qpgraph

April 19, 2010

`EcoliOxygen`	*Preprocessed microarray oxygen deprivation data and filtered RegulonDB data*

Description

The data consist of two objects, one containing normalized gene expression microarray data from Escherichia coli (E. coli) and the other containing a subset of filtered RegulonDB transcription regulatory relationships on E. coli.

Usage

```
data(EcoliOxygen)
```

Format

`gds680.eset`	`ExpressionSet` object containing n=43 experiments of various mutants under oxygen c
`filtered.regulon6.1`	Data frame object containing a subset of the E. coli transcriptional network from RegulonD

Source

Covert, M.W., Knight, E.M., Reed, J.L., Herrgard, M.J., and Palsson, B.O. Integrating high-throughput and computational data elucidates bacterial networks. *Nature*, 429(6987):92-96, 2004.

Gama-Castro, S., Jimenez-Jacinto, V., Peralta-Gil, M., Santos-Zavaleta, A., Penaloza-Spinola, M.I., Contreras-Moreira, B., Segura-Salazar, J., Muniz-Rascado, L., Martinez-Flores, I., Salgado, H., Bonavides-Martinez, C., Abreu-Goodger, C., Rodriguez-Penagos, C., Miranda-Rios, J., Morett, E., Merino, E., Huerta, A.M., Trevino-Quintanilla, L., and Collado-Vides, J. RegulonDB (version 6.0): gene regulation model of Escherichia coli K-12 beyond transcription, active (experimental) annotated promoters and Textpresso navigation. *Nucleic Acids Res.*, 36(Database issue):D120-124, 2008.

References

Castelo, R. and Roverato, A. Reverse engineering molecular regulatory networks from microarray data with qp-graphs. *J. Comp. Biol.*, 16(2):213-227, 2009.

Examples

```
data(EcoliOxygen)
```

---

qpAnyGraph *A graph*

Description

Obtains an undirected graph from a matrix of pairwise measurements

Usage

```
qpAnyGraph(measurementsMatrix, threshold=NULL, remove=c("below", "above"),
           topPairs=NULL, decreasing=TRUE, pairup.i=NULL, pairup.j=NULL,
           return.type=c("incidence.matrix", "edge.list", "graphNEL", "graphAM")
```

Arguments

`measurementsMatrix`	matrix of pairwise measurements.
`threshold`	threshold on the measurements below or above which pairs of variables are assumed to be disconnected in the resulting graph.
`remove`	direction of the removal with the threshold. It should be either `"below"` (default) or `"above"`.
`topPairs`	number of edges from the top of the ranking, defined by the pairwise measurements in `measurementsMatrix`, to use to form the resulting graph. This parameter is incompatible with a value different from `NULL` in `threshold`.
`decreasing`	logical, only applies when topPairs is set; if `TRUE` then the ranking is made in decreasing order; if `FALSE` then is made in increasing order.
`pairup.i`	subset of vertices to pair up with subset `pairup.j`
`pairup.j`	subset of vertices to pair up with subset `pairup.i`
`return.type`	type of data structure on which the resulting undirected graph should be returned. Either a logical incidence matrix with cells set to TRUE when the two indexing variables are connected in the graph (default), or a list of edges in a matrix where each row corresponds to one edge and the two columns contain the two vertices defining each edge, or a `graphNEL-class` object, or a `graphAM-class` object.

Details

This function requires the `graph` package when `return.type=graphNEL` or `return.type=graphAM`.

Value

The resulting undirected graph as either an incidence matrix, a `graphNEL` object or a `graphAM` object, depending on the value of the `return.type` parameter. Note that when some gold-standard graph is available for comparison, a value for the parameter `threshold` can be found by calculating a precision-recall curve with `qpPrecisionRecall` with respect to this gold-standard, and then using `qpPRscoreThreshold`. Parameters `threshold` and `topPairs` are mutually exclusive, that is, when we specify with `topPairs=n` that we want a graph with `n` edges then `threshold` cannot be used.

Author(s)

R. Castelo and A. Roverato

References

Castelo, R. and Roverato, A. A robust procedure for Gaussian graphical model search from microarray data with p larger than n, *J. Mach. Learn. Res.*, 7:2621-2650, 2006.

See Also

qpNrr qpAvgNrr qpEdgeNrr qpGraph qpGraphDensity qpClique qpPrecisionRecall qpPRscoreThreshold

Examples

```
nVar <- 50 # number of variables
maxCon <- 5  # maximum connectivity per variable
nObs <- 30 # number of observations to simulate

I <- qpRndGraph(n.vtx=nVar, n.bd=maxCon)
K <- qpI2K(I)

X <- qpSampleMvnorm(K, nObs)

pcc.estimates <- qpPCC(X)

# the higher the threshold
g <- qpAnyGraph(abs(pcc.estimates$R), threshold=0.9,
                remove="below")

# the sparser the qp-graph
(sum(g)/2) / (nVar*(nVar-1)/2)

# the lower the threshold
g <- qpAnyGraph(abs(pcc.estimates$R), threshold=0.5,
                remove="below")

# the denser the graph
(sum(g)/2) / (nVar*(nVar-1)/2)
```

`qpAvgNrr` *Average non-rejection rate estimation*

Description

Estimates average non-rejection rates for every pair of variables.

Usage

```
## S4 method for signature 'ExpressionSet':
qpAvgNrr(data, qOrders=4, nTests=100, alpha=0.05,
                                    pairup.i=NULL, pairup.j=NULL,
                                    type=c("arith.mean"), verbose=TRUE,
                                    identicalQs=TRUE, R.code.only=FALSE)
## S4 method for signature 'data.frame':
qpAvgNrr(data, qOrders=4, nTests=100, alpha=0.05,
                                 pairup.i=NULL, pairup.j=NULL,
                                 long.dim.are.variables=TRUE,
                                 type=c("arith.mean"), verbose=TRUE,
                                 identicalQs=TRUE, R.code.only=FALSE)
## S4 method for signature 'matrix':
qpAvgNrr(data, qOrders=4, nTests=100, alpha=0.05,
                             pairup.i=NULL, pairup.j=NULL,
                             long.dim.are.variables=TRUE,
                             type=c("arith.mean"), verbose=TRUE,
                             identicalQs=TRUE, R.code.only=FALSE)
```

Arguments

`data`	data set from where to estimate the average non-rejection rates. It can be an ExpressionSet object, a data frame or a matrix.
`qOrders`	either a number of partial-correlation orders or a vector of vector of particular orders to be employed in the calculation.
`nTests`	number of tests to perform for each pair for variables.
`alpha`	significance level of each test.
`pairup.i`	subset of vertices to pair up with subset `pairup.j`
`pairup.j`	subset of vertices to pair up with subset `pairup.i`
`long.dim.are.variables`	logical; if TRUE it is assumed that when the data is a data frame or a matrix, the longer dimension is the one defining the random variables; if FALSE, then random variables are assumed to be at the columns of the data frame or matrix.
`type`	type of average. By now only the arithmetic mean is available.
`verbose`	show progress on the calculations.
`identicalQs`	use identical conditioning subsets for every pair of vertices (default), otherwise sample a new collection of `nTests` subsets for each pair of vertices.
`R.code.only`	logical; if FALSE then the faster C implementation is used (default); if TRUE then only R code is executed.

Details

Note that when specifying a vector of particular orders `q`, these values should be in the range 1 to `min(p,n-3)`, where `p` is the number of variables and `n` the number of observations. The computational cost increases linearly within each `q` value and quadratically in `p`. When setting `identicalQs` to `FALSE` the computational cost may increase between 2 times and one order of magnitude (depending on `p` and `q`) while asymptotically the estimation of the non-rejection rate converges to the same value.

Value

A symmetric matrix of estimated average non-rejection rates with the diagonal set to `NA`. When using the arguments `pairup.i` and `pairup.j`, those cells outside the constraint pairs will get also a `NA` value.

Author(s)

R. Castelo and A. Roverato

References

Castelo, R. and Roverato, A. Reverse engineering molecular regulatory networks from microarray data with qp-graphs. *J. Comp. Biol.*, 16(2):213-227, 2009.

See Also

`qpNrr` `qpEdgeNrr` `qpHist` `qpGraphDensity` `qpClique`

Examples

```
nVar <- 50 # number of variables
maxCon <- 5  # maximum connectivity per variable
nObs <- 30 # number of observations to simulate

I <- qpRndGraph(n.vtx=nVar, n.bd=maxCon)
K <- qpI2K(I)

X <- qpSampleMvnorm(K, nObs)

avgnrr.estimates <- qpAvgNrr(X, verbose=FALSE)

summary(avgnrr.estimates[upper.tri(avgnrr.estimates) & I])

summary(avgnrr.estimates[upper.tri(avgnrr.estimates) & !I])
```

`qpCItest` *Conditional independence test*

Description

Performs a conditional independence test between two variables given a conditioning set.

Usage

```
## S4 method for signature 'ExpressionSet':
qpCItest(data, i=1, j=2, Q=c(), R.code.only=FALSE)
## S4 method for signature 'data.frame':
qpCItest(data, i=1, j=2, Q=c(),
                                 long.dim.are.variables=TRUE, R.code.only=FALSE)
## S4 method for signature 'matrix':
qpCItest(data, N=NULL, i=1, j=2, Q=c(),
                             long.dim.are.variables=TRUE, R.code.only=FALSE)
```

Arguments

`data`	data set where the test should be performed. It can be either an `ExpressionSet` object, a data frame, or a matrix. If it is a matrix and the matrix is squared then this function assumes the matrix is the sample covariance matrix of the data and the sample size parameter `N` should be provided.
`N`	number of observations in the data set. Only necessary when the sample covariance matrix is provided through the `data` parameter.
`i`	index or name of one of the two variables.
`j`	index or name of the other variable.
`Q`	indexes or names of the variables forming the conditioning set.
`long.dim.are.variables`	logical; if TRUE it is assumed that when data are in a data frame or in a matrix, the longer dimension is the one defining the random variables (default); if FALSE, then random variables are assumed to be at the columns of the data frame or matrix.
`R.code.only`	logical; if FALSE then the faster C implementation is used (default); if TRUE then only R code is executed.

Details

Note that the size of possible `Q` sets should be in the range 1 to `min(p,n-3)`, where `p` is the number of variables and `n` the number of observations. The computational cost increases linearly with the number of variables in `Q`.

Value

A list with two members, the t-statistic value and the p-value on rejecting the null hypothesis of independence.

Author(s)

R. Castelo and A. Roverato

References

Castelo, R. and Roverato, A. A robust procedure for Gaussian graphical model search from microarray data with p larger than n, *J. Mach. Learn. Res.*, 7:2621-2650, 2006.

See Also

`qpNrr` `qpEdgeNrr`

Examples

```
# in this graph 3 is conditionally independent of 4 given 1 AND 2

I <- matrix(c(FALSE,  TRUE,  TRUE,  TRUE,
              TRUE,  FALSE,  TRUE,  TRUE,
              TRUE,   TRUE, FALSE, FALSE,
              TRUE,   TRUE, FALSE, FALSE), nrow=4, ncol=4, byrow=TRUE)
K <- qpI2K(I)
```

```
X <- qpSampleMvnorm(K, N=100)

qpCItest(X, i=3, j=4, Q=1, long.dim.are.variables=FALSE)

qpCItest(X, i=3, j=4, Q=c(1,2), long.dim.are.variables=FALSE)
```

`qpCliqueNumber` *Clique number*

Description

Calculates the size of the largest maximal clique (the so-called clique number or maximum clique size) in a given undirected graph.

Usage

```
qpCliqueNumber(g, exact.calculation=TRUE, return.vertices=FALSE,
               approx.iter=100, verbose=TRUE)
```

Arguments

`g`	either a `graphNEL` object or an incidence matrix of the given undirected graph.
`exact.calculation`	logical; if TRUE then the exact clique number is calculated; if FALSE then a lower bound is given instead.
`return.vertices`	logical; if TRUE a set of vertices forming a maximal clique of maximum size is returned; if FALSE only the maximum clique size is returned.
`approx.iter`	number of iterations to be employed in the calculation of the lower bound (i.e., only applies when `exact.calculation=FALSE`.
`verbose`	show progress on calculations.

Details

The calculation of the clique number of an undirected graph is an NP-complete problem which means that its computational cost is bounded by an exponential running time (Pardalos and Xue, 1994). The current implementation uses C code from the GNU GPL Cliquer library by Niskanen and Ostergard (2003) based on the, probably the fastest to date, algorithm by Ostergard (2002).

The lower bound on the maximum clique size is calculated by ranking the vertices by their connectivity degree, put the first vertex in a set and go through the rest of the ranking adding those vertices to the set that form a clique with the vertices currently within the set. Once the entire ranking has been examined a large clique should have been built and eventually one of the largests ones. This process is repeated a number of times (`approximation.iterations`) each of which the ranking is altered with increasing levels of randomness acyclically (altering 1 to $p$ vertices and again). Larger values of `approximation.iterations` should provide tighter lower bounds and eventually the exact maximum clique size.

Value

the size of the largest maximal clique in the given graph, also known as its clique number.

Author(s)

R. Castelo

References

Castelo, R. and Roverato, A. A robust procedure for Gaussian graphical model search from microarray data with p larger than n. *J. Mach. Learn. Res.*, 7:2621-2650, 2006.

Niskanen, S. Ostergard, P. Cliquer User's Guide, Version 1.0. Communications Laboratory, Helsinki University of Technology, Espoo, Finland, Tech. Rep. T48, 2003. (`http://users.tkk.fi/~pat/cliquer.html`)

Ostergard, P. A fast algorithm for the maximum clique problem. Discrete Appl. Math. 120:197-207, 2002.

Pardalos, P.M. and Xue, J. The maximum clique problem. *J. Global Optim.*, 4:301-328, 1994.

See Also

`qpClique`

Examples

```
nVertices <- 50 # number of vertices
maxCon <- 5  # maximum connectivity per vertex

I <- qpRndGraph(n.vtx=nVertices, n.bd=maxCon)

qpCliqueNumber(I, verbose=FALSE)

maxCon <- 10  # maximum connectivity per vertex

I <- qpRndGraph(n.vtx=nVertices, n.bd=maxCon)

qpCliqueNumber(I, verbose=FALSE)
```

`qpClique` *Complexity of the resulting qp-graphs*

Description

Calculates and plots the size of the largest maximal clique (the so-called clique number or maximum clique size) as function of the non-rejection rate.

Usage

```
qpClique(nrrMatrix, N=NA, threshold.lim=c(0,1), breaks=5, plot=TRUE,
          exact.calculation=TRUE, approx.iter=100,
          qpCliqueOutput=NULL, density.digits=0,
          logscale.clqsize=FALSE,
          titleclq="maximum clique size as function of threshold",
          verbose=FALSE)
```

Arguments

`nrrMatrix`	matrix of non-rejection rates.
`N`	number of observations from where the non-rejection rates were estimated.
`threshold.lim`	range of threshold values on the non-rejection rate.
`breaks`	either a number of threshold bins or a vector of threshold breakpoints.
`plot`	logical; if TRUE makes a plot of the result; if FALSE it does not.
`exact.calculation`	logical; if TRUE then the exact clique number is calculated; if FALSE then a lower bound is given instead.
`approx.iter`	number of iterations to be employed in the calculation of the lower bound (i.e., only applies when `exact.calculation=FALSE`).
`qpCliqueOutput`	output from a previous call to `qpClique`. This allows one to plot the result changing some of the plotting parameters without having to do the calculation again.
`density.digits`	number of digits in the reported graph densities.
`logscale.clqsize`	logical; if TRUE then the scale for the maximum clique size is logarithmic which is useful when working with more than 1000 variables; FALSE otherwise (default).
`titleclq`	main title to be shown in the plot.
`verbose`	show progress on calculations.

Details

The estimate of the complexity of the resulting qp-graphs is calculated as the area enclosed under the curve of maximum clique sizes.

The maximum clique size, or clique number, is obtained by calling the function `qpCliqueNumber` The calculation of the clique number of an undirected graph is an NP-complete problem which means that its computational cost is bounded by an exponential running time (Pardalos and Xue, 1994). Therefore, giving breakpoints between 0.95 and 1.0 may result into very dense graphs which can lead to extremely long execution times. If it is necessary to look at that range of breakpoints it is recommended either to use the lower bound on the clique number (`exact.calculation=FALSE`) or to look at `qpGraphDensity`.

Value

A list with the maximum clique size and graph density as function of threshold, an estimate of the complexity of the resulting qp-graphs across the thresholds, the threshold on the non-rejection rate that provides a maximum clique size strictly smaller than the sample size N and the resulting maximum clique size.

Author(s)

R. Castelo and A. Roverato

References

Castelo, R. and Roverato, A. A robust procedure for Gaussian graphical model search from microarray data with p larger than n. *J. Mach. Learn. Res.*, 7:2621-2650, 2006.

Pardalos, P.M. and Xue, J. The maximum clique problem. *J. Global Optim.*, 4:301-328, 1994.

See Also

`qpCliqueNumber` `qpGraphDensity`

Examples

```
nVar <- 50 # number of variables
maxCon <- 5  # maximum connectivity per variable
nObs <- 30 # number of observations to simulate

I <- qpRndGraph(n.vtx=nVar, n.bd=maxCon)
K <- qpI2K(I)

X <- qpSampleMvnorm(K, nObs)

# the higher the q the less complex the qp-graph

nrr.estimates <- qpNrr(X, q=1, verbose=FALSE)

qpClique(nrr.estimates, plot=FALSE)$complexity

nrr.estimates <- qpNrr(X, q=5, verbose=FALSE)

qpClique(nrr.estimates, plot=FALSE)$complexity
```

`qpDscale` *Scale a matrix by its diagonal*

Description

Scales the values of a matrix by the values on the diagonal.

Usage

```
qpDscale(V)
```

Arguments

`V`	matrix with numerical values.

Details

The scaling of the values of the matrix by its diagonal is made using the outer product of the inverted square root of the diagonal.

Value

A scaled matrix such that the diagonal values become the unit.

Author(s)

R. Castelo and A. Roverato

References

Castelo, R. and Roverato, A. A robust procedure for Gaussian graphical model search from microarray data with p larger than n. *J. Mach. Learn. Res.*, 7:2621-2650, 2006.

See Also

qpK2R

Examples

```
m <- matrix(1:9, nrow=3)

m

qpDscale(m)
```

---

qpEdgeNrr *Non-rejection rate estimation for a pair of variables*

Description

Estimates non-rejection rate for one pair of variables.

Usage

```
## S4 method for signature 'ExpressionSet':
qpEdgeNrr(data, i=1, j=2, q=1, nTests=100,
                                    alpha=0.05, R.code.only=FALSE)
## S4 method for signature 'data.frame':
qpEdgeNrr(data, i=1, j=2, q=1, nTests=100,
                                 alpha=0.05, long.dim.are.variables=TRUE,
                                 R.code.only=FALSE)
## S4 method for signature 'matrix':
qpEdgeNrr(data, N=NULL, i=1, j=2, q=1, nTests=100,
                             alpha=0.05, long.dim.are.variables=TRUE,
                             R.code.only=FALSE)
```

Arguments

`data`	data set from where the non-rejection rate should be estimated. It can be either an `ExpressionSet` object, a data frame, or a matrix. If it is a matrix and the matrix is squared then this function assumes the matrix is the sample covariance matrix of the data and the sample size parameter `N` should be provided.

`N`	number of observations in the data set. Only necessary when the sample covariance matrix is provided through the `data` parameter.
`i`	index or name of one of the two variables.
`j`	index or name of the other variable.
`q`	partial-correlation order.
`nTests`	number of tests to perform for each pair for variables.
`alpha`	significance level of each test.
`long.dim.are.variables`	logical; if TRUE it is assumed that when data are in a data frame or in a matrix, the longer dimension is the one defining the random variables (default); if FALSE, then random variables are assumed to be at the columns of the data frame or matrix.
`R.code.only`	logical; if FALSE then the faster C implementation is used (default); if TRUE then only R code is executed.

Details

The estimation of the non-rejection rate for a pair of variables is calculated as the fraction of tests that accept the null hypothesis of independence given a set of randomly sampled q-order conditionals.

Note that the possible values of `q` should be in the range 1 to `min(p,n-3)`, where `p` is the number of variables and `n` the number of observations. The computational cost increases linearly with `q`.

Value

An estimate of the non-rejection rate for the particular given pair of variables.

Author(s)

R. Castelo and A. Roverato

References

Castelo, R. and Roverato, A. A robust procedure for Gaussian graphical model search from microarray data with p larger than n, *J. Mach. Learn. Res.*, 7:2621-2650, 2006.

See Also

`qpNrr` `qpAvgNrr` `qpHist` `qpGraphDensity` `qpClique`

Examples

```
# in this graph 3 is conditional independent of 4 given 1 AND 2

I <- matrix(c(FALSE,  TRUE,  TRUE,  TRUE,
              TRUE,  FALSE,  TRUE,  TRUE,
              TRUE,   TRUE, FALSE, FALSE,
              TRUE,   TRUE, FALSE, FALSE), nrow=4, ncol=4, byrow=TRUE)
K <- qpI2K(I)

X <- qpSampleMvnorm(K, N=100)
```

```
qpEdgeNrr(X, i=3, j=4, q=1, long.dim.are.variables=FALSE)

qpEdgeNrr(X, i=3, j=4, q=2, long.dim.are.variables=FALSE)
```

---

qpFunctionalCoherence

Functional coherence estimation

Description

Estimates functional coherence for a given transcriptional regulatory network.

Usage

```
qpFunctionalCoherence(I, TFgenes, chip, minRMsize=5, verbose=FALSE)
```

Arguments

`I`	incidence matrix of the undirected graph representing the transcriptional regulatory network.
`TFgenes`	vector of transcription factor gene (matching the genes at the rows and column names of `I`.
`chip`	name of the `.db` package containing the Gene Ontology (GO) annotations.
`minRMsize`	minimum size of the target gene set in each regulatory module where functional enrichment will be calculated and thus where functional coherence will be estimated.
`verbose`	logical; if TRUE the function will show progress on the calculations; if FALSE the function will remain quiet (default).

Details

This function estimates the functional coherence of a transcriptional regulatory network represented by means of an undirected graph encoded by an incidence matrix and of a set of transcription factor genes. The functional coherence of a transcriptional regulatory network is calculated as specified by Castelo and Roverato (2009) and corresponds to the distribution of individual functional coherence values of every of the regulatory modules of the network each of them defined as a transcription factor and its set of putatively regulated target genes. In the calculation of the functional coherence value of a regulatory module, Gene Ontology (GO) annotations are employed through the given annotation `.db` package and the conditional hyper-geometric test implemented in the `GOstats` package from Bioconductor.

Value

A list with three slots, a first one containing the transcriptional regulatory network as a list of regulatory modules and their targets, a second one containing this same network but including only those modules with GO BP annotations and a third one consisting of a vector of functional coherence values.

Author(s)

R. Castelo and A. Roverato

References

Castelo, R. and Roverato, A. Reverse engineering molecular regulatory networks from microarray data with qp-graphs. *J. Comp. Biol.*, 16(2):213-227, 2009.

See Also

qpAvgNrr qpGraph

Examples

```
library(annotate)
library(org.EcK12.eg.db)

# load RegulonDB data from this package
data(EcoliOxygen)

# pick two TFs from the RegulonDB data in this package

TFgenes <- c("mhpR", "iscR")

# get their Entrez Gene Identifiers
TFgenesEgIDs <- unlist(mget(TFgenes, revmap(getAnnMap("SYMBOL", "org.EcK12.eg.db"))))

# get all genes involved in their regulatory modules from
# the RegulonDB data in this package
mt <- match(filtered.regulon6.1[,"EgID_TF"], TFgenesEgIDs)

allGenes <- as.character(unique(as.vector(
            as.matrix(filtered.regulon6.1[!is.na(mt),
                                          c("EgID_TF","EgID_TG")]))))

mtTF <- match(filtered.regulon6.1[,"EgID_TF"],allGenes)
mtTG <- match(filtered.regulon6.1[,"EgID_TG"],allGenes)

# select the corresponding subset of the RegulonDB data in this package
subset.filtered.regulon6.1 <- filtered.regulon6.1[!is.na(mtTF) & !is.na(mtTG),]
TFi <- match(subset.filtered.regulon6.1[,"EgID_TF"], allGenes)
TGi <- match(subset.filtered.regulon6.1[,"EgID_TG"], allGenes)
subset.filtered.regulon6.1 <- cbind(subset.filtered.regulon6.1,
                                    idx_TF=TFi, idx_TG=TGi)

# build an incidence matrix representing the transcriptional regulatory
# relationships from these regulatory modules
p <- length(allGenes)
incidenceMatrix <- matrix(FALSE, nrow=p, ncol=p)
rownames(incidenceMatrix) <- colnames(incidenceMatrix) <- allGenes
idxTFTG <- as.matrix(subset.filtered.regulon6.1[,c("idx_TF","idx_TG")])
incidenceMatrix[idxTFTG] <-
  incidenceMatrix[cbind(idxTFTG[,2],idxTFTG[,1])] <- TRUE

# calculate functional coherence on these regulatory modules
fc <- qpFunctionalCoherence(incidenceMatrix, TFgenesEgIDs, "org.EcK12.eg.db")

print(sprintf("the %s module has a FC value of %.2f",
              mget(names(fc$functionalCoherenceValues),org.EcK12.egSYMBOL),
              fc$functionalCoherenceValues))
```

`qpGetCliques` *Clique list*

Description

Finds the set of (maximal) cliques of a given undirected graph.

Usage

```
qpGetCliques(g, clqspervtx=FALSE, verbose=TRUE)
```

Arguments

`g`	either a `graphNEL` object or an incidence matrix of the given undirected graph.
`clqspervtx`	logical; if TRUE then the resulting list returned by the function includes additionally p entries at the beginning (p=number of variables) each corresponding to a vertex in the graph and containing the indices of the cliques where that vertex belongs to; if FALSE these additional entries are not included (default).
`verbose`	show progress on calculations.

Details

To find the list of all (maximal) cliques in an undirected graph is an NP-complete problem which means that its computational cost is bounded by an exponential running time (Karp, 1972). For this reason, this is an extremely time and memory consuming computation for large dense graphs. The current implementation uses C code from the GNU GPL Cliquer library by Niskanen and Ostergard (2003).

Value

A list of maximal cliques. When `clqspervtx=TRUE` the first p entries (p=number of variables) contain, each of them, the indices of the cliques where that particular vertex belongs to.

Author(s)

R. Castelo

References

Castelo, R. and Roverato, A. A robust procedure for Gaussian graphical model search from microarray data with p larger than n. *J. Mach. Learn. Res.*, 7:2621-2650, 2006.

Niskanen, S. Ostergard, P. Cliquer User's Guide, Version 1.0. Communications Laboratory, Helsinki University of Technology, Espoo, Finland, Tech. Rep. T48, 2003. (`http://users.tkk.fi/~pat/cliquer.html`)

Karp, R.M. Reducibility among combinatorial problems. In R.E. Miller and J.W. Thatcher (eds.): *Complexity of Computer Computations*, 85-103, New York: Plenum, 1972.

See Also

`qpCliqueNumber` `qpIPF`

Examples

```
nVertices <- 50 # number of vertices
maxCon <- 5  # maximum connectivity per vertex

I <- qpRndGraph(n.vtx=nVertices, n.bd=maxCon)

clqs <- qpGetCliques(I, verbose=FALSE)

summary(unlist(lapply(clqs, length)))

maxCon <- 10  # maximum connectivity per vertex

I <- qpRndGraph(n.vtx=nVertices, n.bd=maxCon)

clqs <- qpGetCliques(I, verbose=FALSE)

summary(unlist(lapply(clqs, length)))
```

`qpGraphDensity` *Densities of resulting qp-graphs*

Description

Calculates and plots the graph density as function of the non-rejection rate.

Usage

```
qpGraphDensity(nrrMatrix, threshold.lim=c(0,1), breaks=5,
               plot=TRUE, qpGraphDensityOutput=NULL,
               density.digits=0,
               titlegd="graph density as function of threshold")
```

Arguments

`nrrMatrix`	matrix of non-rejection rates.
`threshold.lim`	range of threshold values on the non-rejection rate.
`breaks`	either a number of threshold bins or a vector of threshold breakpoints.
`plot`	logical; if TRUE makes a plot of the result; if FALSE it does not.
`qpGraphDensityOutput`	output from a previous call to `qpGraphDensity`. This allows one to plot the result changing some of the plotting parameters without having to do the calculation again.
`density.digits`	number of digits in the reported graph densities.
`titlegd`	main title to be shown in the plot.

Details

The estimate of the sparseness of the resulting qp-graphs is calculated as one minus the area enclosed under the curve of graph densities.

Value

A list with the graph density as function of threshold and an estimate of the sparseness of the resulting qp-graphs across the thresholds.

Author(s)

R. Castelo and A. Roverato

References

Castelo, R. and Roverato, A. A robust procedure for Gaussian graphical model search from microarray data with p larger than n, *J. Mach. Learn. Res.*, 7:2621-2650, 2006.

See Also

qpNrr qpAvgNrr qpEdgeNrr qpClique

Examples

```
nVar <- 50 # number of variables
maxCon <- 5  # maximum connectivity per variable
nObs <- 30 # number of observations to simulate

I <- qpRndGraph(n.vtx=nVar, n.bd=maxCon)
K <- qpI2K(I)

X <- qpSampleMvnorm(K, nObs)

# the higher the q the sparser the qp-graphs

nrr.estimates <- qpNrr(X, q=1, verbose=FALSE)

qpGraphDensity(nrr.estimates, plot=FALSE)$sparseness

nrr.estimates <- qpNrr(X, q=5, verbose=FALSE)

qpGraphDensity(nrr.estimates, plot=FALSE)$sparseness
```

`qpgraph-package` *The q-order partial correlation graph learning software, qpgraph.*

Description

q-order partial correlation graphs, or qp-graphs for short, are undirected Gaussian graphical Markov models that represent q-order partial correlations. They are useful for learning undirected graphical Gaussian Markov models from data sets where the number of random variables p exceeds the available sample size n as, for instance, in the case of microarray data where they can be employed to reverse engineer a molecular regulatory network.

Details

Package:	qp
Version:	1.1.4
Date:	15-10-2009
biocViews:	Microarray, GeneExpression, Transcription, Pathways, Bioinformatics, GraphsAndNetworks
Suggests:	mvtnorm, graph, Rgraphviz, annotate, genefilter, Category (>= 2.9.7), org.EcK12.eg.db (>= 2.2.6), GOstats
License:	GPL (>= 2)
URL:	`http://functionalgenomics.upf.edu/qpgraph`

Functions

- `qpNrr` estimates non-rejection rates for every pair of variables.
- `qpAvgNrr` estimates average non-rejection rates for every pair of variables.
- `qpEdgeNrr` estimate the non-rejection rate of one pair of variables.
- `qpCItest` performs a conditional independence test between two variables given a conditioning set.
- `qpHist` plots the distribution of non-rejection rates.
- `qpGraph` obtains a qp-graph from a matrix of non-rejection rates.
- `qpAnyGraph` obtains an undirected graph from a matrix of pairwise measurements.
- `qpGraphDensity` calculates and plots the graph density as function of the non-rejection rate.
- `qpCliqueNumber` calculates the size of the largest maximal clique (the so-called clique number or maximum clique size) in a given undirected graph.
- `qpClique` calculates and plots the size of the largest maximal clique (the so-called clique number or maximum clique size) as function of the non-rejection rate.
- `qpGetCliques` finds the set of (maximal) cliques of a given undirected graph.
- `qpIPF` performs maximum likelihood estimation of a sample covariance matrix given the independence constraints from an input list of (maximal) cliques.
- `qpPAC` estimates partial correlation coefficients and corresponding P-values for each edge in a given undirected graph, from an input data set.
- `qpPCC` estimates pairwise Pearson correlation coefficients and their corresponding P-values between all pairs of variables from an input data set.
- `qpRndGraph` builds a random undirected graph with a bounded maximum connectivity degree on every vertex.
- `qpSampleMvnorm` samples independent observations from a multivariate normal distribution with a given mean vector and a given concentration matrix.
- `qpI2K` builds a random concentration matrix containing zeroes on those entries associated to pairs of variables that are disconnected on a given undirected graph.
- `qpK2R` obtains the partial correlation coefficients from a given concentration matrix.
- `qpPrecisionRecall` calculates the precision-recall curve for a given measure of association between all pairs of variables in a matrix.
- `qpPRscoreThreshold` calculates the score threshold at a given precision or recall level from a given precision-recall curve.

- `qpImportNrr` imports non-rejection rates.
- `qpFunctionalCoherence` estimates functional coherence of using Gene Ontology annotations.

This package provides an implementation of the procedures described in (Castelo and Roverato, 2006, 2009). An example of its use for reverse-engineering of transcriptional regulatory networks from microarray data is available in the vignette `qpTxRegNet`. This package is a contribution to the Bioconductor (Gentleman et al., 2004) and gR (Lauritzen, 2002) projects.

Author(s)

R. Castelo and A. Roverato

Maintainer: R. Castelo <robert.castelo@upf.edu>

References

Castelo, R. and Roverato, A. A robust procedure for Gaussian graphical model search from microarray data with p larger than n. *J. Mach. Learn. Res.*, 7:2621-2650, 2006.

Castelo, R. and Roverato, A. Reverse engineering molecular regulatory networks from microarray data with qp-graphs. *J. Comput. Biol.*, 16(2):213-227, 2009.

Gentleman, R.C., Carey, V.J., Bates, D.M., Bolstad, B., Dettling, M., Dudoit, S., Ellis, B., Gautier, L., Ge, Y., Gentry, J., Hornik, K. Hothorn, T., Huber, W., Iacus, S., Irizarry, R., Leisch, F., Li, C., Maechler, M. Rosinni, A.J., Sawitzki, G., Smith, C., Smyth, G., Tierney, L., Yang, T.Y.H. and Zhang, J. Bioconductor: open software development for computational biology and bioinformatics. *Genome Biol.*, 5:R80, 2004.

Lauritzen, S.L. (2002). gRaphical Models in R. *R News*, 3(2)39.

`qpGraph` *The qp-graph*

Description

Obtains a qp-graph from a matrix of non-rejection rates

Usage

```
qpGraph(nrrMatrix, threshold=NULL, topPairs=NULL, pairup.i=NULL, pairup.j=NULL,
        return.type=c("incidence.matrix", "edge.list", "graphNEL", "graphAM"))
```

Arguments

`nrrMatrix`	matrix of non-rejection rates.
`threshold`	threshold on the non-rejection rate above which pairs of variables are assumed to be disconnected in the resulting qp-graph.
`topPairs`	number of edges from the top of the ranking, defined by the non-rejection rates in `nrrMatrix`, to use to form the resulting qp-graph. This parameter is incompatible with a value different from `NULL` in `threshold`.
`pairup.i`	subset of vertices to pair up with subset `pairup.j`
`pairup.j`	subset of vertices to pair up with subset `pairup.i`

`return.type` type of data structure on which the resulting undirected graph should be returned. Either a logical incidence matrix with cells set to TRUE when the two indexing variables are connected in the qp-graph (default), or a list of edges in a matrix where each row corresponds to one edge and the two columns contain the two vertices defining each edge, or a `graphNEL-class` object, or a `graphAM-class` object.

Details

This function requires the `graph` package when `return.type=graphNEL` or `return.type=graphAM`.

Value

The resulting qp-graph as either an incidence matrix, a `graphNEL` object or a `graphAM` object, depending on the value of the `return.type` parameter. Note that when some gold-standard graph is available for comparison, a value for the parameter `threshold` can be found by calculating a precision-recall curve with `qpPrecisionRecall` with respect to this gold-standard, and then using `qpPRscoreThreshold`. Parameters `threshold` and `topPairs` are mutually exclusive, that is, when we specify with `topPairs=n` that we want a qp-graph with `n` edges then `threshold` cannot be used.

Author(s)

R. Castelo and A. Roverato

References

Castelo, R. and Roverato, A. A robust procedure for Gaussian graphical model search from microarray data with p larger than n, *J. Mach. Learn. Res.*, 7:2621-2650, 2006.

See Also

`qpNrr` `qpAvgNrr` `qpEdgeNrr` `qpAnyGraph` `qpGraphDensity` `qpClique` `qpPrecisionRecall` `qpPRscoreThreshold`

Examples

```
nVar <- 50 # number of variables
maxCon <- 5  # maximum connectivity per variable
nObs <- 30 # number of observations to simulate

I <- qpRndGraph(n.vtx=nVar, n.bd=maxCon)
K <- qpI2K(I)

X <- qpSampleMvnorm(K, nObs)

nrr.estimates <- qpNrr(X, q=5, verbose=FALSE)

# the higher the threshold
g <- qpGraph(nrr.estimates, threshold=0.9)

# the denser the qp-graph
(sum(g)/2) / (nVar*(nVar-1)/2)

# the lower the threshold
```

```
g <- qpGraph(nrr.estimates, threshold=0.5)

# the sparser the qp-graph
(sum(g)/2) / (nVar*(nVar-1)/2)
```

---

qpHist — *Histograms of non-rejection rates*

Description

Plots the distribution of non-rejection rates.

Usage

```
qpHist(nrrMatrix, K=NULL,
       titlehist = "all estimated\nnon-rejection rates", freq=TRUE)
```

Arguments

`nrrMatrix`	matrix of non-rejection rates.
`K`	concentration matrix of the generative distribution (whenever available).
`titlehist`	main title of the histogram(s).
`freq`	logical; if TRUE, the histograms show frequencies (counts) of occurrence of the different non-rejection rate values; if FALSE, then probability densities are plotted

Details

This function plots histograms using the R-function `hist` and therefore the way they are displayed follows that of this R-function.

Value

None

Author(s)

R. Castelo and A. Roverato

References

Castelo, R. and Roverato, A. A robust procedure for Gaussian graphical model search from microarray data with p larger than n, *J. Mach. Learn. Res.*, 7:2621-2650, 2006.

See Also

`qpNrr` `qpAvgNrr` `qpEdgeNrr` `qpGraphDensity` `qpClique`

Examples

```
nVar <- 50 # number of variables
maxCon <- 5  # maximum connectivity per variable
nObs <- 30 # number of observations to simulate

I <- qpRndGraph(n.vtx=nVar, n.bd=maxCon)
K <- qpI2K(I)

X <- qpSampleMvnorm(K, nObs)

nrr.estimates <- qpNrr(X, q=5, verbose=FALSE)

qpHist(nrr.estimates, K)
```

`qpI2K` *Random concentration matrix*

Description

Builds a random concentration matrix containing zeroes on those entries associated to pairs of variables that are disconnected on a given undirected graph.

Usage

```
qpI2K(I, verbose=FALSE, R.code.only=FALSE)
```

Arguments

`I`	incidence matrix of an undirected graph.
`verbose`	show progress on the calculations.
`R.code.only`	logical; if FALSE then the faster C implementation is used (default); if TRUE then only R code is executed.

Details

The random concentration matrix is built by first generating a matrix of random correlations using the method from Marsaglia and Oltkin (1984). Second, this matrix is inverted to obtain an initial random covariance matrix. Third, this covariance matrix is adjusted to the independence constraints of the input undirected graph by using the function `qpIPF` and finally is inverted to obtain the final random concentration matrix.

Value

A random concentration matrix with zeroes at the empty adjacencies of the undirected graph defined by the input incidence matrix.

Author(s)

R. Castelo and A. Roverato

References

Castelo, R. and Roverato, A. A robust procedure for Gaussian graphical model search from microarray data with p larger than n. *J. Mach. Learn. Res.*, 7:2621-2650, 2006.

See Also

`qpSampleMvnorm` `qpK2R`

Examples

```
nVar <- 50 # number of variables
maxCon <- 5  # maximum connectivity per variable

I <- qpRndGraph(n.vtx=nVar, n.bd=maxCon)
K <- qpI2K(I)

realI <- K != 0
diag(realI) <- FALSE

sum(realI) / 2

sum(I) / 2

# all present edges (dependencies) in realI must be in I
identical(I & realI, realI)

# all missing edges (independencies) in I must be in realI
identical(!I & !realI, !I)
```

`qpImportNrr` *Import non-rejection rates*

Description

Imports non-rejection rates from an external flat file.

Usage

```
qpImportNrr(filename, nTests)
```

Arguments

`filename`	name of the flat file with the data on the non-rejection rates.
`nTests`	number of tests performed in the estimation of these non-rejection rates.

Details

This function expects a flat file with three tab-separated columns corresponding to, respectively, 0-based index of one of the variables, 0-based index of the other variable, number of non-rejected tests for the pair of variables of that row in the text file. An example of a few lines of that file would be:

```
6         3         95
6         4         98
6         5         23
7         0         94
7         1         94
```

After reading the file the function builds a matrix of non-rejection rates by dividing the number of non-rejected tests by `nTests`. Note that if the flat file to be imported would eventually have directly the rates instead of the number of tests, these can be also imported by setting `nTests=1`.

This function is thought to be used to read files obtained from the standalone parallel version of `qpNrr` which can be downloaded from `http://functionalgenomics.upf.edu/qp`.

Value

A symmetric matrix of non-rejection rates with the diagonal set to the `NA` value.

Author(s)

R. Castelo and A. Roverato

References

Castelo, R. and Roverato, A. A robust procedure for Gaussian graphical model search from microarray data with p larger than n, *J. Mach. Learn. Res.*, 7:2621-2650, 2006.

See Also

`qpNrr`

`qpIPF` *Iterative proportional fitting algorithm*

Description

Performs maximum likelihood estimation of a sample covariance matrix given the independence constraints from in input list of (maximal) cliques.

Usage

```
qpIPF(vv, clqlst, tol = 0.001, verbose = FALSE, R.code.only = FALSE)
```

Arguments

`vv`	input matrix, in the context of this package, the sample covariance matrix.
`clqlst`	list of maximal cliques obtained from an undirected graph by using the function `qpGetCliques`.
`tol`	tolerance under which the iterative algorithm stops.
`verbose`	show progress on calculations.
`R.code.only`	logical; if FALSE then the faster C implementation is used (default); if TRUE then only R code is executed.

Details

The Iterative proportional fitting algorithm (see, Whittaker, 1990, pp. 182-185) adjusts the input matrix to the independence constraints in the undirected graph from where the input list of cliques belongs to, by going through each of the cliques fitting the marginal distribution over the clique for the fixed conditional distribution of the clique. It stops when the adjusted matrix at the current iteration differs from the matrix at the previous iteration in less or equal than a given tolerance value.

Value

The input matrix adjusted to the constraints imposed by the list of cliques, i.e., a maximum likelihood estimate of the sample covariance matrix that includes the independence constraints encoded in the undirected graph formed by the given list of cliques.

Author(s)

R. Castelo and A. Roverato

References

Castelo, R. and Roverato, A. A robust procedure for Gaussian graphical model search from microarray data with p larger than n. *J. Mach. Learn. Res.*, 7:2621-2650, 2006.

Whittaker, J. *Graphical models in applied multivariate statistics.* Wiley, 1990.

See Also

`qpGetCliques` `qpPAC`

Examples

```
nVar <- 50 # number of variables
maxCon <- 5  # maximum connectivity per variable
nObs <- 10 # number of observations to simulate

I <- qpRndGraph(n.vtx=nVar, n.bd=maxCon)
K <- qpI2K(I)
Sigma <- qpDscale(solve(K)) # true covariance matrix

X <- qpSampleMvnorm(K, nObs)

# scaled sample covariance matrix
S <- qpDscale(cov(X))

# more efficient scaled sample covariance matrix
clqs <- qpGetCliques(I, verbose=FALSE)
S2 <- qpIPF(S, clqs)
S2 <- qpDscale(S2)

# mean squared error of S
mean((abs(Sigma-S)^2)[upper.tri(Sigma)])

# mean squared error of S2
mean((abs(Sigma-S2)^2)[upper.tri(Sigma)])
```

qpK2R *Partial correlation coefficients*

Description

Obtains partial correlation coefficients from a given concentration matrix.

Usage

```
qpK2R(K)
```

Arguments

K	concentration matrix.

Details

Partial correlation coefficients are obtained from a concentration matrix by scaling it by its diagonal and swapping the sign of the off-diagonal elements.

Value

A matrix with partial correlation coefficients.

Author(s)

R. Castelo and A. Roverato

References

Castelo, R. and Roverato, A. A robust procedure for Gaussian graphical model search from microarray data with p larger than n. *J. Mach. Learn. Res.*, 7:2621-2650, 2006.

See Also

qpI2K

Examples

```
nVar <- 50 # number of variables
maxCon <- 5  # maximum connectivity per variable

I <- qpRndGraph(n.vtx=nVar, n.bd=maxCon)
K <- qpI2K(I)

R <- qpK2R(K)

# true partial correlation coefficients of the present edges
summary(abs(R[upper.tri(R) & I]))

# true partial correlation coefficients of the missing edges
summary(abs(R[upper.tri(R) & !I]))
```

`qpNrr` *Non-rejection rate estimation*

Description

Estimates non-rejection rates for every pair of variables.

Usage

```
## S4 method for signature 'ExpressionSet':
qpNrr(data, q=1, nTests=100, alpha=0.05, pairup.i=NULL,
                                 pairup.j=NULL, verbose=TRUE, identicalQs=TRUE,
                                 R.code.only=FALSE)
## S4 method for signature 'data.frame':
qpNrr(data, q=1, nTests=100, alpha=0.05, pairup.i=NULL,
                              pairup.j=NULL, long.dim.are.variables=TRUE,
                              verbose=TRUE, identicalQs=TRUE, R.code.only=FALSE)
## S4 method for signature 'matrix':
qpNrr(data, q=1, nTests=100, alpha=0.05, pairup.i=NULL,
                          pairup.j=NULL, long.dim.are.variables=TRUE,
                          verbose=TRUE, identicalQs=TRUE, R.code.only=FALSE)
```

Arguments

`data`	data set from where to estimate the non-rejection rates. It can be an ExpressionSet object, a data frame or a matrix.
`q`	partial-correlation order to be employed.
`nTests`	number of tests to perform for each pair for variables.
`alpha`	significance level of each test.
`pairup.i`	subset of vertices to pair up with subset `pairup.j`
`pairup.j`	subset of vertices to pair up with subset `pairup.i`
`long.dim.are.variables`	logical; if TRUE it is assumed that when data are in a data frame or in a matrix, the longer dimension is the one defining the random variables (default); if FALSE, then random variables are assumed to be at the columns of the data frame or matrix.
`verbose`	show progress on the calculations.
`identicalQs`	use identical conditioning subsets for every pair of vertices (default), otherwise sample a new collection of `nTests` subsets for each pair of vertices.
`R.code.only`	logical; if FALSE then the faster C implementation is used (default); if TRUE then only R code is executed.

Details

Note that the possible values of `q` should be in the range 1 to `min(p,n-3)`, where `p` is the number of variables and `n` the number of observations. The computational cost increases linearly with `q` and quadratically in `p`. When setting `identicalQs` to `FALSE` the computational cost may increase between 2 times and one order of magnitude (depending on `p` and `q`) while asymptotically the estimation of the non-rejection rate converges to the same value.

Value

A symmetric matrix of estimated non-rejection rates with the diagonal set to `NA`. When using the arguments `pairup.i` and `pairup.j`, those cells outside the constraint pairs will get also a `NA` value.

Author(s)

R. Castelo and A. Roverato

References

Castelo, R. and Roverato, A. A robust procedure for Gaussian graphical model search from microarray data with p larger than n, *J. Mach. Learn. Res.*, 7:2621-2650, 2006.

See Also

`qpAvgNrr` `qpEdgeNrr` `qpHist` `qpGraphDensity` `qpClique`

Examples

```
nVar <- 50 # number of variables
maxCon <- 5  # maximum connectivity per variable
nObs <- 30 # number of observations to simulate

I <- qpRndGraph(n.vtx=nVar, n.bd=maxCon)
K <- qpI2K(I)

X <- qpSampleMvnorm(K, nObs)

nrr.estimates <- qpNrr(X, q=5, verbose=FALSE)

summary(nrr.estimates[upper.tri(nrr.estimates) & I])

summary(nrr.estimates[upper.tri(nrr.estimates) & !I])
```

`qpPAC` *Estimation of partial correlation coefficients*

Description

Estimates partial correlation coefficients (PACs) and their corresponding P-values in a given undirected graph, from an input data set.

Usage

```
## S4 method for signature 'ExpressionSet':
qpPAC(data, g, return.K=FALSE,
                                   verbose=TRUE, R.code.only=FALSE)
## S4 method for signature 'data.frame':
qpPAC(data, g, return.K=FALSE,
                                long.dim.are.variables=TRUE, verbose=TRUE,
                              R.code.only=FALSE)
## S4 method for signature 'matrix':
qpPAC(data, g, return.K=FALSE,
                         long.dim.are.variables=TRUE, verbose=TRUE,
                         R.code.only=FALSE)
```

Arguments

`data`	data set from where to estimate the partial correlation coefficients. It can be an ExpressionSet object, a data frame or a matrix.
`g`	either a `graphNEL` object or an incidence matrix of the given undirected graph.
`return.K`	logical; if TRUE this function also returns the concentration matrix `K`; if FALSE it does not return it (default).
`long.dim.are.variables`	logical; if TRUE it is assumed that when data are in a data frame or in a matrix, the longer dimension is the one defining the random variables (default); if FALSE, then random variables are assumed to be at the columns of the data frame or matrix.
`verbose`	show progress on the calculations.
`R.code.only`	logical; if FALSE then the faster C implementation is used (default); if TRUE then only R code is executed.

Details

The estimation of PACs requires that the sample size `n` is strictly larger than the number of variables `p`. In the context of microarray data and regulatory networks, genes play the role of variables and thus normally `p >> n`. For this reason, we can estimate PACs from the edges of a regulatory network represented by an undirected graph `G` if and only if the maximum clique size of the graph, noted `w(G)`, is strictly smaller than the sample size `n` (number of experiments in the microarray data context).

In the context of this package, the undirected graph should correspond to a qp-graph (see function `qpGraph`) we have selected by thresholding on the (average) non-rejection rate calculated from this same data set using the functions `qpNrr` or `qpAvgNrr`. If the resulting graph is sparse enough we may have a chance to meet the requirement of `w(G) < n` and the function `qpClique` can be useful to investigate this. In the context of transcriptional regulatory networks we may consider to remove edges between non-transcription factor genes which will substantially increase the sparseness of the network.

The PAC estimation is done by first obtaining a maximum likelihood estimate of the sample covariance matrix of the input data set using the `{link{qpIPF}` function and the P-values are calculated based on the estimation of the standard errors of the edges following the procedure by Roverato and Whittaker (1996).

Value

A list with two matrices, one with the estimates of the PACs and the other with their P-values.

Author(s)

R. Castelo and A. Roverato

References

Castelo, R. and Roverato, A. A robust procedure for Gaussian graphical model search from microarray data with p larger than n. *J. Mach. Learn. Res.*, 7:2621-2650, 2006.

Castelo, R. and Roverato, A. Reverse engineering molecular regulatory networks from microarray data with qp-graphs. *J. Comp. Biol.*, 16(2):213-227, 2009.

Roverato, A. and Whittaker, J. Standard errors for the parameters of graphical Gaussian models. *Stat. Comput.*, 6:297-302, 1996.

See Also

`qpGraph` `qpCliqueNumber` `qpClique` `qpGetCliques` `qpIPF`

Examples

```
nVar <- 50 # number of variables
maxCon <- 5  # maximum connectivity per variable
nObs <- 30 # number of observations to simulate

I <- qpRndGraph(n.vtx=nVar, n.bd=maxCon)
K <- qpI2K(I)

X <- qpSampleMvnorm(K, nObs)

nrr.estimates <- qpNrr(X, verbose=FALSE)

g <- qpGraph(nrr.estimates, 0.5)

pac.estimates <- qpPAC(X, g=g, verbose=FALSE)

# estimated partial correlation coefficients of the present edges
summary(abs(pac.estimates$R[upper.tri(pac.estimates$R) & I]))

# estimated partial correlation coefficients of the missing edges
summary(abs(pac.estimates$R[upper.tri(pac.estimates$R) & !I]))
```

`qpPCC` *Estimation of Pearson correlation coefficients*

Description

Estimates Pearson correlation coefficients (PCCs) and their corresponding P-values between all pairs of variables from an input data set.

Usage

```
## S4 method for signature 'ExpressionSet':
qpPCC(data)
## S4 method for signature 'data.frame':
qpPCC(data, long.dim.are.variables=TRUE)
## S4 method for signature 'matrix':
qpPCC(data, long.dim.are.variables=TRUE)
```

Arguments

`data`	data set from where to estimate the Pearson correlation coefficients. It can be an ExpressionSet object, a data frame or a matrix.
`long.dim.are.variables`	logical; if TRUE it is assumed that when data are in a data frame or in a matrix, the longer dimension is the one defining the random variables (default); if FALSE, then random variables are assumed to be at the columns of the data frame or matrix.

Details

The calculations made by this function are the same as the ones made for a single pair of variables by the function `cor.test` but for all the pairs of variables in the data set.

Value

A list with two matrices, one with the estimates of the PCCs and the other with their P-values.

Author(s)

R. Castelo and A. Roverato

See Also

`qpPAC`

Examples

```
nVar <- 50 # number of variables
maxCon <- 5 # maximum connectivity per variable
nObs <- 30 # number of observations to simulate

I <- qpRndGraph(n.vtx=nVar, n.bd=maxCon)
K <- qpI2K(I)

X <- qpSampleMvnorm(K, nObs)

pcc.estimates <- qpPCC(X)

# Pearson correlation coefficients of the present edges
summary(abs(pcc.estimates$R[upper.tri(pcc.estimates$R) & I]))

# Pearson correlation coefficients of the missing edges
summary(abs(pcc.estimates$R[upper.tri(pcc.estimates$R) & !I]))
```

qpPrecisionRecall *Calculation of precision-recall curves*

Description

Calculates the precision-recall curve (see Fawcett, 2006) for a given measure of association between all pairs of variables in a matrix.

Usage

```
qpPrecisionRecall(measurementsMatrix, refI, decreasing=TRUE, pairup.i=NULL,
                  pairup.j=NULL, recallSteps=c(seq(0,0.1,0.005),seq(0.2,1.0,0.1)
```

Arguments

`measurementsMatrix`	matrix containing the measure of association between all pairs of variables.
`refI`	incidence matrix of reference from which to calculate the precision-recall curve.
`decreasing`	logical; if TRUE then the measurements are ordered in decreasing order; if FALSE then in increasing order.
`pairup.i`	subset of vertices to pair up with subset `pairup.j`.
`pairup.j`	subset of vertices to pair up with subset `pairup.i`.
`recallSteps`	steps of the recall on which to calculate precision.

Details

The `measurementsMatrix` should be symmetric and may have also contain `NA` values which will not be taken into account. That is an alternative way to restricting the variable pairs with the parameters `pairup.i` and `pairup.j`.

Value

A matrix where rows correspond to recall steps and columns correspond, respetively, to the actual recall, the precision, the number of true positives at that recall rate and the threshold score that yields that recall rate.

Author(s)

R. Castelo and A. Roverato

References

Fawcett, T. An introduction to ROC analysis. *Pattern Recogn. Lett.*, 27:861-874, 2006.

See Also

`qpPRscoreThreshold` `qpGraph` `qpAvgNrr` `qpPCC`

Examples

```
nVar <- 50 # number of variables
maxCon <- 5  # maximum connectivity per variable
nObs <- 30 # number of observations to simulate

I <- qpRndGraph(n.vtx=nVar, n.bd=maxCon)
K <- qpI2K(I)

X <- qpSampleMvnorm(K, nObs)

nrr.estimates <- qpNrr(X, q=5, verbose=FALSE) # NRR !

pcc.estimates <- qpPCC(X) # PCC !

# calculate area under the precision-recall curve

nrr.prerec <- qpPrecisionRecall(nrr.estimates, refI=K!=0, decreasing=FALSE,
                                recallSteps=seq(0, 1, 0.1))
f <- approxfun(nrr.prerec[, c("Recall", "Precision")])
integrate(f, 0, 1)$value

pcc.prerec <- qpPrecisionRecall(abs(pcc.estimates$R), refI=K!=0,
                                recallSteps=seq(0, 1, 0.1))
f <- approxfun(pcc.prerec[, c("Recall", "Precision")])
integrate(f, 0, 1)$value
```

`qpPRscoreThreshold` *Calculation of scores thresholds attaining nominal precision or recall levels*

Description

Calculates the score threshold at a given precision or recall level from a given precision-recall curve.

Usage

```
qpPRscoreThreshold(preRecFun, level, recall.level=TRUE, max.score=9999999)
```

Arguments

`preRecFun`	precision-recall function (output from `qpPrecisionRecall`).
`level`	recall or precision level.
`recall.level`	logical; if TRUE then it is assumed that the value given in the level parameter corresponds to a desired level of recall; if FALSE then it is assumed a desired level of precision.
`max.score`	maximum score given by the method that produced the precision-recall function to an association.

Value

The score threshold at which a given level of precision or recall is attained by the given precision-recall function. For levels that do not form part of the given function their score is calculated by linear interpolation and for this reason is important to carefully specify a proper value for the `max.score` parameter.

Author(s)

R. Castelo and A. Roverato

References

Fawcett, T. An introduction to ROC analysis. *Pattern Recogn. Lett.*, 27:861-874, 2006.

See Also

`qpPrecisionRecall` `qpGraph`

Examples

```
nVar <- 50 # number of variables
maxCon <- 5  # maximum connectivity per variable
nObs <- 30 # number of observations to simulate

I <- qpRndGraph(n.vtx=nVar, n.bd=maxCon)
K <- qpI2K(I)

X <- qpSampleMvnorm(K, nObs)

nrr.estimates <- qpNrr(X, q=1, verbose=FALSE)

nrr.prerec <- qpPrecisionRecall(nrr.estimates, K!=0, decreasing=FALSE, recallSteps=seq(0,

qpPRscoreThreshold(nrr.prerec, level=0.5, recall.level=TRUE, max.score=0)

qpPRscoreThreshold(nrr.prerec, level=0.5, recall.level=FALSE, max.score=0)
```

`qpRndGraph` *Random undirected graphs with maximum connectivity degree*

Description

Builds a random undirected graph with a bounded maximum connectivity degree (boundary) on every vertex.

Usage

```
qpRndGraph(n.vtx, n.bd)
```

Arguments

`n.vtx`	number of vertices.
`n.bd`	maximum boundary for every vertex.

Details

This is a very simple function to generate random undirected graphs where we impose a maximum order of correlation between disconnected vertices when using it to sample multivariate normal data reflecting the conditional independencies encoded in this graph. Note that the maximum order of correlation between two disconnected vertices is bounded by the minimum degree of connectivity of the two vertices.

The algorithm employed is not designed to enforce a uniform probability distribution on the set of graphs with the given maximum boundary that may be sampled with positive probability.

Value

The incidence matrix of the resulting graph.

Author(s)

R. Castelo and A. Roverato

References

Castelo, R. and Roverato, A. A robust procedure for Gaussian graphical model search from microarray data with p larger than n, *J. Mach. Learn. Res.*, 7:2621-2650, 2006.

See Also

`qpSampleMvnorm`

Examples

```
nVertices <- 50 # number of vertices
maxCon <- 5  # maximum connectivity per vertex

I <- qpRndGraph(n.vtx=nVertices, n.bd=maxCon)

summary(rowSums(I))
```

`qpSampleMvnorm` *Sample multivariate normal observations*

Description

Samples independent observations from a multivariate normal distribution with a given mean vector and a given concentration matrix.

Usage

```
qpSampleMvnorm(K, N, mean = rep(0, nrow(K)))
```

Arguments

`K`	concentration matrix of the multivariate normal distribution.
`N`	number of observations to sample.
`mean`	mean vector of the multivariate normal distribution.

Details

This function requires the `mvtnorm` package. This function is designed to be used to sample multivariate normal observations from a randomly generated concentration matrix that has a zero-structure reflecting the conditional independencies from a, possibly also randomly generated, undirected graph. The function `qpI2K` can be employed to obtain such a concentration matrix.

Value

A matrix where rows correspond to observations and columns to random variables.

Author(s)

R. Castelo and A. Roverato

References

Castelo, R. and Roverato, A. A robust procedure for Gaussian graphical model search from microarray data with p larger than n. *J. Mach. Learn. Res.*, 7:2621-2650, 2006.

See Also

`qpI2K`

Examples

```
nVar <- 50 # number of variables
maxCon <- 5  # maximum connectivity per variable
nObs <- 30 # number of observations to simulate

I <- qpRndGraph(n.vtx=nVar, n.bd=maxCon)
K <- qpI2K(I)

X <- qpSampleMvnorm(K, nObs)

dim(X)
```

Index