula J Griffin <paulajgriffin@gmail.com>

Description
Companion package to 'Detection of multiple perturbations in multi-omics biological networks'

License Apache License

Imports igraph, Matrix

R topics documented:

blockNorms *Get Frobenius norms of submatrices with optional weights*

Description

Get Frobenius norms of submatrices with optional weights

Usage

```
blockNorms(M, id, W = NULL)
```

Arguments

M	square matrix of interest
id	vector grouping elements of M
W	optional weights (square matrix, dimensions equal to length(id))

Value

matrix of Frobenius norms of submatrices

findPerturbations *Get node-wise test statistics*

Description

Get node-wise test statistics

Usage

```
findPerturbations(Y, Omega, Sigma, id, sequential = FALSE)
```

Arguments

Y	matrix or data frame (rows=subjects, columns=variables)
Omega	precision matrix
Sigma	covariance matrix
id	vector mapping variables to nodes
sequential	boolean, whether or not to perform sequential adjustments

Value

test statistics for perturbations at each individual node

`getAIC` *Get AIC of a given network configuration*

Description

Get AIC of a given network configuration

Usage

```
getAIC(S, n, Omega, id)
```

Arguments

`S`	sample covariance matrix
`n`	number of samples
`Omega`	estimated precision
`id`	vector grouping elements of S, Omega

Value

Akaike information criterion for model implied by Omega

`getBIC` *Get BIC of a given network configuration*

Description

Get BIC of a given network configuration

Usage

```
getBIC(S, n, Omega, id)
```

Arguments

`S`	sample covariance matrix
`n`	number of samples
`Omega`	estimated precision
`id`	vector grouping elements of S, Omega

Value

Bayesian information criterion for model implied by Omega

getDual *Get value of dual function (called by maLasso1)*

Description

Get value of dual function (called by maLasso1)

Usage

```
getDual(Sigma)
```

Arguments

`Sigma`	current estimated covariance

Value

value of dual function for optimization problem

getEBIC *Get extended BIC of a given network configuration*

Description

Get extended BIC of a given network configuration

Usage

```
getEBIC(S, n, Omega, id, gamma = 0.5)
```

Arguments

`S`	sample covariance matrix
`n`	number of samples
`Omega`	estimated precision
`id`	vector grouping elements of S, Omega
`gamma`	EBIC parameter (default 0.5)

Value

extended Bayesian information criterion for model implied by Omega

`getHammingDist` *Get Hamming distance of a given network configuration (truth required)*

Description

Get Hamming distance of a given network configuration (truth required)

Usage

```
getHammingDist(Omega, Theta, id)
```

Arguments

`Omega`	estimated precision
`Theta`	true network graph of joint nodes
`id`	vector grouping elements of Omega

Value

Hamming distance between Theta and graph implied by Omega

`getLambdas` *Generate lambdas to try*

Description

Generate penalty parameters spanning a range of 1 to n.nodes submatrix blocks

Usage

```
getLambdas(S, id, length.out = 10)
```

Arguments

`S`	sample covariance
`id`	vector of node identifiers
`length.out`	number of lambda parameters to return

Value

vector of length.out equally spaced lambdas

getPrimal *Get value of primal function (called by maLasso1)*

Description

Get value of primal function (called by maLasso1)

Usage

```
getPrimal(Omega, S, W, id, lambda)
```

Arguments

`Omega`	current estimated precision
`S`	sample covariance matrix
`W`	weight matrix
`id`	vector grouping elements of S, Omega
`lambda`	penalty parameter

Value

value of primal function for optimization problem

groupCondLRTs *Sequentially test for perturbations at a series of locations*

Description

Sequentially test for perturbations at a series of locations

Usage

```
groupCondLRTs(Y, Omega, Sigma, perturb.mat, ret = "stat")
```

Arguments

`Y`	matrix or data frame (rows=subjects, columns=variables)
`Omega`	precision matrix
`Sigma`	covariance matrix
`perturb.mat`	matrix indicating sets of which variables to test for perturbations
`ret`	return statistics ('stat') or p-values ('pval')

Value

test statistics for perturbations at each of perturb.mat rows

groupLRT *Test for a perturbation at a specific location*

Description

Test for a perturbation at a specific location

Usage

```
groupLRT(Y, Omega, Sigma, perturb.vec, ret = "stat")
```

Arguments

`Y`	matrix or data frame (rows=subjects, columns=variables)
`Omega`	precision matrix
`Sigma`	covariance matrix
`perturb.vec`	vector indicating which variables to test for perturbations
`ret`	value to return ('stat' or 'pval')

Value

test statistic for a perturbation at perturb.vec=TRUE locations only

groupLRTs *Test for perturbations at a series of locations*

Description

Test for perturbations at a series of locations

Usage

```
groupLRTs(Y, Omega, Sigma, perturb.mat, ret = "stat")
```

Arguments

`Y`	matrix or data frame (rows=subjects, columns=variables)
`Omega`	precision matrix
`Sigma`	covariance matrix
`perturb.mat`	matrix indicating sets of which variables to test for perturbations
`ret`	value to return ('stat' or 'pval')

Value

test statistics for perturbations at each of perturb.mat rows

`isComputationallyPD` *Check if a matrix is computationally positive-definite*

Description

Check if a matrix is computationally positive-definite

Usage

```
isComputationallyPD(M)
```

Arguments

`M`	matrix of interest

Value

TRUE if matrix is computationally positive-definite

`maLasso` *Multi-attribute network estimation*

Description

Estimates block precisions and covariances for a multi-attribute network based on a Gaussian graphical model with zero mean vector.

Usage

```
maLasso(S, n, id, lambda, W = NULL, update = 100, max.gap = 0.5,
  max.iter = 100, min.t = .Machine$double.eps)
```

Arguments

`S`	sample covariance matrix
`n`	number of samples
`id`	vector of node identifiers
`lambda`	tuning parameter for penalty
`W`	optional penalty weight matrix (same dimensions as S)
`update`	how often to print updates
`max.gap`	maximum allowable primal-dual gap
`max.iter`	maximum number of iterations to complete (overriedes max.gap)
`min.t`	minimum step size (overrides max.gap, max.iter)

Value

list of precision, covariance, optimization status, lambda, and number components

maLasso1 *Multi-attribute subgraph estimation*

Description

This should only be called from within maLasso.

Usage

```
maLasso1(S, n, id, lambda, W, update = 100, max.gap = 0.5, max.iter = 100,
  min.t = .Machine$double.eps)
```

Arguments

S	sample covariance matrix
n	number of samples
id	vector of node identifiers
lambda	tuning parameter for penalty
W	penalty weight matrix (same dimensions as S)
update	how often to print updates
max.gap	maximum allowable primal-dual gap
max.iter	maximum number of iterations to complete (overriedes max.gap)
min.t	minimum step size (overrides max.gap, max.iter)

Value

list of precision, covariance, optimization status, lambda

mapggm *mapggm*

Description

mapggm

`runLasso` *Wrapper function for multi-attribute network estimation via lasso*

Description

Wrapper function for multi-attribute network estimation via lasso

Usage

```
runLasso(S, n, id, lambda, W = NULL, mode = 1, update = 100,
  max.gap = 0.5, max.iter = 100, min.t = .Machine$double.eps)
```

Arguments

`S`	sample covariance matrix
`n`	number of samples
`id`	vector assigning variables to nodes
`lambda`	penalty tuning parameter
`W`	optional weight matrix
`mode`	1 (multiattribute; default), 2 (unstructured together), 3 (separately)
`update`	how often to print updates in optimization
`max.gap`	maximum allowable primal/dual gap
`max.iter`	maximum number of iterations to optimize (overrides max.gap)
`min.t`	minimum step size (overrides max.gap, max.iter)

Value

list of precision, covariance, optimization status, lambda, and number components

`runLassoSelect` *Wrapper function for multi-attribute network estimation plus selection via lasso*

Description

Wrapper function for multi-attribute network estimation plus selection via lasso

Usage

```
runLassoSelect(S, n, id, lambda.range, W = NULL, mode = 1, update = 100,
  max.gap = 0.5, max.iter = 100, min.t = .Machine$double.eps,
  method = "EBIC", Theta.true = NULL, plot = NULL, gamma = 0.5)
```

Arguments

`S`	sample covariance matrix
`n`	number of samples
`id`	vector assigning variables to nodes
`lambda.range`	vector of lambdas to try
`W`	optional weight matrix
`mode`	1 (multiattribute; default), 2 (unstructured together), 3 (separately)
`update`	how often to print updates in optimization
`max.gap`	maximum allowable primal/dual gap
`max.iter`	maximum number of iterations to optimize (overrides max.gap)
`min.t`	minimum step size (overrides max.gap, max.iter
`method`	how to select the best model ('EBIC', 'BIC', 'AIC', or 'hamming')
`Theta.true`	true underlying graph (needed for 'hamming' method only)
`plot`	boolean, whether or not to make diagnostic plot
`gamma`	gamma parameter for EBIC method (default 0.5)

Value

list of precision, covariance, optimization status, lambda, and number components

`updateNodes` *Update covariance/precision matrix estimates*

Description

Update covariance/precision matrix estimates

Usage

```
updateNodes(t.step, id, S, Omega.tmp, Sigma.tmp, lambda, W)
```

Arguments

`t.step`	step size currently in use
`id`	vector of node identifiers
`S`	sample covariance
`Omega.tmp`	current estimate of precision matrix
`Sigma.tmp`	current estimate of covariance matrix
`lambda`	penalty parameter
`W`	weight matrix

Value

list of updated covariance and precision estimates

Index