Package 'ldstatsHD'

August 18, 2016

Type Package

Title Linear Dependence Statistics for High-Dimensional Data

Version 1.0.0

Date 2016-08-16

Author Adria Caballe <a.caballe@sms.ed.ac.uk>

Maintainer Adria Caballe <a.caballe@sms.ed.ac.uk>

Description Statistical methods related
to the estimation and testing of multiple correlation, partial correlation and regression coefficient matrices when data is high-dimensional.

License GPL-3

LazyLoad no

Repository CRAN

Depends R (>= 2.10.0), huge

Imports RBGL, cluster, camel, igraph, VGAM, evd, MASS, Matrix,
fExtremes, corpcor, robustbase

Suggests JGL

NeedsCompilation no

Date/Publication 2016-08-18 15:57:55

R topics documented:

`ldstatsHD-package` *Linear Dependence Statistics for High-Dimensional data*

Description

This package consists of functions with statistical methods related to the estimation and testing of multiple correlation, partial correlation and regression coefficient matrices when data is high-dimensional ($n < p$).

Joint estimation of two partial correlation matrices (see `wfgl`) and joint estimation of two regression coefficient matrices (see `wfrl`) are currently implemented in the package. These use a weighted-fused lasso penalized maximum likelihood estimator such that they encourage both sparsity and similarity between estimated matrices.

ldstatsHD also contains approaches to select the sparsity tuning parameter of graphical lasso estimators such that several risk functions based on characteristics of the estimated networks are available (see `lambdaSelection`). Among others, statistics that measure clustering structure or network connectivity can be used to find the desired networks.

It finally includes statistical methods that test global dependence characteristics: (i) a test for equality of two correlation matrices as well as a test for Identity correlation matrix (see `eqCorrMatTest`); (ii) a test for equality of two correlation matrix rows as well as a test to check if a variable is linearly independent of the rest of the variables in a dataset (see `eqCorTestByRows`).

A particularity of the implemented methods in **ldstatsHD** is that it permits cases where datasets are dependent (e.g. paired data).

ldstatsHD provides two partial correlation matrix simulators such that all methods can be tested using using simulated data: see `pcorSimulator` to generate a single partial correlation / dataset and `pcorSimulatorJoint` to generate a joint partial correlation matrix and two (dependent) datasets.

Details

Package:	ldstatsHD
Type:	Package
Version:	1.0.1
Date:	2016-07-08
License:	GPL-2
LazyLoad:	yes

Author(s)

Caballe, Adria <a.caballe@sms.ed.ac.uk>, Natalia Bochkina and Claus Mayer.

References

To come

`agnesCoef` *AGNES coefficient*

Description

computes the AGNES coefficient of a graph structure.

Usage

```
agnesCoef(A)
```

Arguments

`A`	`matrix` or `Matrix` object with adjacency matrix of a graph.

Details

The input of the AGNES hierarchical algorithm is the dissimilarity matrix of the graph structure `A` computed by `graphCorr`. Then the R function `agnes` is used to calculate the coefficient.

Value

AGNES coefficient magnitude for the given graph structure.

Author(s)

Caballe, Adria <a.caballe@sms.ed.ac.uk>, Natalia Bochkina and Claus Mayer.

References

Rousseeuw, P., A. Struyf, and M. Hubert (2013). cluster: Cluster Analysis Basics and Extensions.

See Also

lambdaSelection for lambda selection approaches.

Examples

```
EX1        <- pcorSimulator(nobs = 50, nclusters = 2, nnodesxcluster = c(40,30),
                            pattern = "powerLaw")
y           <- EX1$y
out3        <- huge(y, method = "mb", lambda = 0.4)
PATH        <- out3$path[[1]]
hm          <- agnesCoef(PATH)
```

---

agnesLambdaSelection *AGNES regularization parameter selection*

Description

agnesLambdaSelection is a function designed to select the regularization parameter in graphical models. It selects the most clustered conditional dependence graph structure where clusters are defined by the hierarchical algorithm agnes (See details).

Usage

```
agnesLambdaSelection(obj, way = "direct", nite = 10, subsvec = NULL,
                     eps = 0.05, until = NULL, minNodes = 30,
                     distF = c("correlation","shortPath"))
```

Arguments

obj	an object of class huge, camel.tiger or wfgl.
way	name that uniquely identifies "direct" (default), "rand.sampling" for random subsets algorithm and "int.sampling" for intelligent subsets algorithm.
nite	vector with the number of iterations used for each lambda (only if way = "rand.sampling" or way = "int.sampling").
subsvec	vector with the number of subsamples used for each lambda (only if way = "rand.sampling" or way = "int.sampling"). If NULL, argument minNodes determines the number of subsamples for all lambdas.

eps	acceptance tolerance for subsets of variables.
until	the last path used in obj. If NULL, all paths are used to select lambda.
minNodes	minimum number of nodes with connections to compute the AGNES coefficient (the coefficient is zero for paths with less nodes than minNodes).
distF	distance function used to find the dissimilarity matrix from the graph: name that uniquely identifies "correlation" and "shortPath".

Details

AGNES algorithm finds λ by minimizing the risk function

$$R_{AGNES}(\lambda) = -AC(\lambda)$$

where $AC(\lambda)$ is the AGNES coefficient calculated using the R function agnes. Using AGNES we select the λ that maximizes the between vs within cluster dissimilarities ratio given the dissimilarity matrix of the graph (see graphCorr and graphDist for possible dissimilarities).

A variable subset selection algorithm is available to estimate $AC(\lambda)$ for very high-dimensional data. It is recommended in order to save memory space and computational time. Especially way = "int.sampling" which tends to finds similar lambda selections to the default procedure.

agnesLambdaSelection gives a good recovery of global network characteristics when the true partial correlation matrix is block diagonal.

Value

An object of class lambdaSelection containing the following components:

opt.lambda	optimal lambda.
crit.coef	coefficients for each lambda given the criterion AGNES.
criterion	with value "AGNES".

Author(s)

Caballe, Adria <a.caballe@sms.ed.ac.uk>, Natalia Bochkina and Claus Mayer.

References

Caballe, A., N. Bochkina, and C. Mayer (2016). Selection of the Regularization Parameter in Graphical Models using network charactaristics. eprint arXiv:1509.05326, 1-25.

See Also

lambdaSelection for other lambda selection approaches and agnes for clustering implementation.

Examples

```
# example to use agnes function
EX1         <- pcorSimulator(nobs = 70, nclusters = 3, nnodesxcluster = c(40,30,20),
                             pattern = "powerLaw")
y           <- EX1$y
Lambda.SEQ  <- seq(.25, 0.70, length.out=40)
out3        <- huge(y, method = "mb", lambda = Lambda.SEQ)
AG.COEF     <- agnesLambdaSelection(out3, distF = "shortPath", way = "direct")
print(AG.COEF)
```

`aicAndbicLambdaSelection`

AIC/BIC regularization parameter selection

Description

`aicAndbicLambdaSelection` is a function designed to select the regularization parameter in graphical models. It selects the graph with smallest AIC or BIC coefficients.

Usage

```
aicAndbicLambdaSelection(obj,  y, criterion = c("AIC", "BIC"))
```

Arguments

`obj`	an object of class `huge` or `camel.tiger`.
`y`	original $n \times p$ data set.
`criterion`	coefficients and optimal lambdas to be stored: to select from `"AIC"`, `"BIC"` or both.

Value

An object of class `lambdaSelection` containing the following components:

`opt.lambda`	optimal lambdas for AIC and BIC.
`crit.coef`	coefficients for each lambda given the criterion AIC and BIC.
`criterion`	with value defined by argument `criterion`.

Author(s)

Caballe, Adria <a.caballe@sms.ed.ac.uk>, Natalia Bochkina and Claus Mayer.

References

Caballe, A., N. Bochkina, and C. Mayer (2016). Selection of the Regularization Parameter in Graphical Models using network charactaristics. eprint arXiv:1509.05326, 1-25.

See Also

lambdaSelection for other lambda selection approaches.

Examples

```
# example to use aicAndBic function
EX1         <- pcorSimulator(nobs = 50, nclusters = 3, nnodesxcluster = c(40,30,30),
                             pattern = "powerLaw")
y           <- EX1$y
Lambda.SEQ  <- seq(.35, 0.70, length.out = 40)
out3        <- huge(y, method = "glasso", lambda = Lambda.SEQ, cov.output = TRUE)
AIC.COEF    <- aicAndbicLambdaSelection(out3, y = y)
print(AIC.COEF)
```

---

amseLambdaSelection *Augmented-MSE regularization parameter selection*

Description

amseLambdaSelection is a function designed to select the regularization parameter lambda in graphical models that compromises global clustering structure and variability of the graph.

Usage

```
amseLambdaSelection(obj, pathIni, y, generator = c("subsampling", "montecarlo"),
                    pB = 0.7, nite = 10, method = "mb", from = 1, until = NULL,
                    distF = c("correlation","shortPath"), oneByone = FALSE,
                    many = 3)
```

Arguments

obj	an object of class huge or camel.tiger.
pathIni	path with best global characteristics (for instance the agnesLambdaSelection selected path).
y	original $n \times p$ data set.

generator	type of generator to find the mean squared error: name that uniquely identifies "subsampling" or "montecarlo".
pB	proportion of observations used in subsampling iterations.
nite	number of iterations used to estimate the mean square error.
method	method used to estimate the networks: name that uniquely identifies "mb", "glasso" or "tiger".
from	starting point in lambda sequence.
until	last point in lambda sequence. If until = NULL, all lambda sequence is explored.
distF	distance function used to find the dissimilarity matrix from the graph: name that uniquely identifies "correlation" or "shortPath".
oneByone	If TRUE, the estimation process is done separately for each λ.
many	If oneByone = TRUE, the estimation process is done separately for every many λ's.

Details

A-MSE algorithm finds λ by minimizing the risk function

$$R_{AMSE}(\lambda) = E(\sum_{i>j} |\delta_{ij} - \hat{\delta}_{ij}^{\lambda}|^2)$$

where $\hat{\delta}_{ij}^{\lambda}$ is the dissimilarity matrix of the graph (see graphCorr). The expected value is approximated by either subsampling or Monte Carlo and the theoretical δ_{ij} is approximated by the graph in pathIni.

We recommend using the AGNES algorithm agnesLambdaSelection to approximate pathIni since provides good reference of global network structure for clustered-based graph structures. Then, A-MSE gives a good trade-off between graph variability and global network characteristics.

If pathIni is given by agnesLambdaSelection and generator = "subsampling", then the lambda selected is always smaller than the lambda obtained by AGNES.

The oneByone approach is suggested to save memory space for very high-dimensional data.

Value

An object of class lambdaSelection containing the following components:

opt.lambda	optimal lambda.
crit.coef	coefficients for each lambda given the criterion A-MSE.
criterion	with value "A-MSE".

Author(s)

Caballe, Adria <a.caballe@sms.ed.ac.uk>, Natalia Bochkina and Claus Mayer.

References

Caballe, A., N. Bochkina, and C. Mayer (2016). Selection of the Regularization Parameter in Graphical Models using network charactaristics. eprint arXiv:1509.05326, 1-25.

See Also

lambdaSelection for other lambda selection approaches.

Examples

```
# example to use amse function
EX1         <- pcorSimulator(nobs = 70, nclusters = 3, nnodesxcluster = c(40,30,20),
                             pattern = "powerLaw")
y           <- EX1$y
Lambda.SEQ  <- seq(.25, 0.70, length.out = 40)
out3        <- huge(y, method = "mb", lambda = Lambda.SEQ)
AG.COEF     <- agnesLambdaSelection(out3, distF = "shortPath", way = "direct")
AG.LAMB     <- which(AG.COEF$opt.lambda == Lambda.SEQ)

## not run
#AAG.COEF    <- amseLambdaSelection(out3, out3$path[[AG.LAMB]], y = y,
#                 distF = "shortPath", from = AG.LAMB)
#print(AAG.COEF)
```

---

cor2mean *Average square correlation by rows*

Description

Finds in a computationally fast algorithm the average square correlation magnitude for every variable of a dataset.

Usage

```
cor2mean(mat)
```

Arguments

mat	$p \times n$ matrix with the p-variate dataset.

Details

It is especially suitable for high dimensions. For instance it handles well dimensions of order of thousands.

Value

The average square correlation magnitude of the sample correlation matrix (including the diagonal) for every variable in `mat`.

Author(s)

Mayer, Claus, Adria Caballe and Natalia Bochkina.

References

To come

See Also

`cor2mean.adj` for adjusted average square correlation magnitude.

Examples

```
EX1      <- pcorSimulator(nobs = 50, nclusters= 3, nnodesxcluster = c(100,30,50),
                          pattern = "powerLaw", plus = 0)
corsEX1  <- cor2mean(t(EX1$y))
```

---

cor2mean.adj *adjusted average square correlation by rows*

Description

Finds in a computationally fast algorithm the adjusted average square correlation magnitude for every variable of a dataset.

Usage

```
cor2mean.adj(mat)
```

Arguments

`mat`	$p \times n$ matrix with the p-variate dataset.

Details

The adjusted average square correlation of variable i is given by

$$(n-1)/(n-2)\bar{r}_i^2 - 1/(n-2)$$

where n is the sample size and $\bar{r}_i^2$ is the average square correlation matrix for the ith row, which is computed by `cor2mean`.

Value

A vector containing the adjusted square average correlation (excluding diagonal) for every variable.

Author(s)

Mayer, Claus, Adria Caballe and Natalia Bochkina.

References

To come

See Also

`cor2mean` for average square correlations.

Examples

```
EX1        <- pcorSimulator(nobs = 50, nclusters= 3, nnodesxcluster = c(100,30,50),
                            pattern = "powerLaw", plus = 0)
corsEX1     <- cor2mean(t(EX1$y))
corsadjEX1  <- cor2mean.adj(t(EX1$y))
```

---

eqCorrMatTest *equality of two correlation matrices test*

Description

tests the equality of two correlation matrices coming from two independent (or paired) datasets, that can possibly be high dimensional.

Usage

```
eqCorrMatTest(D1, D2 = NULL, testStatistic = c("AS", "max", "exc"),
              testNullDist = c("asyIndep","asyDep", "np"), nite= 500,
              paired = FALSE, threshold = 2.3, excAdj = TRUE, exact = FALSE,
              conf.level = 0.95, ...)
```

Arguments

`D1`	first population dataset in matrix $n_1 \times p$ form.
`D2`	second population dataset in matrix $n_2 \times p$ form. If `D2 = NULL`, an hypothesis testing for non-identity correlation matrix is performed instead.
`testStatistic`	test statistic used for the hypothesis testing: name that uniquely identifies `"AS"` for average of squares based test, `"max"` for an extreme values test and `"exc"` for an exceedances-based test. If `"all"`, all tests are performed.

testNullDist	Null distribution approximation. Either assuming independence between sample coefficients ("asyIndep"), accounting for dependence in a parametric distribution ("asyDep"), or accounting for dependence by a non-parametric distribution ("np").
nite	number of iterations used to generate the permuted samples (only if testNullDist = "asyDep" or testNullDist = "np"). It is also used to approximate a null distribution by Monte Carlo when testNullDist = "asyIndep" and testStatistic = "exc".
paired	if TRUE, observations in D1 and D2 are assumed to be matched (n_1 must be equal to n_2).
threshold	exceedance threshold used if testStatistic = "exc".
excAdj	weight for the exceedances test. If excAdj = FALSE the test statistic is given by the squared exceedances. In contrast, If excAdj = TRUE the test statistic is given by the squared of the exceedances minus the threshold.
exact	permuted samples method: if TRUE it forces to have the exact same number of observations in the two conditions in the samples exchanging process. If FALSE, permutations are made exchanging matched observations from the two datasets randomly with probability equal to 0.5.
conf.level	confidence level of the interval.
...	arguments passed to or from other methods to the low level.

Details

The extreme value test is the most powerful test against very sparse alternatives whereas sum of squares test is the most powerful when the true differential correlation matrix is dense. Otherwise, for a reasonable selection of the exceedance threshold, the exceedances test overperforms the power of the other two tests.

Paired structures can be used in this function. For instance, if paired = TRUE then the correlation between sample correlation coefficients in the two matrices is estimated to adjust the test statistic.

Asymptotic independence tests are fast since they do not compute permuted samples and can be used, even for paired data, under weakly dependent assumptions (very sparse correlation matrices) when sample sizes are large. If these assumptions do not hold, wrong representations of the p-values under H_0 could be found, in which case, permuted based null distributions should be used instead.

Testing if a correlation matrix is the identity matrix can also be performed when D2 = NULL. Note that the same type of test statistics and null distributions are available in this setting. Nevertheless, Monte Carlo simulations are used instead of permuted samples. Here null distributions are approximated by computing sample correlation matrices of generated data following a $N_p(0, I)$ under the assumption of normality.

Value

An object of class eqCorrMatTest containing the test statistics, p-values and confidence intervals for the selected tests.

Author(s)

Caballe, Adria <a.caballe@sms.ed.ac.uk>, Natalia Bochkina, Claus Mayer and Ioannis Papastathopoulos.

References

To come

See Also

eqCorTestByRows for testing linear independence and equality of sames rows in two correlation matrices.

Examples

```
#### data
EX2 <- pcorSimulatorJoint(nobs = 50, nclusters = 3, nnodesxcluster = c(40,40,40),
                          pattern = "pow", diffType = "cluster", dataDepend = "diag",
                          pdiff=0.5)

#### eq. corr. mat. test
## not run
#test1 <- eqCorrMatTest(EX2$D1, EX2$D2, testStatistic = c("AS", "exc", "max"),
#                       paired = TRUE, nite = 400)
#print(test1)

## not run
#test2 <- eqCorrMatTest(EX2$D1, NULL, testStatistic = c("AS", "exc", "max"),
#                       paired = TRUE, nite = 400)
#print(test2)
```

---

eqCorTestByRows *Correlation matrices test by rows*

Description

Tests whether the gth row of a correlation matrix is either non-zero or different to the same row of another correlation matrix. Allows for paired data.

Usage

```
eqCorTestByRows(D1, D2 = NULL, testStatistic = c("AS", "max"), nite = 200,
                paired = FALSE, exact = TRUE, subMatComp = FALSE, iniP = 1,
                finP = NULL, conf.level = 0.95)
```

Arguments

D1	first population dataset in matrix $n_1 \times p$ form.
D2	second population dataset in matrix $n_2 \times p$ form. If D2 = NULL non-zero correlation rows test is performed instead.

testStatistic	test statistic used for the hypothesis testing: name that uniquely identifies "AS" for average of squares based test and "max" for an extreme value test.
nite	number of iterations used to generate the permuted samples.
paired	if TRUE, observations in D1 and D2 are assumed to be matched (n_1 must be equal to n_2).
exact	permuted samples method: if TRUE it forces to have the exact same number of observations in the two conditions in the samples exchanging process. If FALSE, permutations are made exchanging matched observations from the two datasets randomly with probability equal to 0.5.
subMatComp	used to reduce computational time when using the test in very high dimensional data. If TRUE correlation sub-matrices are used, if FALSE, whole correlation matrices are computed.
iniP	only for subMatComp = TRUE and D2 != NULL. First row to be tested.
finP	only for subMatComp = TRUE and D2 != NULL. Last row to be tested.
conf.level	confidence level of the interval.

Details

This test uses a sum of squares based test statistic as given by the adjusted squared correlation cor2mean.adj as well as an extreme value based test statistic as given by max.

Null distributions are approximated differently when testing equality of two correlation rows and testing if correlation rows are equal to zero. In the first case, permuted samples are used to construct the confidence interval (see details in eqCorrMatTest). In the latter, they are found using Monte Carlo samples. For instance, n iid observations from a normal distribution $N(0, 1)$ are generated. Then, the adjusted square (or absolute maximum) correlations between these montecarlo samples and the original data $D1$ are found.

Value

An object of class eqCorTestByRows containing the following components:

AStest	average of squares test statistics.
pvalAS	average of squares test p-values.
ciAS	average of of squares test statistic confidence interval.
Maxtest	extreme value test statistics.
pvalMax	extreme value test p-values.
ciMax	extreme value test statistic confidence interval.

Author(s)

Caballe, Adria <a.caballe@sms.ed.ac.uk>, Natalia Bochkina and Claus Mayer.

References

to come.

See Also

plot.eqCorTestByRows for graphical representation.
eqCorrMatTest for testing equality of two correlation matrices.

Examples

```
#### data
EX2 <- pcorSimulatorJoint(nobs = 200, nclusters = 3, nnodesxcluster = c(60,40,50),
                          pattern = "pow", diffType = "cluster", dataDepend = "diag",
                          pdiff = 0.5)

#### eq corr by rows
## not run
#test1 <- eqCorTestByRows(EX2$D1, EX2$D2, testStatistic = c("AS", "max"),
#                          nite = 200, paired = TRUE, exact = TRUE, subMatComp = FALSE,
#                          iniP = 1, finP = 40, conf.level = 0.95)
#print(test1)

#### zero corr by rows
#test2 <- eqCorTestByRows(EX2$D1, testStatistic = c("AS", "max"), nite = 1000,
#                          conf.level = 0.95)
#print(test2)
```

---

estradaIndex *Estrada Index of a graph structure*

Description

Computes the Estrada Index given the adjacency matrix of a graph structure.

Usage

```
estradaIndex(A)
```

Arguments

A	matrix or Matrix object with adjacency matrix of a graph.

Details

The Estrada Index is calculated by

$$EE(\lambda) = \sum_{j=1}^{p} \exp(\gamma_j(\lambda)),$$

where $\gamma_1(\lambda), \ldots, \gamma_p(\lambda)$ are the eigenvalues of A_G^{λ}.

Value

Estrada index coefficient

Author(s)

Caballe, Adria <a.caballe@sms.ed.ac.uk>, Natalia Bochkina and Claus Mayer.

References

Estrada, E. (2011). The structure of complex networks. New York: OXFORD University press.

See Also

lambdaSelection for lambda selection approaches.

Examples

```
EX1         <- pcorSimulator(nobs = 50, nclusters=2, nnodesxcluster=c(40,30),
                             pattern="powerLaw")
y           <- EX1$y
out3        <- huge(y, method = "mb", lambda = 0.4)
PATH        <- out3$path[[1]]
hm          <- estradaIndex(PATH)
```

---

graphCorr *One minus graph correlation matrix*

Description

graphCorr computes the dissimilarity matrix (one minus the correlation matrix) of a graph structure.

Usage

```
graphCorr(A, nodesDegree = NULL)
```

Arguments

A	matrix or Matrix object with adjacency matrix of a graph.
nodesDegree	vector with nodes degree (in case it is been previously calculated).

Details

The similarity matrix of a graph is given by

$$\sigma_{ij} = \frac{\eta_{ij}}{\sqrt{k_i k_j}},$$

where η_{ij} is defined by the number of common neighbors of nodes i and j and k_i is the degree of the node i. The dissimilarity matrix is given by

$$\delta_{ij} = 1 - \sigma_{ij}.$$

Value

a dissimilarity lower triangular matrix with one minus the correlation of the graph nodes.

Author(s)

Caballe, Adria <a.caballe@sms.ed.ac.uk>, Natalia Bochkina and Claus Mayer.

References

Costa, L. and F. Rodrigues (2007). Characterization of complex networks: A survey of measurements. Advances in Physics 56(1), 167-242.

See Also

graphDist for inverse of the geodesic distance matrix.

Examples

```
# example to use of graphCorr function
EX1         <- pcorSimulator(nobs = 50, nclusters = 2, nnodesxcluster = c(40,30),
                             pattern = "powerLaw")
y           <- EX1$y
out3        <- huge(y, method = "mb", lambda = 0.4)
gc          <- graphCorr(out3$path[[1]])
```

---

graphDist *Inverse of the geodesic distance matrix*

Description

graphDist computes the dissimilarity matrix (inverse of geodesic distance) of a graph structure.

Usage

```
graphDist(A)
```

Arguments

A	matrix or Matrix object with adjacency matrix of a graph.

Details

The geodesic distance between two nodes i and j of a graph is given by the shortest number of edges so that we can go from one of the nodes to the other.

Value

a dissimilarity lower triangular matrix with the inverse of the geodesic distance matrix.

Author(s)

Caballe, Adria <a.caballe@sms.ed.ac.uk>, Natalia Bochkina and Claus Mayer.

References

Costa, L. and F. Rodrigues (2007). Characterization of complex networks: A survey of measurements. Advances in Physics 56(1), 167-242.

See Also

graphCorr for graph correlation matrix.

Examples

```
# example to use of graphDist function
EX1         <- pcorSimulator(nobs = 50, nclusters = 2, nnodesxcluster = c(40,30),
                             pattern = "powerLaw")
y           <- EX1$y
out3        <- huge(y, method = "mb", lambda = 0.4)
gd          <- graphDist(out3$path[[1]])
```

harmMeanDist *Harmonic mean of network distances*

Description

Finds the harmonic mean of the geodesic distances between nodes in a graph.

Usage

```
harmMeanDist(A, nodesDegree = NULL)
```

Arguments

A	matrix or Matrix object with adjacency matrix of a graph.
nodesDegree	vector with nodes degree (in case it is been previously calculated).

Details

The geodesic distance d_{ij} between two nodes i and j of a graph is given by the shortest number of edges so that we can go from one of the nodes to the other. Then, the harmonic mean of these distances is given by

$$\bar{d} = [p(p-1)/(2\sum_{i<j} d_{ij})]^{-1}.$$

Value

The harmonic mean value.

Author(s)

Caballe, Adria <a.caballe@sms.ed.ac.uk>, Natalia Bochkina and Claus Mayer.

References

Costa, L. and F. Rodrigues (2007). Characterization of complex networks: A survey of measurements. Advances in Physics 56(1), 167-242.

See Also

lambdaSelection for lambda selection approaches.

Examples

```
# example to use of harmonic mean function
EX1         <- pcorSimulator(nobs = 50, nclusters = 2, nnodesxcluster = c(40,30),
                             pattern = "powerLaw")
y           <- EX1$y
out3        <- huge(y, method = "mb", lambda = 0.4)
PATH        <- out3$path[[1]]
hm          <- harmMeanDist(PATH)
```

`lambdaSelection` *Regularization parameter selection based on network characteristics*

Description

`lambdaSelection` is a function designed to select the sparsity regularization parameter in graphical models.

There are available seven different criterion to select lambda with risk functions based on network characteristics: Path connectivity (PC), AGlommerative NESted (AGNES), Augmented-MSE (A-MSE), Vulnerability (VUL), AIC/BIC and StARS (from huge package).

Usage

```
lambdaSelection(obj, criterion = c("PC","AGNES","A-MSE","VUL","STARS", "AIC", "BIC"),
                ...)
```

Arguments

`obj`	an object of class huge, camel.tiger or wfgl.
`criterion`	regularization parameter selection approach: name that uniquely identifies "PC" (Path connectivity), "AGNES" (AGlommerative NESted Algorithm), "A-MSE" (Augmented mean square error), "VUL" (Maximum VULnerability), "AIC"/"BIC" (minimum AIC/BIC) or "StARS" (Stability approach).
`...`	arguments passed to or from other methods to the low level. See pcLambdaSelection, agnesLambdaSelection, amseLambdaSelection, vulLambdaSelection, aicAndbicLambdaSelection and huge for details.

Details

This function considers seven ways of selecting the regularization parameter in graphical models by minimizing a certain risk function based only on network characteristics of the underlying structure of Ω

$$\hat{\lambda} = \arg\min_{\lambda} R(\lambda, \hat{G}_{\lambda}),$$

where $\hat{G}_{\lambda}$ is the estimated graph structure of $\hat{\Omega}$. For instance see pcLambdaSelection, agnesLambdaSelection, amseLambdaSelection, vulLambdaSelection, aicAndbicLambdaSelection and huge for the implemented criterions to select λ.

For wfgl objects, only `criterion = c("PC","AGNES","VUL")` are available.

Value

An object of class `lambdaSelection` containing the following components:

`opt.lambda`	optimal lambda.
`crit.coef`	coefficients for each lambda given the criterion.
`criterion`	criterion used to select lambda.

Note

For large dimensions, `"A-MSE"`, `"VUL"` (the most) and `"StARS"` can be computationally intensive.

Author(s)

Caballe, Adria <a.caballe@sms.ed.ac.uk>, Natalia Bochkina and Claus Mayer.

References

Caballe, A., N. Bochkina, and C. Mayer (2016). Selection of the Regularization Parameter in Graphical Models using network charactaristics. eprint arXiv:1509.05326, 1-25.

See Also

pcLambdaSelection, agnesLambdaSelection, amseLambdaSelection vulLambdaSelection, aicAndbicLambdaSelection and huge.

Examples

```
# example to use agnes function
EX1         <- pcorSimulator(nobs = 50, nclusters = 3, nnodesxcluster = c(40,30,30),
                             pattern="powerLaw")
y           <- EX1$y
Lambda.SEQ  <- seq(.35,0.70, length.out = 40)
out3        <- huge(y, method = "mb", lambda = Lambda.SEQ)
PC.COEF     <- lambdaSelection(out3, criterion = "PC")
#AG.COEF     <- lambdaSelection(out3, criterion = "AGNES")
```

`pcLambdaSelection` *Path Connectivity regularization parameter selection*

Description

`pcLambdaSelection` is a function designed to select the regularization parameter in graphical models. It selects the graph which captures the biggest drop in graph connectivity.

Usage

```
pcLambdaSelection(obj)
```

Arguments

`obj`	an object of class `huge`, `camel.tiger` or `wfgl`.

Details

Path Connectivity (PC) algorithm finds λ by maximizing the biggest drop of connectivity in estimated graphs. We define connectivity by the average geodesic distance between pairs of nodes (see `graphDist`).

PC gives a fast and suitable way to select λ when there are distinct clusters in the data. Given two graphs, corresponding to two consecutive λ's, the difference between the average geodesic distance will be large if the first graph contains edges that connect different clusters which are not present in the second graph.

Note that PC should be used when fitting graphical models with an equidistant sequence for λ.

Value

An object of class `lambdaSelection` containing the following components:

`opt.lambda`	optimal lambda.
`crit.coef`	coefficients for each lambda given the criterion PC.
`criterion`	with value `"PC"`.

Author(s)

Caballe, Adria <a.caballe@sms.ed.ac.uk>, Natalia Bochkina and Claus Mayer.

References

Caballe, A., N. Bochkina, and C. Mayer (2016). Selection of the Regularization Parameter in Graphical Models using network charactaristics. eprint arXiv:1509.05326, 1-25.

See Also

lambdaSelection for other lambda selection approaches.

Examples

```
# example to use pc function
EX1         <- pcorSimulator(nobs = 70, nclusters = 3, nnodesxcluster = c(40,30,20),
                             pattern = "powerLaw")
y           <- EX1$y
Lambda.SEQ  <- seq(.25,0.70,length.out = 40)
out3        <- huge(y, method = "mb", lambda = Lambda.SEQ)
PC.COEF     <- pcLambdaSelection(out3)
print(PC.COEF)
```

---

pcorSimulator *Partial Correlation Matrix simulator*

Description

pcorSimulator creates a block diagonal positive definite precision matrix with three possible graph structures: hubs-based, power-law and random. Then, it generates samples from a multivariate normal distribution with covariance matrix given by the inverse of such precision matrix.

Usage

```
pcorSimulator(nobs, nclusters, nnodesxcluster, pattern = "powerLaw",
              low.strength = 0.5, sup.strength = 0.9, nhubs = 5,
              degree.hubs = 20, nOtherEdges = 30, alpha = 2.3, plus = 0,
              prob = 0.05, perturb.clust = 0, mu = 0,
              probSign = 0.5, seed = sample(10000, nclusters))
```

Arguments

nobs	number of observations.
nclusters	number of clusters or blocks of variables.
nnodesxcluster	number of nodes/variables per cluster.
pattern	graph structure pattern: name that uniquely identifies "hubs", "powerLaw" and "random".
low.strength	minimum magnitude for nonzero partial correlation elements before regularization.
sup.strength	maximum magnitude for nonzero partial correlation elements before regularization.

nhubs	number of hubs per cluster (if pattern = "hubs").
degree.hubs	degree of hubs (if pattern = "hubs").
nOtherEdges	number of edges for non-hub nodes (if pattern = "hubs").
alpha	positive coefficient for the Riemman function in power-law distributions.
plus	power-law distribution added complexity (zero by default).
prob	probability of edge presence for random networks (if pattern = "random").
perturb.clust	proportion of the total number of edges that are connecting two different clusters.
mu	expected values vector to generate data (zero by default).
probSign	probability of positive sign for non-zero partial correlation coefficients. Thus, negative signs are obtained with probability 1-probSign.
seed	vector with seeds for each cluster.

Details

Hubs-based networks are graphs where only few nodes have a much higher degree (or connectivity) than the rest. Power-law networks assume that the variable p_k, which denotes the fraction of nodes in the network that has degree k, is given by a power-law distribution

$$p_k = \frac{k^{-\alpha}}{\varsigma(\alpha)},$$

for $k \geq 1$, a constant $\alpha > 0$ and the normalizing function $\varsigma(\alpha)$ which is the Riemann zeta function. Finally, random networks are also defined by the distribution in the proportion p_k. In this case, p_k follows a binomial distribution

$$p_k = \binom{p}{k} \theta^k (1-\theta)^{p-k},$$

where the parameter θ determines the proportion of edges (or sparsity) in the graph.

The regularization is given by $\Omega^{(1)} = \Omega^{(0)} + \delta I$, with δ such that the condition number of $\Omega^{(1)}$ is less than the number of nodes.

Value

An object of class pcorSim containing the following components:

y	generated data set.
hubs	hub nodes position.
edgesInGraph	edges given by the non-zero elements in the precision matrix.
omega	precision matrix used to generate the data.
covMat	covariance matrix used to generate the data.
path	adjacency matrix corresponding to the non-zero structure of omega.

Author(s)

Caballe, Adria <a.caballe@sms.ed.ac.uk>, Natalia Bochkina and Claus Mayer.

References

Cai, T., W. Liu, and X. Luo (2011). A Constrained L1 Minimization Approach to Sparse Precision Matrix Estimation. Journal of the American Statistical Association 106(494), 594-607.

Newman, M. (2003). The structure and function of complex networks. SIAM REVIEW 45, 167-256.

Caballe, A., N. Bochkina, and C. Mayer (2016). Selection of the Regularization Parameter in Graphical Models using network charactaristics. eprint arXiv:1509.05326, 1-25.

See Also

`plot.pcorSim` for graphical representation of the generated partial correlation matrix.
`pcorSimulatorJoint` for joint partial correlation matrix generation.

Examples

```
# example to use pcorSimulator function

EX1 <- pcorSimulator(nobs = 50, nclusters=3, nnodesxcluster=c(100,30,50),
                pattern="powerLaw", plus=0)
print(EX1)

EX2 <- pcorSimulator(nobs = 25, nclusters=2, nnodesxcluster=c(60,40),
                pattern="powerLaw", plus=1)
print(EX2)
```

---

pcorSimulatorJoint *joint partial correlation matrices simulator*

Description

`pcorSimulatorJointPaired` creates two similar positive definite precision matrices with three possible graph structures: hubs-based, power-law and random. Moreover, it allows for three types of differential graph structures: random differences, clustered differences or a mixture of the two. Then, it generates (dependent) datasets from a multivariate normal distribution defined by the inverse of such precision matrices.

Usage

```
pcorSimulatorJoint(nobs, nclusters, nnodesxcluster, pattern = "hubs",
                   diffType = "cluster", dataDepend = "ind", low.strength = 0.5,
                    sup.strength = 0.9, pdiff = 0, nhubs = 5, degree.hubs = 20,
                    nOtherEdges = 30, alpha = 2.3, plus = 0, prob = 0.05,
                    perturb.clust = 0, mu = 0, diagCCtype = "dicot",
                    diagNZ.strength = .5, mixProb = 0.5, probSign = 0.5,
                    exactZeroTh = 0.05, seed = sample(10000,nclusters+2))
```

Arguments

nobs	number of observations.
nclusters	number of clusters or blocks of variables.
nnodesxcluster	number of nodes/variables per cluster.
pattern	graph structure pattern: name that uniquely identifies "hubs", "power" and "random".
diffType	pattern in differential edges: name that uniquely identifies "random", "cluster" or "mixed".
dataDepend	model used to describe the dependent structure for the data: name that uniquely identifies "ind" (no dependence), "diagOmega", "mult" or "add".
low.strength	minimum magnitude for nonzero partial correlation elements before regularization.
sup.strength	maximum magnitude for nonzero partial correlation elements before regularization.
pdiff	proportion of differential edges from the total number edges in each graph.
nhubs	number of hubs per cluster (if pattern = "hubs").
degree.hubs	degree of hubs (if pattern = "hubs").
nOtherEdges	number of edges for non-hub nodes (if pattern = "hubs").
alpha	positive coefficient for the Riemman function in power-law distributions.
plus	power-law distribution added complexity (zero by default).
prob	probability of edge existence for random networks (if pattern="random").
perturb.clust	proportion of the total number of edges that are connecting two different clusters.
mu	expected values vector to generate data (zero by default).
diagCCtype	way to generate diagonal values of either cross partial correlation matrix (if dataDepend = "diagOmega") or cross correlation matrix (if dataDepend = "mult" or dataDepend = "add"): name that uniquely identifies "dicot" or "beta13" (see details).
diagNZ.strength	magnitude for the non-zero elements in the diagonal of the cross (partial) correlation when diagCCtype = "dicot".
mixProb	proportion of random differential connections if diffType = "mixed". The remaining connections are given by a cluster type.
probSign	probability of positive sign for non-zero partial correlation coefficients. Thus, negative signs are obtained with probability 1-probSign.
exactZeroTh	partial correlation coefficients smaller than exactZeroTh are considered exact zeros.
seed	vector with seeds for each cluster.

Details

First, pcorSimulator is used to create a common precision matrix among the two populations. Then, differential edges are added based on the next two patterns: Cluster - a graph cluster is zero in one condition and non-zero in the other condition; Random - differential connections are given randomly in the graph.

Paired structure is defined by arguments dataDepend and diagCCtype. Additive (dataDepend = "add") and multiplicative (dataDepend = "mult") models are used on the cross-covariance matrix such that $\Sigma_{XY} = \Delta\Sigma_X\Delta^t$, with diagonal matrix Δ, $0 \leq \Delta_{ii} < 1$ and $\Sigma_{XY} = \Delta\Sigma_X^{1/2}\Sigma_Y^{1/2}\Delta^t$ respectively where diagonal coefficients in Δ are defined by diagCCtype. A simplification is also considered by assuming that variables in one data set are only conditionally dependent to the same variables of the other data set, hence assuming a diagonal structure in the cross joint partial correlation matrix that can also be defined by Δ. For the three models, In case diagCCtype = "dicot" the diagonal elements in Δ have zero/non-zero structure (with non-zero coefficients given in the parameter Δ). In case diagCCtype = "beta13" the diagonal elements are generated by a beta distribution with shape parameter equal to 1 and scale parameter equal to 3.

Value

An object of class pcorSimJoint containing the following components:

D1	dataset for first population.
D2	dataset for second population.
omega1	precision matrix for first population.
omega2	precision matrix for second population.
P	total number of variables.
diffs	differential edges.
delta	generated values for the dependent structure.
covJ	joint covariance matrix used to generate the data.
path1	adjacency matrix corresponding to the non-zero structure of omega1.
path2	adjacency matrix corresponding to the non-zero structure of omega2.

Author(s)

Caballe, Adria <a.caballe@sms.ed.ac.uk>, Natalia Bochkina and Claus Mayer.

References

Cai, T., W. Liu, and X. Luo (2011). A Constrained L1 Minimization Approach to Sparse Precision Matrix Estimation. Journal of the American Statistical Association 106(494), 594-607.

Newman, M. (2003). The structure and function of complex networks. SIAM REVIEW 45, 167-256.

Wit, E. and A. Abbruzzo (2015, feb). Factorial graphical models for dynamic networks. Network Science 3(01), 37-57.

Caballe, A., N. Bochkina, and C. Mayer (2016). Selection of the Regularization Parameter in Graphical Models using network charactaristics. eprint arXiv:1509.05326, 1-25.

See Also

pcorSimulator for precision matrix generator.
plot.pcorSimJoint for plotting joint partial correlation matrices.

Examples

```
# example to use pcorSimulatorJoint function
EX1 <- pcorSimulatorJoint(nobs = 50, nclusters = 2, nnodesxcluster = c(30, 40),
                          pattern = "pow", diffType = "cluster", dataDepend = "ind",
                          pdiff = 0.2, diagCCtype = "dicot", diagNZ.strength = .5)
print(EX1)

EX2 <- pcorSimulatorJoint(nobs = 50, nclusters = 2, nnodesxcluster = c(30, 40),
                          pattern = "pow", diffType = "rand", dataDepend = "diag",
                          pdiff = 0.05, diagCCtype = "beta")
print(EX2)
```

---

plot.eqCorTestByRows *plot for equality of two correlation matrices by rows test*

Description

graphical representation for the equality of two correlation matrices test by rows: confidence intervals of the test statistics.

Usage

```
## S3 method for class 'eqCorTestByRows'
plot(x, mains = c("AS CI", "max CI"), xlabs = c("",""), ylabs = c("",""),
      pch = "-", ownCols = TRUE, ...)
```

Arguments

x	object of class eqCorTestByRows.
mains	vector of size two with main of plots (for average of squares test and extreme value test).
xlabs	vector of size two with xlabs of plots (for average of squares test and extreme value test).
ylabs	vector of size two with ylabs of plots (for average of squares test and extreme value test).
pch	pch given to identify confidence interval limits.
ownCols	if ownCols = TRUE green and black colors are generated with green lines identifying significant variables.
...	arguments passed to or from other methods to the low level.

Author(s)

Caballe, Adria <a.caballe@sms.ed.ac.uk>, Natalia Bochkina and Claus Mayer.

See Also

eqCorTestByRows for equality of two correlation matrices by rows test.

Examples

```
EX2 <- pcorSimulatorJoint(nobs = 200, nclusters = 3, nnodesxcluster = c(60,40,50),
                          pattern = "pow", diffType = "cluster", dataDepend = "diag",
                          pdiff=0.5)

#### eq corr by rows
## not run
#eqCorEX2 <- eqCorTestByRows(EX2$D1, EX2$D2, testStatistic = c("AS", "max"),
#                            nite = 200, paired = TRUE, exact = TRUE,
#                            subMatComp = FALSE, iniP = 1, finP = 40,
#                            conf.level = 0.95)
#plot(eqCorEX2)
```

---

plot.pcorSim *Partial correlation matrix simulator plot*

Description

graphical representation of the non-zero partial correlation matrix structure

Usage

```
## S3 method for class 'pcorSim'
plot(x, vertex.size = 3, vertex.label = NA, hubsCol = TRUE, ...)
```

Arguments

x	object of class pcorSim.
vertex.size	plot.igraph parameter: vertex sizes.
vertex.label	plot.igraph parameter: vertex label.
hubsCol	if TRUE hub nodes are highlighted in a different color.
...	arguments passed to or from other methods to the low level.

Author(s)

Caballe, Adria <a.caballe@sms.ed.ac.uk>, Natalia Bochkina and Claus Mayer.

See Also

pcorSimulator for partial correlation matrix generation.
pcorSimulatorJoint for joint partial correlation matrix generation.

Examples

```
EX1 <- pcorSimulator(nobs = 50, nclusters = 3, nnodesxcluster = c(100,30,50),
                     pattern="powerLaw", plus = 0)
plot(EX1)

EX2 <- pcorSimulator(nobs = 25, nclusters = 2, nnodesxcluster = c(60,40),
                     pattern = "powerLaw", plus = 1)
plot(EX2)
```

---

plot.pcorSimJoint *Joint partial correlation matrix simulator plot*

Description

graphical representation of two non-zero partial correlation structures

Usage

```
## S3 method for class 'pcorSimJoint'
plot(x, minn = 0, col = c("blue","red","green"), vertex.size=3,
      edgesThickness = FALSE, ...)
```

Arguments

x	object of class pcorSimJoint.
minn	used for visualization purposes in very dense networks. It only plots nodes that have degree larger than minn.
col	vector defining edge colors: common edges (first element), only non-zero coefficients for first population (second element) and only non-zero coefficients for second population (third element).
vertex.size	plot.igraph parameter: vertex sizes.
edgesThickness	if TRUE, an edge thickness is proportional to the magnitude of its underlying estimated partial correlation coefficient.
...	arguments passed to or from other methods to the low level.

Author(s)

Caballe, Adria <a.caballe@sms.ed.ac.uk>, Natalia Bochkina and Claus Mayer.

See Also

pcorSimulatorJoint for joint partial correlation matrix generation.

Examples

```
EX1 <- pcorSimulatorJoint(nobs = 50, nclusters = 2, nnodesxcluster = c(30, 40),
                          pattern = "pow", diffType = "cluster", dataDepend = "ind",
                          pdiff = 0.2, diagCCtype = "dicot", diagNZ.strength = .5)
plot(EX1, edgesThickness = TRUE)

EX2 <- pcorSimulatorJoint(nobs = 50, nclusters = 3, nnodesxcluster = c(30, 40,60),
                           pattern = "pow", diffType = "cluster", dataDepend = "diag",
                           pdiff = 0.4, diagCCtype = "beta")
plot(EX2)
```

---

plot.wfgl *Joint partial correlation matrix estimator plot*

Description

graphical representation of the non-zero joint partial correlation structure.

Usage

```
## S3 method for class 'wfgl'
plot(x, minn = 0, col = c("blue","red","green"), vertex.size = 3,
      edgesThickness = FALSE, zoomThick = 10, ...)
```

Arguments

x	object of class wfgl.
minn	used for visualization purposes in very dense networks. It only plots nodes that have degree larger than minn.
col	vector defining estimated edge colors: common edges (first element), only non-zero coefficients for first population (second element) and only non-zero coefficients for second population (third element).
vertex.size	plot.igraph parameter: vertex sizes.
edgesThickness	if TRUE, an edge thickness is proportional to the magnitude of its underlying estimated partial correlation coefficient.

zoomThick	it increases the thickness of all edges by zoomThick times (used for visualization purposes).
...	arguments passed to or from other methods to the low level.

Author(s)

Caballe, Adria <a.caballe@sms.ed.ac.uk>, Natalia Bochkina and Claus Mayer.

See Also

wfgl for joint estimation of multiple partial correlation matrices.

Examples

```
EX2   <- pcorSimulatorJoint(nobs =50, nclusters = 3, nnodesxcluster = c(30, 30,30),
                            pattern = "pow", diffType = "cluster", dataDepend = "diag",
                            diagCCtype = "dicot", diagNZ.strength = 0.6)
## not run
#wfgl1 <- wfgl(EX2$D1, EX2$D2, lambda1 = 0.01, lambda2 = 0.1, paired = TRUE,
#              automLambdas = TRUE, sigmaEstimate = "CRmad",
#              pairedEst = "Reg-based-sim", maxiter = 30)
#plot(wfgl1)
#plot(wfgl1, minn = 1, edgesThickness = TRUE)
```

---

plot.wfrl *Joint regression coefficient matrix estimator plot*

Description

graphical representation of the non-zero joint regression coefficients structure

Usage

```
## S3 method for class 'wfrl'
plot(x, minn = 0, col = c("blue","red","green"), vertex.size = 2,
      vertex.color = c("red", "blue"), edgesThickness = FALSE,
      zoomThick = 10, ...)
```

Arguments

x	object of class wfrl.
minn	used visualization purposes in very dense networks. It only plots nodes that have degree larger than minn.

col	vector defining estimated edge colors: common edges (first element), only non-zero coefficients for first population (second element) and only non-zero coefficients for second population (third element).
vertex.size	plot.igraph parameter: vertex sizes.
vertex.color	vector defining the vertex colors for directed graph: first element describes the color of explanatory variables and second element describes the color for response variables.
edgesThickness	if TRUE, an edge thickness is proportional to the magnitude of its underlying estimated partial correlation coefficient.
zoomThick	it increases the thickness of all edges by zoomThick times (used for visualization purposes).
...	arguments passed to or from other methods to the low level.

Details

It produces a directed graph structure that connects explanatory variables to response variables.

Author(s)

Caballe, Adria <a.caballe@sms.ed.ac.uk>, Natalia Bochkina and Claus Mayer.

See Also

wfrl for joint estimation of regression coefficients.

Examples

```
N <- 200
EX2 <- pcorSimulatorJoint(nobs = N, nclusters = 3, nnodesxcluster = c(60,40,50),
                          pattern = "pow", diffType = "cluster", dataDepend = "diag",
                         low.strength = 0.5, sup.strength = 0.9, pdiff = 0.5, nhubs = 5,
                          degree.hubs = 20,  nOtherEdges = 30, alpha = 2.3, plus = 0,
                         prob = 0.05, perturb.clust = 0.2, mu = 0, diagCCtype = "dicot",
                          diagNZ.strength = 0.6, mixProb = 0.5, probSign = 0.7,
                          exactZeroTh = 0.05)

P           <- EX2$P
q           <- 50
BETA1       <- array(0,dim=c(P,q))
diag(BETA1) <- rep(0.35,q)
BETA2       <- BETA1
diag(BETA2)[c(1:floor(q/2))]<-0
sigma2  <- 1.3
Q       <- scale(EX2$D1)
W       <- scale(EX2$D2)
X       <- Q%*%BETA1 + mvrnorm(N,rep(0,q),diag(rep(sigma2,q)))
Y       <- W%*%BETA2 + mvrnorm(N,rep(0,q),diag(rep(sigma2,q)))
D1      <- list(scale(X),scale(Y))
D2      <- list(scale(Q),scale(W))
```

```
## not run
#wfrl1  <- wfrl(D1, D2, lambda1 = 0.01, lambda2 = 0.05, automLambdas = TRUE, paired = FALSE,
#               sigmaEstimate = "CRmad", maxiter=30, tol=1e-05)
#plot(wfrl1)
#plot(wfrl1, minn = 1, edgesThickness = TRUE)
```

`vulLambdaSelection` *Vulnerability regularization parameter selection*

Description

`vulLambdaSelection` is a function designed to select the regularization parameter in graphical models. It selects the graph with largest average nodes vulnerability.

Usage

```
vulLambdaSelection(obj, loo = FALSE, subOut =  10, nite = 50)
```

Arguments

`obj`	an object of class `huge`, `camel.tiger` or `wfgl`.
`loo`	if `TRUE` an exhaustive leave-one-out procedure is done, otherwise it is used a subsampling approach with `nite` iterations and leaving out `subOut` variables.
`subOut`	number of variables left out in each iteration (only used if `loo = FALSE`).
`nite`	number of iterations (only used if `loo = FALSE`).

Details

Vulnerability algorithm finds lambda by minimizing the risk function

$$R_{VUL}(\lambda) = -\sum_{i=1}^{p} \frac{E^{\lambda} - E_i^{\lambda}}{E^{\lambda}}$$

where E^{λ} is the global efficiency of the original network and E_i^{λ} is the global efficiency of the network once eliminating the node i. Global efficiency is defined by the harmonic mean of the geodesic distance (see `graphDist`).

Vulnerability gives λ selection that contains the most vulnerable graph, meaning that the removal of a node in the network in average would affect the most the estimated graph.

Value

An object of class `lambdaSelection` containing the following components:

`opt.lambda`	optimal lambda.
`crit.coef`	coefficients for each lambda given the criterion VUL.
`criterion`	with value `"VUL"`.

Author(s)

Caballe, Adria <a.caballe@sms.ed.ac.uk>, Natalia Bochkina and Claus Mayer.

References

Costa, L. and F. Rodrigues (2007). Characterization of complex networks: A survey of measurements. Advances in Physics 56(1), 167-242.

See Also

lambdaSelection for other lambda selection approaches.

Examples

```
# example to use vul function
EX1         <- pcorSimulator(nobs = 50, nclusters = 2, nnodesxcluster = c(40,30),
                             pattern="powerLaw")
y           <- EX1$y
Lambda.SEQ  <- seq(.35, 0.70, length.out = 10)
out3        <- huge(y, method = "mb", lambda = Lambda.SEQ)
## not run
#VUL.COEF    <- vulLambdaSelection(out3)
#print(VUL.COEF)
```

---

wfgl *weighted fused graphical lasso*

Description

wfgl estimates joint partial correlation matrices from two multivariate normal distributed datasets using an ADMM based algorithm. Allows for paired data.

Usage

```
wfgl(D1, D2, lambda1, lambda2, paired = TRUE, automLambdas = TRUE,
     sigmaEstimate = "CRmad", pairedEst = "Reg-based-sim", maxiter = 30,
     tol = 1e-05, nsubset = 10000, weights = c(1,1), rho=1, rho.increment = 1,
     triangleCorrection = TRUE, alphaTri = 0.01, temporalFolders = FALSE,
     notOnlyLambda2 = TRUE)
```

Arguments

D1	first population dataset in matrix $n_1 \times p$ form.
D2	second population dataset in matrix $n_2 \times p$ form.
lambda1	tuning parameter for sparsity in the precision matrices (sequence of lambda1 is allowed).
lambda2	tuning parameter for similarity between the precision matrices in the two populations (only one value allowed at a time).
paired	if TRUE, observations in D1 and D2 are assumed to be matched (n_1 must be equal to n_2).
automLambdas	if TRUE the lambda's are estimated automatically with lambda1 and lambda2 being expected false positive rate levels.
sigmaEstimate	robust method used to estimate the variance of estimated partial correlations: name that uniquely identifies "mad", "IQR" or "CRmad" (default). This measure is used to automatically select the tuning parameter (when automLambdas = TRUE).
pairedEst	type of estimator for the correlation of estimated partial correlation coefficients when "paired = TRUE": to select from Reg-based and Reg-based-sim (default). This measure is used to weight similarity penalization lambda2 for different pairs of variables.
maxiter	maximum number of iterations for the ADMM algorithm.
tol	convergence tolerance
nsubset	maximum number of estimated partial correlation coefficients (chosen randomly) used to select lambda1 and lambda2 automatically (when automLambdas = TRUE).
weights	weights for the two populations to find the inverse covariance matrices.
rho	regularization parameter used to compute matrix inverse by eigen value decomposition (default of 1).
rho.increment	default of 1.
triangleCorrection	if TRUE the estimated triangle graph structures are tested.
alphaTri	significance level for the tested triangle graph structures.
temporalFolders	if TRUE temporal files are created and eliminated within the procedure. It is used to free R memory space when the dimension is very large (order of thousands).
notOnlyLambda2	if FALSE only lambda2 is found automatically.

Details

wfgl uses a weighted-fused graphical lasso maximum likelihood estimator by solving:

$$\hat{\Omega}^{\lambda}_{WFGL} = \arg\max_{\Omega_X,\Omega_Y} [\sum_{k=X,Y} \log\det\Omega_k - tr(\Omega_k S_k) - P_{\lambda_1,\lambda_2,V}(\Omega_X,\Omega_Y)],$$

with

$$P_{\lambda_1,\lambda_2,V}(\Omega_X,\Omega_Y) = \lambda_1||\Omega_X||_1 + \lambda_1||\Omega_Y||_1 + \lambda_2 \sum_{i,j} v_{ij}|\Omega_{Y_{ij}} - \Omega_{X_{ij}}|,$$

where λ_1 is the sparsity tuning parameter, λ_2 is the similarity tuning parameter, and $V = [v_{ij}]$ is a $p \times p$ matrix to weight λ_2 for each coefficient of the differential precision matrix. If datasets are independent (`paired = "FALSE"`), then it is assumed that $v_{ij} = 1$ for all pairs (i, j). Otherwise (`paired = "TRUE"`), weights are estimated in order to account for the dependence structure between datasets in the differential network estimation.

Lambdas can be estimated in each iteration by controlling the expected false positive rate (EFPR) in case `automLambdas = TRUE`. This transforms the problem of selecting the tuning parameters λ_1 and λ_2 to the selection of the desired EFPR. In case `lambda2` is a single value and `lambda1` is a vector with several values, then lambda selection approaches implemented at `lambdaSelection` can also be used.

If `triangleCorrection = TRUE`, the weakest edges of estimated triangular motifs are further tested. The reason is that edges that complete triangular graph structures suffer an overestimation when applying the ADMM due to using regularized inverse procedures.

Value

An object of class `wfgl` containing the following components:

`path` adjacency matrices.

`omega` precision matrices.

`triangleCorrection` determines if triangle structures are tested.

`weakTriangEdges` weakest edges in triangle structures which have been tested.

`weakTriangEdgesPval` p-values for the weakest edge in triangle structures.

`diff_value` convergence control.

`iters` number of iterations used.

`corEst` dependence structure estimated measure used in the estimation to account for dependent datasets.

Author(s)

Caballe, Adria <a.caballe@sms.ed.ac.uk>, Natalia Bochkina and Claus Mayer.

References

Danaher, P., P. Wang, and D. Witten (2014). The joint graphical lasso for inverse covariance estimation across multiple classes. Journal of the Royal Statistical Society: Series B (Statistical Methodology) (2006), 1-20.

Boyd, S. (2010). Distributed Optimization and Statistical Learning via the Alternating Direction Method of Multipliers. Foundations and Trends in Machine Learning 3(1), 1-122.

See Also

`plot.wfgl` for graphical representation.
`wfrl` for weighted fused regression lasso.

Examples

```
# example to use of wfgl
EX2 <- pcorSimulatorJoint(nobs =50, nclusters = 3, nnodesxcluster = c(30, 30,30),
                          pattern = "pow", diffType = "cluster", dataDepend = "diag",
                         low.strength = 0.5, sup.strength = 0.9, pdiff = 0.5, nhubs = 5,
                          degree.hubs = 20,  nOtherEdges = 30, alpha = 2.3, plus = 0,
                         prob = 0.05, perturb.clust = 0.2, mu = 0, diagCCtype = "dicot",
                          diagNZ.strength = 0.6, mixProb = 0.5, probSign = 0.7,
                          exactZeroTh = 0.05)
## not run
#wfgl1 <- wfgl(EX2$D1, EX2$D2, lambda1 = 0.05, lambda2 = 0.1, paired = TRUE,
#             automLambdas = TRUE, sigmaEstimate = "CRmad", pairedEst = "Reg-based-sim",
#              maxiter = 30)
#print(wfgl1)
```

---

wfrl *joint estimation of multiple regression coefficient matrices*

Description

wfrl estimates jointly two regression coefficient matrices from multivariate normal distributed datasets using an ADMM based algorithm.

Usage

```
wfrl(D1, D2, lambda1, lambda2, automLambdas = TRUE, paired = TRUE,
 sigmaEstimate = "CRmad", maxiter=30, tol=1e-05, nsubset = 10000,
 rho = 1, rho.increment = 1, notOnlyLambda2 = TRUE)
```

Arguments

D1	list with the response variables. Two matrices in the list corresponding to the response variables of the two populations.
D2	list with the explanatory variables. Two matrices in the list corresponding to the explanatory variables of the two populations.
lambda1	tuning parameter for sparsity in the regression coefficients.
lambda2	tuning parameter for similarity between the regression coefficients in the two populations.
automLambdas	if TRUE the lambda's are estimated automatically with lambda1 and lambda2 being expected false positive rate levels.
paired	if TRUE, observations in D1 and D2 are assumed to be matched (n_1 must be equal to n_2).

sigmaEstimate	robust method used to estimate the variance of estimated partial correlations: name that uniquely identifies "mad", "IQR" or "CRmad" (default). This measure is used to automatically select the tuning parameter (when automLambdas = TRUE).
maxiter	maximum number of iterations for the ADMM algorithm.
tol	convergence tolerance.
nsubset	maximum number of estimated partial correlation coefficients (chosen randomly) used to select lambda1 and lambda2 automatically.
rho	regularization parameter used to compute matrix inverse by eigen value decomposition (default of 1).
rho.increment	default of 1.
notOnlyLambda2	if FALSE only lambda2 is found automatically.

Details

wfrl uses a weighted-fused least squares lasso maximum likelihood estimator by solving:

$$[\hat{\beta}_H, \hat{\beta}_T] = \arg\min_{\beta_H, \beta_T} \left[\frac{1}{2n} ||Y - \beta_H X||_2^2 + \frac{1}{2n} ||Q - \beta_T W||_2^2 + P_{\lambda_1, \lambda_2, V}(\beta) \right]$$

with

$$P_{\lambda_1, \lambda_2, V}(\beta) = \lambda_1 ||\beta_H||_1 + \lambda_1 ||\beta_T||_1 + \lambda_2 ||V \circ (\beta_T - \beta_H)||_1.$$

where λ_1 is the sparsity tuning parameter, λ_2 is the similarity tuning parameter, and $V = [v_{ij}]$ is a $p \times p$ matrix to weight λ_2 for each coefficient of the differential precision matrix. If datasets are independent (paired = "FALSE"), then it is assumed that $v_{ij} = 1$ for all pairs (i, j). Otherwise (paired = "TRUE"), weights are estimated in order to account for the dependence structure between datasets in the differential network estimation. An ADMM-type recursive algorithm is used to solve the optimization problem.

See details in wfgl for transforming the selection problem of the tuning parameters λ_1 and λ_2.

Value

An object of class wfrl containing the following components:

regCoef	regression coefficients.
path	non-zero structure for the regression coefficients.
diff_value	convergence control.
iters	number of iterations used.

Author(s)

Caballe, Adria <a.caballe@sms.ed.ac.uk>, Natalia Bochkina and Claus Mayer.

References

Danaher, P., P. Wang, and D. Witten (2014). The joint graphical lasso for inverse covariance estimation across multiple classes. Journal of the Royal Statistical Society: Series B (Statistical Methodology) (2006), 1-20.

Boyd, S. (2010). Distributed Optimization and Statistical Learning via the Alternating Direction Method of Multipliers. Foundations and Trends in Machine Learning 3(1), 1-122.

See Also

plot.wfrl for graphical representation.
wfgl for joint partial correlation estimation.

Examples

```
# example to use of wfrl
N <- 200
EX2 <- pcorSimulatorJoint(nobs = N, nclusters = 3, nnodesxcluster = c(30, 30,30),
                          pattern = "pow", diffType = "cluster", dataDepend = "diag",
                         low.strength = 0.5, sup.strength = 0.9, pdiff = 0.5, nhubs = 5,
                          degree.hubs = 20,  nOtherEdges = 30, alpha = 2.3, plus = 0,
                         prob = 0.05, perturb.clust = 0.2, mu = 0, diagCCtype = "dicot",
                          diagNZ.strength = 0.6, mixProb = 0.5, probSign = 0.7,
                          exactZeroTh = 0.05)

P           <- EX2$P
q           <- 50
BETA1       <- array(0, dim = c(P, q))
diag(BETA1) <- rep(0.35,q)
BETA2       <- BETA1
diag(BETA2)[c(1:floor(q/2))] <- 0
sigma2      <- 1.3
Q           <- scale(EX2$D1)
W           <- scale(EX2$D2)
X           <- Q%*%BETA1 + mvrnorm(N,rep(0,q),diag(rep(sigma2,q)))
Y           <- W%*%BETA2 + mvrnorm(N,rep(0,q),diag(rep(sigma2,q)))
D1          <- list(scale(X), scale(Y))
D2          <- list(scale(Q), scale(W))
## not run
#wfrl1      <- wfrl(D1, D2, lambda1 = 0.05, lambda2 = 0.05, automLambdas = TRUE, paired = FALSE,
#                sigmaEstimate = "CRmad", maxiter = 30, tol = 1e-05, nsubset = 10000, rho = 1,
#                    rho.increment = 1, notOnlyLambda2 = TRUE)
#print(wfrl1)
```

Index