Package ‘qpgraph’

April 5, 2014

Title Reverse engineering of molecular regulatory networks with qp-graphs

Version 1.18.9

Author R. Castelo and A. Roverato

Description q-order partial correlation graphs, or qp-graphs for short, are undirected Gaussian graphical Markov models built from 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.

Depends R (>= 3.0.0)

Imports methods, parallel, Matrix (>= 1.0), annotate, graph (>= 1.40.1), Biobase, GGBase, AnnotationDbi, mvtnorm, qtl,Rgraphviz

Suggests BiocStyle, genefilter, org.EcK12.eg.db

Enhances rlecuyer, snow, Category, GOstats

Maintainer Robert Castelo <robert.castelo@upf.edu>

License GPL (>= 2)

biocViews Microarray, GeneExpression, Transcription, Pathways,NetworkInference, GraphsAndNetworks, GeneRegulation

LazyData yes

URL http://functionalgenomics.upf.edu/qpgraph

R topics documented:

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 built from 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.

Functions

- `qpNrr` estimates non-rejection rates for every pair of variables.
- `qpAvgNrr` estimates average non-rejection rates for every pair of variables.
- `qpGenNrr` estimates generalized 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.
- `qpRndWishart` random generation for the Wishart distribution.
- `qpCov` calculates the sample covariance matrix, just as the function `cov()` but returning a `dspMatrix-class` object which efficiently stores such a dense symmetric matrix.
- `qpG2Sigma` builds a random covariance matrix from an undrected graph. The inverse of the resulting matrix contains zeroes at the missing edges of the given undirected graph.
- `qpUnifRndAssociation` builds a matrix of uniformly random association values between -1 and +1 for all pairs of variables that follow from the number of variables given as input argument.
- `qpK2ParCor` obtains the partial correlation coefficients from a given concentration matrix.
- `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.
- 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.
- qpFunctionalCoherence estimates functional coherence of a given transcriptional regulatory network using Gene Ontology annotations.
- qpTopPairs reports a top number of pairs of variables according to either an association measure and/or occurring in a given reference graph.
- qpPlotNetwork plots a network using the Rgraphviz library.

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 and, the same directory, contains a pre-print of a book chapter describing the basic functionality of the package which serves the purpose of a basic users's guide. This package is a contribution to the Bioconductor (Gentleman et al., 2004) and gR (Lauritzen, 2002) projects.

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. 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.

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

Description

The data consist of two classes of 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 depr
subset.gds680.eset	ExpressionSet object corresponding to a subset of gds680.eset defined by the transcript
filtered.regulon6.1	Data frame object containing a subset of the E. coli transcriptional network from RegulonD
subset.filtered.regulon6.1	Subset of filtered.regulon6.1 containing the transcriptional regulatory relationships in

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)
ls()
```

---

eQTLcross-class   *eQTL experimental cross model class*

Description

The expression quantitative trait loci (eQTL) experimental cross model class serves the purpose of holding all necessary information to simulate genetical genomics data from an experimental cross.

Author(s)

R. Castelo

graphParam-class *Graph parameter classes*

Description

Graph parameter classes are defined to ease the simulation of different types of graphs by using a single interface `rgraphBAM()`.

Author(s)

R. Castelo

HMgmm-class *Homogeneous mixed graphical Markov model*

Description

The `"HMgmm"` class is the class of homogeneous mixed graphical Markov models defined within the `qpgraph` package to store simulate and manipulate this type of graphical Markov models (GMMs).

An homogeneous mixed GMM is a family of multivariate conditional Gaussian distributions on mixed discrete and continuous variables sharing a set of conditional independences encoded by means of a marked graph. Further details can be found in the book of Lauritzen (1996).

Objects from the Class

Objects can be created by calls of the form `HMgmm(g, ...)` corresponding to constructor methods or `rHMgmm(n, g, ...)` corresponding to random simulation methods.

Slots

`pI`: Object of class `"integer"` storing the number of discrete random variables.

`pY`: Object of class `"integer"` storing the number of continuous random variables.

`g`: Object of class `graphBAM-class` storing the associated marked graph.

`vtype`: Object of class `"factor"` storing the type (discrete or continuous) of each random variable.

`dLevels`: Object of class `"integer"` storing the number of levels of each discrete random variable.

`a`: Object of class `"numeric"` storing the vector of additive linear effects on continuous variables connected to discrete ones.

`rho`: Object of class `"numeric"` storing the value of the marginal correlation between two continuous random variables.

`sigma`: Object of class `dspMatrix-class` storing the covariance matrix.

`mean`: Object of class `"numeric"` storing the mean vector.

`eta2`: Object of class `"numeric"` storing for each continuous variable connected to a discrete one, the fraction of variance of the continuous variable explained by the discrete one.

Methods

`HMgmm(g)` Constructor method where `g` can be either an adjacency matrix or a `graphBAM-class` object.

`rHMgmm(n, g)` Constructor simulation method that allows one to simulate homogeneous mixed GMMs where `n` is the number of GMMs to simulate and `g` can be either a markedGraphParam object, an adjacency matrix or a `graphBAM-class` object.

`names(x)` Accessor method to obtain the names of the elements in the object `x` that can be retrieved with the `$` accessor operator.

`$` Accessor operator to retrieve elements of the object in an analogous way to a `list`.

`dim(x)` Dimension of the homogeneous mixed GMM corresponding to the number of discrete and continuous random variables.

`dimnames(x)` Names of the discrete and continuous random variables in the homogeneous mixed GMM.

`show(object)` Method to display some bits of information about the input homogeneous mixed GMM specified in `object`.

`summary(object)` Method to display a sumamry of the main features of the input homogeneous mixed GMM specified in `object`.

`plot(x, ...)` Method to plot the undirected graph associated to the the input homogeneous mixed GMM specified in `x`. It uses the plotting capabilities from the `Rgraphviz` library to which further arguments specified in `...` are further passed.

Author(s)

R. Castelo

References

Lauritzen, S.L. *Graphical models*. Oxford University Press, 1996.

See Also

UGgmm

`qpAllCItests` *Tests of conditional independence*

Description

Performs a test of conditional independence for every pair of variables.

Usage

```
## S4 method for signature matrix
qpAllCItests(X, I=NULL, Q=NULL, pairup.i=NULL, pairup.j=NULL,
                              long.dim.are.variables=TRUE, exact.test=TRUE,
                              use=c("complete.obs", "em"), tol=0.01,
                         return.type=c("p.value", "statn", "all"), verbose=TRUE,
                          R.code.only=FALSE, clusterSize=1, estimateTime=FALSE,
                              nAdj2estimateTime=10)
```

Arguments

`X` data set from where to estimate the non-rejection rates. It can be an ExpressionSet object, a data frame or a matrix.

`I` indexes or names of the variables in `X` that are discrete. See details below regarding this argument.

`Q` indexes or names of the variables in `X` forming the conditioning set.

`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.

`exact.test` logical; if `FALSE` an asymptotic conditional independence test is employed with mixed (i.e., continuous and discrete) data; if `TRUE` (default) then an exact conditional independence test with mixed data is employed. See details below regarding this argument.

`use` a character string defining the way in which calculations are done in the presence of missing values. It can be either `"complete.obs"` (default) or `"em"`.

`tol` maximum tolerance controlling the convergence of the EM algorithm employed when the argument `use="em"`.

`return.type` type of value returned by this function. By default `"p.value"` indicates that a list containing a matrix of p-values from all performed conditional independence (CI) tests will be returned. If `return.type="statn"` then a list containing the matrix of the statistics and the sample sizes on each CI test, will be returned. If `return.type="all"` then all previous matrices of values will be returned within a list.

`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.

`clusterSize` size of the cluster of processors to employ if we wish to speed-up the calculations by performing them in parallel. A value of 1 (default) implies a single-processor execution. The use of a cluster of processors requires having previously loaded the packages `snow` and `rlecuyer`.

`estimateTime` logical; if `TRUE` then the time for carrying out the calculations with the given parameters is estimated by calculating for a limited number of adjacencies, specified by `nAdj2estimateTime`, and extrapolating the elapsed time; if `FALSE` (default) calculations are performed normally till they finish.

`nAdj2estimateTime` number of adjacencies to employ when estimating the time of calculations (`estimateTime=TRUE`). By default this has a default value of 10 adjacencies and larger values should provide more accurate estimates. This might be relevant when using a cluster facility.

Details

When `I` is set different to `NULL` then mixed graphical model theory is employed and, concretely, it is assumed that the data comes from an homogeneous conditional Gaussian distribution. By default, with `exact.test=TRUE`, an exact test for conditional independence is employed, otherwise an asymptotic one will be used. Full details on these features can be found in Tur, Roverato and Castelo (2014).

Value

A list with three entries called `p.value`, `statistic` and `n` corresponding to a `dspMatrix-class` symmetric matrix of p-values for the null hypothesis of coindtional independence with the diagonal set to `NA` values, an analogous matrix of the statistics of each test and of the sample sizes, respectively. These returned values, however, depend on the setting of argument `return.type` which, by default, enables only returning the matrix of p-values. If arguments `pairup.i` and `pairup.j` are employed, those cells outside the constrained pairs will get also a `NA` value.

Note, however, that when `estimateTime=TRUE`, then instead of the matrix of estimated non-rejection rates, a vector specifying the estimated number of days, hours, minutes and seconds for completion of the calculations is returned.

Author(s)

R. Castelo, A. Roverato and I. Tur

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.

Tur, I., Roverato, A. and Castelo, R. Mapping eQTL networks with mixed graphical models. *Submitted*, `http://arxiv.org/abs/1402.4547`, 2014.

See Also

`qpCItest`

Examples

```
library(mvtnorm)

nVar <- 50  ## number of variables
maxCon <- 3 ## maximum connectivity per variable
nObs <- 30  ## number of observations to simulate

set.seed(123)

A <- qpRndGraph(p=nVar, d=maxCon)
Sigma <- qpG2Sigma(A, rho=0.5)
X <- rmvnorm(nObs, sigma=as.matrix(Sigma))

alltests <- qpAllCItests(X, verbose=FALSE)

## distribution of p-values for the present edges
summary(alltests$p.value[upper.tri(alltests$p.value) & A])

## distribution of p-values for the missing edges
summary(alltests$p.value[upper.tri(alltests$p.value) & !A])
```

---

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("adjacency.matrix", "edge.list", "graphNEL", "graphAM", "graphBAM"))
```

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 adjacency 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, or a `graphBAM-class` object.

Details

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

Value

The resulting undirected graph as either an adjacency 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

```
require(mvtnorm)

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

set.seed(123)

A <- qpRndGraph(p=nVar, d=maxCon)
Sigma <- qpG2Sigma(A, rho=0.5)
X <- rmvnorm(nObs, sigma=as.matrix(Sigma))
```

```
## estimate Pearson correlations
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(X, qOrders=4, I=NULL, restrict.Q=NULL,
                                       fix.Q=NULL, nTests=100, alpha=0.05,
                                pairup.i=NULL, pairup.j=NULL, type=c("arith.mean"),
                                       verbose=TRUE, identicalQs=TRUE,
                                      exact.test=TRUE, use=c("complete.obs", "em"),
                                       tol=0.01, R.code.only=FALSE, clusterSize=1,
                                       estimateTime=FALSE, nAdj2estimateTime=10)
## S4 method for signature data.frame
qpAvgNrr(X, qOrders=4, I=NULL, restrict.Q=NULL,
                                 fix.Q=NULL, nTests=100, alpha=0.05, pairup.i=NULL,
                                    pairup.j=NULL, long.dim.are.variables=TRUE,
                                    type=c("arith.mean"), verbose=TRUE,
                                    identicalQs=TRUE, exact.test=TRUE,
                          use=c("complete.obs", "em"), tol=0.01, R.code.only=FALSE,
                           clusterSize=1, estimateTime=FALSE, nAdj2estimateTime=10)
## S4 method for signature matrix
qpAvgNrr(X, qOrders=4, I=NULL, restrict.Q=NULL, fix.Q=NULL,
                               nTests=100, alpha=0.05, pairup.i=NULL,
                               pairup.j=NULL, long.dim.are.variables=TRUE,
                               type=c("arith.mean"), verbose=TRUE,
                               identicalQs=TRUE, exact.test=TRUE,
```

```
use=c("complete.obs", "em"), tol=0.01, R.code.only=FALSE,
clusterSize=1, estimateTime=FALSE, nAdj2estimateTime=10)
```

Arguments

`X` 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.

`I` indexes or names of the variables in `X` that are discrete. When `X` is an `ExpressionSet` then `I` may contain only names of the phenotypic variables in `X`. See details below regarding this argument.

`restrict.Q` indexes or names of the variables in `X` that restrict the sample space of conditioning subsets Q.

`fix.Q` indexes or names of the variables in `X` that should be fixed within every conditioning conditioning subsets Q.

`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.

`exact.test` logical; if `FALSE` an asymptotic conditional independence test is employed with mixed (i.e., continuous and discrete) data; if `TRUE` (default) then an exact conditional independence test with mixed data is employed.

`use` a character string defining the way in which calculations are done in the presence of missing values. It can be either `"complete.obs"` (default) or `"em"`.

`tol` maximum tolerance controlling the convergence of the EM algorithm employed when the argument `use="em"`.

`R.code.only` logical; if `FALSE` then the faster C implementation is used (default); if `TRUE` then only R code is executed.

`clusterSize` size of the cluster of processors to employ if we wish to speed-up the calculations by performing them in parallel. A value of 1 (default) implies a single-processor execution. The use of a cluster of processors requires having previously loaded the packages `snow` and `rlecuyer`.

`estimateTime` logical; if `TRUE` then the time for carrying out the calculations with the given parameters is estimated by calculating for a limited number of adjacencies, specified by `nAdj2estimateTime`, and extrapolating the elapsed time; if `FALSE` (default) calculations are performed normally till they finish.

`nAdj2estimateTime`
: number of adjacencies to employ when estimating the time of calculations (`estimateTime=TRUE`). By default this has a default value of 10 adjacencies and larger values should provide more accurate estimates. This might be relevant when using a cluster facility.

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.

When `I` is set different to `NULL` then mixed graphical model theory is employed and, concretely, it is assumed that the data comes from an homogeneous conditional Gaussian distribution. In this setting further restrictions to the maximum value of `q` apply, concretely, it cannot be smaller than `p` plus the number of levels of the discrete variables involved in the marginal distributions employed by the algorithm. By default, with `exact.test=TRUE`, an exact test for conditional independence is employed, otherwise an asymptotic one will be used. Full details on these features can be found in Tur, Roverato and Castelo (2014).

Value

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

Note, however, that when `estimateTime=TRUE`, then instead of the matrix of estimated average non-rejection rates, a vector specifying the estimated number of days, hours, minutes and seconds for completion of the calculations is returned.

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.

Tur, I., Roverato, A. and Castelo, R. Mapping eQTL networks with mixed graphical models. *Submitted*, `http://arxiv.org/abs/1402.4547`, 2014.

See Also

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

Examples

```
require(mvtnorm)

nVar <- 50  ## number of variables
maxCon <- 3 ## maximum connectivity per variable
nObs <- 30  ## number of observations to simulate

set.seed(123)

A <- qpRndGraph(p=nVar, d=maxCon)
Sigma <- qpG2Sigma(A, rho=0.5)
X <- rmvnorm(nObs, sigma=as.matrix(Sigma))

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

## distribution of average non-rejection rates for the present edges
summary(avgnrr.estimates[upper.tri(avgnrr.estimates) & A])

## distribution of average non-rejection rates for the missing edges
summary(avgnrr.estimates[upper.tri(avgnrr.estimates) & !A])

## Not run:
library(snow)
library(rlecuyer)

## only for moderate and large numbers of variables the
## use of a cluster of processors speeds up the calculations

nVar <- 500
maxCon <- 3
A <- qpRndGraph(p=nVar, d=maxCon)
Sigma <- qpG2Sigma(A, rho=0.5)
X <- rmvnorm(nObs, sigma=as.matrix(Sigma))

system.time(avgnrr.estimates <- qpAvgNrr(X, q=10, verbose=TRUE))
system.time(avgnrr.estimates <- qpAvgNrr(X, q=10, verbose=TRUE, clusterSize=4))

## End(Not run)
```

---

qpBoundary *Maximum boundary size of the resulting qp-graphs*

Description

Calculates and plots the size of the largest vertex boundary as function of the non-rejection rate.

Usage

```
qpBoundary(nrrMatrix, n=NA, threshold.lim=c(0,1), breaks=5, vertexSubset=NULL,
        plot=TRUE, qpBoundaryOutput=NULL, density.digits=0, logscale.bdsize=FALSE,
          titlebd="Maximum boundary 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.
`vertexSubset`	subset of vertices for which their maximum boundary size is calculated with respect to all other vertices.
`plot`	logical; if TRUE makes a plot of the result; if FALSE it does not.
`qpBoundaryOutput`	output from a previous call to `qpBoundary`. 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.bdsize`	logical; if TRUE then the scale for the maximum boundary size is logarithmic which is useful when working with more than 1000 variables; FALSE otherwise (default).
`titlebd`	main title to be shown in the plot.
`verbose`	show progress on calculations.

Details

The maximum boundary is calculated as the largest degree among all vertices of a given qp-graph.

Value

A list with the maximum boundary size and graph density as function of threshold, the threshold on the non-rejection rate that provides a maximum boundary size strictly smaller than the sample size n and the resulting maximum boundary 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.

See Also

`qpHTF` `qpGraphDensity`

Examples

```
require(mvtnorm)

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

set.seed(123)

A <- qpRndGraph(p=nVar, d=maxCon)
Sigma <- qpG2Sigma(A, rho=0.5)
X <- rmvnorm(nObs, sigma=as.matrix(Sigma))

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

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

qpBoundary(nrr.estimates, plot=FALSE)

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

qpBoundary(nrr.estimates, plot=FALSE)
```

---

qpCItest *Conditional independence test*

Description

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

Usage

```
## S4 method for signature smlSet
qpCItest(X, i=1, j=2, Q=c(), exact.test=TRUE, use=c("complete.obs", "em"),
                               tol=0.01, R.code.only=FALSE)
## S4 method for signature ExpressionSet
qpCItest(X, i=1, j=2, Q=c(), exact.test=TRUE, use=c("complete.obs", "em"),
                                       tol=0.01, R.code.only=FALSE)
## S4 method for signature cross
qpCItest(X, i=1, j=2, Q=c(), exact.test=TRUE, use=c("complete.obs", "em"),
                              tol=0.01, R.code.only=FALSE)
## S4 method for signature data.frame
qpCItest(X, i=1, j=2, Q=c(), I=NULL, long.dim.are.variables=TRUE,
                        exact.test=TRUE, use=c("complete.obs", "em"), tol=0.01, R.code.only=FALSE)
## S4 method for signature matrix
qpCItest(X, i=1, j=2, Q=c(), I=NULL, long.dim.are.variables=TRUE,
                     exact.test=TRUE, use=c("complete.obs", "em"), tol=0.01, R.code.only=FALSE)
## S4 method for signature SsdMatrix
qpCItest(X, i=1, j=2, Q=c(), R.code.only=FALSE)
```

Arguments

- `X` — data set where the test should be performed. It can be either an `GGBase::smlSet` object, an `ExpressionSet` object, a `qtl::cross` object, a data frame, a matrix or an `SsdMatrix-class` object. In the latter case, the input matrix should correspond to a sample covariance matrix of data on which we want to test for conditional independence. The function `qpCov()` can be used to estimate such matrices.
- `i` — index or name of one of the two variables in `X` to test.
- `j` — index or name of the other variable in `X` to test.
- `Q` — indexes or names of the variables in `X` forming the conditioning set.
- `I` — indexes or names of the variables in `X` that are discrete. See details below regarding this argument.
- `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.
- `exact.test` — logical; if `FALSE` an asymptotic likelihood ratio test of conditional independence test is employed with mixed (i.e., continuous and discrete) data; if `TRUE` (default) then an exact likelihood ratio test of conditional independence with mixed data is employed. See details below regarding this argument.
- `use` — a character string defining the way in which calculations are done in the presence of missing values. It can be either `"complete.obs"` (default) or `"em"`.
- `tol` — maximum tolerance controlling the convergence of the EM algorithm employed when the argument `use="em"`.
- `R.code.only` — logical; if FALSE then the faster C implementation is used (default); if TRUE then only R code is executed.

Details

When variables in `i, j` and `Q` are continuous and `I=NULL`, this function performs a conditional independence test using a t-test for zero partial regression coefficient (Lauritzen, 1996, pg. 150). 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`.

When variables in `i, j` and `Q` are continuous and discrete (mixed data), indicated with the `I` argument when `X` is a matrix, then mixed graphical model theory (Lauritzen and Wermuth, 1989) is employed and, concretely, it is assumed that data come from an homogeneous conditional Gaussian distribution. By default, with `exact.test=TRUE`, an exact likelihood ratio test for conditional independence is performed (Lauritzen, 1996, pg. 192-194; Tur, Roverato and Castelo, 2014), otherwise an asymptotic one is used.

In this setting further restrictions to the maximum value of `q` apply, concretely, it cannot be smaller than `p` plus the number of levels of the discrete variables involved in the marginal distributions employed by the algorithm.

Value

A list with class "htest" containing the following components:

- `statistic` — in case of pure continuous data and `I=NULL`, the t-statistic for zero partial regression coefficient; when `I!=NULL`, the value `Lambda` of the likelihood ratio if `exact.test=TRUE` and `-n log Lambda` otherwise.
- `parameter` — in case of pure continuous data and `I=NULL`, the degrees of freedom for the t-statistic (`n-q-2`); when `I!=NULL`, the degrees of freedom for `-n log Lambda` of a chi-square distribution under the null hypothesis if `exact.test=FALSE` and the `(a, b)` parameters of a beta distribution under the null if `exact.test=TRUE`.
- `p.value` — the p-value for the test.
- `estimate` — in case of pure continuous data (`I=NULL`), the estimated partial regression coefficient. In case of mixed continuous and discrete data with `I!=NULL`, the estimated partial eta-squared: the fraction of variance from `i` or `j` explained by the other tested variable after excluding the variance explained by the variables in `Q`. If one of the tested variables `i` or `j` is discrete, then the partial eta-squared is calculated on the tested continuous variable. If both, `i` and `j` are continuous, then the partial eta-squared is calculated on variable `i`.
- `alternative` — a character string describing the alternative hypothesis.
- `method` — a character string indicating what type of conditional independence test was performed.
- `data.name` — a character string giving the name(s) of the random variables involved in the conditional independence test.

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.

Lauritzen, S.L. *Graphical models*. Oxford University Press, 1996.

Lauritzen, S.L and Wermuth, N. Graphical Models for associations between variables, some of which are qualitative and some quantitative. *Ann. Stat.*, 17(1):31-57, 1989.

Tur, I., Roverato, A. and Castelo, R. Mapping eQTL networks with mixed graphical models. *Submitted*, http://arxiv.org/abs/1402.4547, 2014.

See Also

`qpCov` `qpNrr` `qpEdgeNrr`

Examples

```
require(mvtnorm)

nObs <- 100 ## number of observations to simulate

## the following adjacency matrix describes an undirected graph
## where vertex 3 is conditionally independent of 4 given 1 AND 2
A <- matrix(c(FALSE,  TRUE,  TRUE,  TRUE,
              TRUE,  FALSE,  TRUE,  TRUE,
              TRUE,   TRUE, FALSE, FALSE,
              TRUE,   TRUE, FALSE, FALSE), nrow=4, ncol=4, byrow=TRUE)
Sigma <- qpG2Sigma(A, rho=0.5)

X <- rmvnorm(nObs, sigma=as.matrix(Sigma))

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)
```

---

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

```
require(mvtnorm)

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

set.seed(123)

A <- qpRndGraph(p=nVar, d=maxCon)
Sigma <- qpG2Sigma(A, rho=0.5)
X <- rmvnorm(nObs, sigma=as.matrix(Sigma))

## 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
```

`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, R.code.only)
```

Arguments

- `g` either a `graphNEL` object or an adjacency 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.
`R.code.only`	logical; if FALSE then the faster C implementation is used (default); if TRUE then only R code is executed.

Details

The calculation of the clique number of an undirected graph is one of the basic NP-complete problems (Karp, 1972) 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 (`approx.iter`) each of which the ranking is altered with increasing levels of randomness acyclically (altering 1 to p vertices and again). Larger values of `approx.iter` should provide tighter lower bounds although it has been proven that no polynomial time algorithm can approximate the maximum clique size within a factor of n^{ϵ} ($\epsilon > 0$), unless P=NP (Feige et al, 1991; Pardalos and Xue, 1994).

Value

a lower bound of 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.

Feige, U., Goldwasser, S., Lov\'asz, L., Safra, S. and Szegedy, M. Approximating the maximum clique is almost NP-Complete. *Proc. 32nd IEEE Symp. on Foundations of Computer Science*, 2-12, 1991.

Karp, R.M. Reducibility among combinatorial problems. *Complexity of computer computations*, 43:85-103, 1972.

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

```
require(graph)

nVar <- 50

set.seed(123)

g1 <- randomEGraph(V=as.character(1:nVar), p=0.3)
qpCliqueNumber(g1, verbose=FALSE)

g2 <- randomEGraph(V=as.character(1:nVar), p=0.7)
qpCliqueNumber(g2, verbose=FALSE)
```

---

qpCov — *Calculation of the sample covariance matrix*

Description

Calculates the sample covariance matrix, just as the function `cov()` but returning a `dspMatrix-class` object which efficiently stores such a dense symmetric matrix.

Usage

```
qpCov(X, corrected=TRUE)
```

Arguments

`X`	data set from where to calculate the sample covariance matrix. As the `cov()` function, it assumes the columns correspond to random variables and the rows to multivariate observations.
`corrected`	flag set to `TRUE` when calculating the sample covariance matrix (default; and set to `FALSE` when calculating the uncorrected sum of squares and deviations.

Details

This function makes the same calculation as the `cov` function but returns a sample covariance matrix stored in the space-efficient class `dspMatrix-class` and, moreover, allows one for calculating the uncorrected sum of squares and deviations which equals `(n-1) * cov()`.

Value

A sample covariance matrix stored as a `dspMatrix-class` object. See the `Matrix` package for full details on this object class.

Author(s)

R. Castelo

See Also

qpPCC

Examples

```
require(graph)
require(mvtnorm)

nVar <- 50 ## number of variables
nObs <- 10 ## number of observations to simulate

set.seed(123)

g <- randomEGraph(as.character(1:nVar), p=0.15)

Sigma <- qpG2Sigma(g, rho=0.5)
X <- rmvnorm(nObs, sigma=as.matrix(Sigma))

S <- qpCov(X)

## estimate Pearson correlation coefficients by scaling the sample covariance matrix
R <- cov2cor(as(S, "matrix"))

## get the corresponding boolean adjacency matrix
A <- as(g, "matrix") == 1

## Pearson correlation coefficients of the present edges
summary(abs(R[upper.tri(R) & A]))

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

---

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

Description

Estimates the non-rejection rate for one pair of variables.

Usage

```
## S4 method for signature smlSet
qpEdgeNrr(X, i=1, j=2, q=1, restrict.Q=NULL, fix.Q=NULL,
                                        nTests=100, alpha=0.05, exact.test=TRUE,
```

```
                                    use=c("complete.obs", "em"), tol=0.01,
                                    R.code.only=FALSE)
## S4 method for signature ExpressionSet
qpEdgeNrr(X, i=1, j=2, q=1, restrict.Q=NULL, fix.Q=NULL,
                                    nTests=100, alpha=0.05, exact.test=TRUE,
                                    use=c("complete.obs", "em"), tol=0.01,
                                    R.code.only=FALSE)
## S4 method for signature data.frame
qpEdgeNrr(X, i=1, j=2, q=1, I=NULL, restrict.Q=NULL, fix.Q=NULL,
                           nTests=100, alpha=0.05, long.dim.are.variables=TRUE,
                         exact.test=TRUE, use=c("complete.obs", "em"), tol=0.01,
                                 R.code.only=FALSE)
## S4 method for signature matrix
qpEdgeNrr(X, i=1, j=2, q=1, I=NULL, restrict.Q=NULL, fix.Q=NULL,
                           nTests=100, alpha=0.05, long.dim.are.variables=TRUE,
                         exact.test=TRUE, use=c("complete.obs", "em"), tol=0.01,
                              R.code.only=FALSE)
## S4 method for signature SsdMatrix
qpEdgeNrr(X, i=1, j=2, q=1, restrict.Q=NULL, fix.Q=NULL,
                                nTests=100, alpha=0.05, R.code.only=FALSE)
```

Arguments

- `X` data set from where the non-rejection rate should be estimated. It can be either an `smlSet` object , an `ExpressionSet` object a data frame, a matrix or an `SsdMatrix-class` object. In the latter case, the input matrix should correspond to a sample covariance matrix of data from which we want to estimate the non-rejection rate for a pair of variables. The function `qpCov()` can be used to estimate such matrices.
- `i` index or name of one of the two variables in `X` to test.
- `j` index or name of the other variable in `X` to test.
- `q` order of the conditioning subsets employed in the calculation.
- `I` indexes or names of the variables in `X` that are discrete when `X` is a matrix or a data frame.
- `restrict.Q` indexes or names of the variables in `X` that restrict the sample space of conditioning subsets Q.
- `fix.Q` indexes or names of the variables in `X` that should be fixed within every conditioning conditioning subsets Q.
- `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.

exact.test — logical; if `FALSE` an asymptotic conditional independence test is employed with mixed (i.e., continuous and discrete) data; if `TRUE` (default) then an exact conditional independence test with mixed data is employed.See details below regarding this argument.

use — a character string defining the way in which calculations are done in the presence of missing values. It can be either `"complete.obs"` (default) or `"em"`.

tol — maximum tolerance controlling the convergence of the EM algorithm employed when the argument `use="em"`.

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 conditional 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`.

When `I` is set different to `NULL` then mixed graphical model theory is employed and, concretely, it is assumed that the data comes from an homogeneous conditional Gaussian distribution. In this setting further restrictions to the maximum value of `q` apply, concretely, it cannot be smaller than `p` plus the number of levels of the discrete variables involved in the marginal distributions employed by the algorithm. By default, with `exact.test=TRUE`, an exact test for conditional independence is employed, otherwise an asymptotic one will be used. Full details on these features can be found in Tur, Roverato and Castelo (2014).

The argument `I` specifying what variables are discrete actually applies only when `X` is a matrix object since in the other cases data types are specified for each data columns or slot.

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.

Tur, I., Roverato, A. and Castelo, R. Mapping eQTL networks with mixed graphical models. *Submitted*, http://arxiv.org/abs/1402.4547, 2014.

See Also

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

Examples

```
require(mvtnorm)

nObs <- 100 ## number of observations to simulate

## the following adjacency matrix describes an undirected graph
## where vertex 3 is conditionally independent of 4 given 1 AND 2
A <- matrix(c(FALSE,  TRUE,  TRUE,  TRUE,
               TRUE,  FALSE,  TRUE,  TRUE,
               TRUE,   TRUE, FALSE, FALSE,
               TRUE,   TRUE, FALSE, FALSE), nrow=4, ncol=4, byrow=TRUE)
Sigma <- qpG2Sigma(A, rho=0.5)

X <- rmvnorm(nObs, sigma=as.matrix(Sigma))

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 specified either as an adjacency matrix with a list of transcription factor gene identifiers or as a list of transcriptional regulatory modules, whose element names determine which genes encode for transcription factor proteins.

Usage

```
## S4 method for signature lsCMatrix
qpFunctionalCoherence(object, TFgenes, geneUniverse=rownames(object),
                                 chip, minRMsize=5, removeGOterm="transcription",
                                                verbose=FALSE, clusterSize=1)
## S4 method for signature lspMatrix
qpFunctionalCoherence(object, TFgenes, geneUniverse=rownames(object),
                                 chip, minRMsize=5, removeGOterm="transcription",
                                                verbose=FALSE, clusterSize=1)
## S4 method for signature lsyMatrix
qpFunctionalCoherence(object, TFgenes, geneUniverse=rownames(object),
                                 chip, minRMsize=5, removeGOterm="transcription",
                                                verbose=FALSE, clusterSize=1)
## S4 method for signature matrix
qpFunctionalCoherence(object, TFgenes, geneUniverse=rownames(object),
                                 chip, minRMsize=5, removeGOterm="transcription",
                                            verbose=FALSE, clusterSize=1)
## S4 method for signature list
```

```
qpFunctionalCoherence(object, geneUniverse=unique(c(names(object), unlist(object, use.names=FALSE))),
                                    chip, minRMsize=5, removeGOterm="transcription",
                                           verbose=FALSE, clusterSize=1)
```

Arguments

object — object containing the transcriptional regulatory modules for which we want to estimate their functional coherence. It can be an adjacency matrix of the undirected graph representing the transcriptional regulatory network or a list of gene target sets where the name of the entry should be the transcription factor gene identifier.

TFgenes — when the input object is a matrix, it is required to provide a vector of transcription factor gene identifiers (which should match somewhere in the row and column names of the matrix.

geneUniverse — vector of all genes considered in the analysis. By default it equals the rows and column names of `object` when it is a matrix, or the set of all different gene identifiers occuring in `object` when it is a list.

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.

removeGOterm — word, or regular pattern, matching GO terms that should be excluded in the transcription factor gene GO annotations, and in the target gene if the regulatory module has only one gene, prior to the calculation of functional coherence.

verbose — logical; if TRUE the function will show progress on the calculations; if FALSE the function will remain quiet (default).

clusterSize — size of the cluster of processors to employ if we wish to speed-up the calculations by performing them in parallel. A value of 1 (default) implies a single-processor execution. The use of a cluster of processors requires having previously loaded the packages `snow` and `rlecuyer`.

Details

This function estimates the functional coherence of a transcriptional regulatory network represented by means of an undirected graph encoded by either an adjacency matrix and a vector of transcription factor genes, or a list of regulatory modules each of them defined by a transcription factor gene and its targets. 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.

When a regulatory module has only one target gene, then no functional enrichment is calculated and, instead, the GO trees, grown from the GO annotations of the transcription factor gene and its target, are directly compared.

Value

A list with the following elements: the transcriptional regulatory network as a list of regulatory modules and their targets; the previous list of regulatory modules but excluding those with no enriched GO BP terms. When the regulatory module has only one target, then instead the GO BP annotations of the target gene are included; 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

```
## example below takes about minute and a half to execute and for
## that reason it is not executed by default
## Not run:
library(GOstats)
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, AnnotationDbi::revmap(org.EcK12.egSYMBOL)))

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

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

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

## End(Not run)
```

qpG2Sigma *Random covariance matrix*

Description

Builds a positive definite matrix from an undirected graph G that can be used as a covariance matrix for a Gaussian graphical model with graph G. The inverse of the resulting matrix contains zeroes at the missing edges of the given undirected graph G.

Usage

```
qpG2Sigma(g, rho=0, matrix.completion=c("HTF", "IPF"), tol=0.001,
          verbose=FALSE, R.code.only=FALSE)
```

Arguments

`g`	undirected graph specified either as a `graphNEL` object or as an adjacency matrix.
`rho`	real number between -1/(n.var-1) and 1 corresponding to the mean marginal correlation
`matrix.completion`	algorithm to employ in the matrix completion operations employed to construct a positive definite matrix with the zero pattern specified in g
`tol`	tolerance under which the matrix completion algorithm stops.
`verbose`	show progress on the calculations.
`R.code.only`	logical; if FALSE then the faster C implementation is used in the internal call to the HTF, or IPF, algorithm (default); if TRUE then only R code is executed.

Details

The random covariance matrix is built by first generating a random matrix with the function `qpRndWishart` from a Wishart distribution whose expected value is a matrix with unit diagonal and constant off-diagonal entries equal to `rho`.

Value

A random positive definite matrix that can be used as a covariance matrix for a Gaussian graphical model with graph `G`.

Author(s)

A. Roverato

References

Castelo, R. and Roverato, A. Utilities for large Gaussian graphical model inference and simulation with the R package qpgraph, submitted.

See Also

`qpRndGraph` `qpGetCliques` `qpIPF` `qpRndWishart` `rmvnorm`

Examples

```
set.seed(123)
G <- qpRndGraph(p=5, d=2)

Sigma <- qpG2Sigma(G, rho=0.5)

round(solve(Sigma), digits=2)

as(G, "matrix")
```

`qpGenNrr` *Generalized non-rejection rate estimation*

Description

Estimates generalized non-rejection rates for every pair of variables from two or more data sets.

Usage

```
## S4 method for signature ExpressionSet
qpGenNrr(X, datasetIdx=1, qOrders=NULL, I=NULL, restrict.Q=NULL,
                           fix.Q=NULL, return.all=FALSE, nTests=100, alpha=0.05,
                         pairup.i=NULL, pairup.j=NULL, verbose=TRUE, identicalQs=TRUE,
                          exact.test=TRUE, use=c("complete.obs", "em"), tol=0.01,
                           R.code.only=FALSE, clusterSize=1, estimateTime=FALSE,
                                nAdj2estimateTime=10)
## S4 method for signature data.frame
qpGenNrr(X, datasetIdx=1, qOrders=NULL, I=NULL, restrict.Q=NULL,
                           fix.Q=NULL, return.all=FALSE, nTests=100, alpha=0.05,
                        pairup.i=NULL, pairup.j=NULL, long.dim.are.variables=TRUE,
                              verbose=TRUE, identicalQs=TRUE, exact.test=TRUE,
                         use=c("complete.obs", "em"), tol=0.01, R.code.only=FALSE,
                         clusterSize=1, estimateTime=FALSE, nAdj2estimateTime=10)
## S4 method for signature matrix
qpGenNrr(X, datasetIdx=1, qOrders=NULL, I=NULL, restrict.Q=NULL,
                           fix.Q=NULL, return.all=FALSE, nTests=100, alpha=0.05,
                        pairup.i=NULL, pairup.j=NULL, long.dim.are.variables=TRUE,
                            verbose=TRUE, identicalQs=TRUE, exact.test=TRUE,
                         use=c("complete.obs", "em"), tol=0.01, R.code.only=FALSE,
                         clusterSize=1, estimateTime=FALSE, nAdj2estimateTime=10)
```

Arguments

`X`	data set from where to estimate the average non-rejection rates. It can be an ExpressionSet object, a data frame or a matrix.
`datasetIdx`	either a single number, or a character string, indicating the column in the phenotypic data of the `ExpressionSet` object, or in the input matrix or data frame, containing the indexes to the data sets. Alternatively, it can be a vector of these indexes with as many positions as samples.
`qOrders`	either a NULL value (default) indicating that a default guess on the q-order will be employed for each data set or a vector of particular orders with one for each data set. The default guess corresponds to the floor of the median value among the valid q orders of the data set.
`I`	indexes or names of the variables in `X` that are discrete. When `X` is an `ExpressionSet` then `I` may contain only names of the phenotypic variables in `X`. See details below regarding this argument.
`restrict.Q`	indexes or names of the variables in `X` that restrict the sample space of conditioning subsets Q.
`fix.Q`	indexes or names of the variables in `X` that should be fixed within every conditioning conditioning subsets Q.
`return.all`	logical; if TRUE all intervining non-rejection rates will be return in a matrix per dataset within a list; FALSE (default) if only generalized non-rejection rates should be returned.
`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.

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.

exact.test — logical; if FALSE an asymptotic conditional independence test is employed with mixed (i.e., continuous and discrete) data; if TRUE (default) then an exact conditional independence test with mixed data is employed.

use — a character string defining the way in which calculations are done in the presence of missing values. It can be either "complete.obs" (default) or "em".

tol — maximum tolerance controlling the convergence of the EM algorithm employed when the argument use="em".

R.code.only — logical; if FALSE then the faster C implementation is used (default); if TRUE then only R code is executed.

clusterSize — size of the cluster of processors to employ if we wish to speed-up the calculations by performing them in parallel. A value of 1 (default) implies a single-processor execution. The use of a cluster of processors requires having previously loaded the packages snow and rlecuyer.

estimateTime — logical; if TRUE then the time for carrying out the calculations with the given parameters is estimated by calculating for a limited number of adjacencies, specified by nAdj2estimateTime, and extrapolating the elapsed time; if FALSE (default) calculations are performed normally till they finish.

nAdj2estimateTime — number of adjacencies to employ when estimating the time of calculations (estimateTime=TRUE). By default this has a default value of 10 adjacencies and larger values should provide more accurate estimates. This might be relevant when using a cluster facility.

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 for the corresponding data set. 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.

When I is set different to NULL then mixed graphical model theory is employed and, concretely, it is assumed that the data comes from an homogeneous conditional Gaussian distribution. In this setting further restrictions to the maximum value of q apply, concretely, it cannot be smaller than p plus the number of levels of the discrete variables involved in the marginal distributions employed by the algorithm. By default, with `exact.test=TRUE`, an exact test for conditional independence is employed, otherwise an asymptotic one will be used. Full details on these features can be found in Tur, Roverato and Castelo (2014).

Value

A list containing the following two or more entries: a first one with name `genNrr` with a `dspMatrix-class` symmetric matrix of estimated generalized non-rejection rates with the diagonal set to `NA` values. When using the arguments `pairup.i` and `pairup.j`, those cells outside the constraint pairs will get also a `NA` value; a second one with name `qOrders` with the q-orders employed in the calculation for each data set; if `return.all=TRUE` then there will be one additional entry for each data set containing the matrix of the non-rejection rates estimated from that data set with the corresponding q-order, using the indexing value of the data set as entry name.

Note, however, that when `estimateTime=TRUE`, then instead of the list with matrices of estimated (generalized) non-rejection rates, a vector specifying the estimated number of days, hours, minutes and seconds for completion of the calculations is returned.

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.

Tur, I., Roverato, A. and Castelo, R. Mapping eQTL networks with mixed graphical models. *Submitted*, `http://arxiv.org/abs/1402.4547`, 2014.

See Also

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

Examples

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

set.seed(123)

## simulate two independent Gaussian graphical models determined
## by two undirected d-regular graphs
model1 <- rUGgmm(dRegularGraphParam(p=nVar, d=maxCon), rho=0.5)
model2 <- rUGgmm(dRegularGraphParam(p=nVar, d=maxCon), rho=0.5)

## simulate two independent data sets from the previous graphical models
X1 <- rmvnorm(nObs, model1)
dim(X1)
X2 <- rmvnorm(nObs, model2)
dim(X2)
```

```
## estimate generalized non-rejection rates from the joint data
nrr.estimates <- qpGenNrr(rbind(X1, X2),
                          datasetIdx=rep(1:2, each=nObs),
                          qOrders=c("1"=5, "2"=5),
                          long.dim.are.variables=FALSE, verbose=FALSE)

## create adjacency matrices from the undirected graphs
## determining the two Gaussian graphical models
A1 <- as(model1$g, "matrix") == 1
A2 <- as(model2$g, "matrix") == 1

## distribution of generalized non-rejection rates for the common present edges
summary(nrr.estimates$genNrr[upper.tri(nrr.estimates$genNrr) & A1 & A2])

## distribution of generalized non-rejection rates for the present edges specific to A1
summary(nrr.estimates$genNrr[upper.tri(nrr.estimates$genNrr) & A1 & !A2])

## distribution of generalized non-rejection rates for the present edges specific to A2
summary(nrr.estimates$genNrr[upper.tri(nrr.estimates$genNrr) & !A1 & A2])

## distribution of generalized non-rejection rates for the common missing edges
summary(nrr.estimates$genNrr[upper.tri(nrr.estimates$genNrr) & !A1 & !A2])

## compare with the average non-rejection rate on the pooled data set
avgnrr.estimates <- qpNrr(rbind(X1, X2), q=5, long.dim.are.variables=FALSE, verbose=FALSE)

## distribution of average non-rejection rates for the common present edges
summary(avgnrr.estimates[upper.tri(avgnrr.estimates) & A1 & A2])

## distribution of average non-rejection rates for the present edges specific to A1
summary(avgnrr.estimates[upper.tri(avgnrr.estimates) & A1 & !A2])

## distribution of average non-rejection rates for the present edges specific to A2
summary(avgnrr.estimates[upper.tri(avgnrr.estimates) & !A1 & A2])

## distribution of average non-rejection rates for the common missing edges
summary(avgnrr.estimates[upper.tri(avgnrr.estimates) & !A1 & !A2])
```

`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 adjacency 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-hard problem which means that its computational cost is bounded by an exponential running time (Garey and Johnson, 1979). 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.

Garey, M.R. and Johnson D.S. *Computers and intractability: a guide to the theory of NP-completeness*. W.H. Freeman, San Francisco, 1979.

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)

See Also

qpCliqueNumber qpIPF

Examples

```
require(graph)

set.seed(123)
nVar <- 50
g1 <- randomEGraph(V=as.character(1:nVar), p=0.3)
clqs1 <- qpGetCliques(g1, verbose=FALSE)

length(clqs1)
```

```
summary(sapply(clqs1, length))

g2 <- randomEGraph(V=as.character(1:nVar), p=0.7)
clqs2 <- qpGetCliques(g2, verbose=FALSE)

length(clqs2)

clqs2 <- qpGetCliques(g2, verbose=FALSE)

summary(sapply(clqs2, length))
```

---

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("adjacency.matrix", "edge.list", "graphNEL", "graphAM", "graphBAM"))
```

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 adjacency 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, or a `graphBAM-class` object.

Details

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

Value

The resulting qp-graph as either an adjacency 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

```
require(mvtnorm)

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

set.seed(123)

A <- qpRndGraph(p=nVar, d=maxCon)
Sigma <- qpG2Sigma(A, rho=0.5)
X <- rmvnorm(nObs, sigma=as.matrix(Sigma))

## estimate non-rejection rates
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)
```

`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

```
require(mvtnorm)

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

set.seed(123)

A <- qpRndGraph(p=nVar, d=maxCon)
Sigma <- qpG2Sigma(A, rho=0.5)
X <- rmvnorm(nObs, sigma=as.matrix(Sigma))

## the higher the q the sparser the qp-graph

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
```

---

qpHist *Histograms of non-rejection rates*

Description

Plots the distribution of non-rejection rates.

Usage

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

Arguments

nrrMatrix	matrix of non-rejection rates.
A	adjacency matrix of an undirected graph whose present and missing edges will be employed to show separately the distribution of non-rejection rates.
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

```
require(mvtnorm)

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

A <- qpRndGraph(p=nVar, d=maxCon)
Sigma <- qpG2Sigma(A, rho=0.5)
X <- rmvnorm(nObs, sigma=as.matrix(Sigma))

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

qpHist(nrr.estimates, A)
```

`qpHTF` *Hastie Tibshirani Friedman algorithm*

Description

Performs maximum likelihood estimation of a covariance matrix given the independence constraints from an input undirected graph.

Usage

```
qpHTF(S, g, tol = 0.001, verbose = FALSE, R.code.only = FALSE)
```

Arguments

`S`	input matrix, in the context of this package, the sample covariance matrix.
`g`	input undirected graph.
`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

This is an alternative to the Iterative Proportional Fitting (IPF) algorithm (see, Whittaker, 1990, pp. 182-185 and `qpIPF`) which also adjusts the input matrix to the independence constraints in the input undirected graph. However, differently to the IPF, it works by going through each of the vertices fitting the marginal distribution over the corresponding vertex boundary. 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. This algorithm is described by Hastie, Tibshirani and Friedman (2009, pg. 634), hence we name it here HTF, and it has the advantage over the IPF that it does not require the list of maximal cliques of the graph which may be exponentially large. In contrast, it requires that the maximum boundary size of the graph is below the number of samples where the input sample covariance matrix `S` was estimated. For the purpose of exploring qp-graphs that meet such a requirement, one can use the function `qpBoundary`.

Value

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

Note

Thanks to Giovanni Marchetti for bringing us our attention to this algorithm and sharing an early version of its implementation on the R package `ggm`.

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.

Hastie, T., Tibshirani, R. and Friedman, J.H. *The Elements of Statistical Learning*, Springer, 2009.

Whittaker, J. *Graphical Models in Applied Multivariate Statistics.* Wiley, 1990.

See Also

`qpBoundary` `qpIPF` `qpPAC`

Examples

```
require(graph)
require(mvtnorm)

nVar <- 50  ## number of variables
nObs <- 100 ## number of observations to simulate

set.seed(123)

g <- randomEGraph(as.character(1:nVar), p=0.15)

Sigma <- qpG2Sigma(g, rho=0.5)
X <- rmvnorm(nObs, sigma=as.matrix(Sigma))

## MLE of the sample covariance matrix
S <- cov(X)

## more efficient MLE of the sample covariance matrix using HTF
S_htf <- qpHTF(S, g)

## get the adjacency matrix and put the diagonal to one
A <- as(g, "matrix")
diag(A) <- 1

## entries in S and S_htf for present edges in g should coincide
max(abs(S_htf[A==1] - S[A==1]))

## entries in the inverse of S_htf for missing edges in g should be zero
max(solve(S_htf)[A==0])
```

---

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 covariance matrix given the independence constraints from an 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

```
require(graph)
require(mvtnorm)

nVar <- 50  ## number of variables
nObs <- 100 ## number of observations to simulate

set.seed(123)
```

```
g <- randomEGraph(as.character(1:nVar), p=0.15)

Sigma <- qpG2Sigma(g, rho=0.5)
X <- rmvnorm(nObs, sigma=as.matrix(Sigma))

## MLE of the sample covariance matrix
S <- cov(X)

## more efficient MLE of the sample covariance matrix using IPF
clqs <- qpGetCliques(g, verbose=FALSE)
S_ipf <- qpIPF(S, clqs)

## get the adjacency matrix and put the diagonal to one
A <- as(g, "matrix")
diag(A) <- 1

## entries in S and S_ipf for present edges in g should coincide
max(abs(S_ipf[A==1] - S[A==1]))

## entries in the inverse of S_ipf for missing edges in g should be zero
max(solve(S_ipf)[A==0])
```

---

qpK2ParCor *Partial correlation coefficients*

Description

Obtains partial correlation coefficients from a given concentration matrix.

Usage

```
qpK2ParCor(K)
```

Arguments

K	positive definite matrix, typically a concentration matrix.

Details

This function applies `cov2cor` to the given concentration matrix and then changes the sign of the off-diagonal entries in order to obtain a partial correlation matrix.

Value

A partial correlation matrix.

Author(s)

R. Castelo and A. Roverato

References

Lauritzen, S.L. *Graphical models*. Oxford University Press, 1996.

See Also

qpG2Sigma

Examples

```
require(graph)

n.var <- 5 # number of variables
set.seed(123)
g <- randomEGraph(as.character(1:n.var), p=0.15)

Sigma <- qpG2Sigma(g, rho=0.5)
K <- solve(Sigma)

round(qpK2ParCor(K), digits=2)

as(g, "matrix")
```

---

qpNrr *Non-rejection rate estimation*

Description

Estimates non-rejection rates for every pair of variables.

Usage

```
## S4 method for signature ExpressionSet
qpNrr(X, q=1, restrict.Q=NULL, fix.Q=NULL, nTests=100,
                                  alpha=0.05, pairup.i=NULL, pairup.j=NULL,
                                 verbose=TRUE, identicalQs=TRUE, exact.test=TRUE,
                        use=c("complete.obs", "em"), tol=0.01, R.code.only=FALSE,
                         clusterSize=1, estimateTime=FALSE, nAdj2estimateTime=10)
## S4 method for signature cross
qpNrr(X, q=1, restrict.Q=NULL, fix.Q=NULL, nTests=100,
                          alpha=0.05, pairup.i=NULL, pairup.j=NULL, verbose=TRUE,
                 identicalQs=TRUE, exact.test=TRUE, use=c("complete.obs", "em"),
                 tol=0.01, R.code.only=FALSE, clusterSize=1, estimateTime=FALSE,
                           nAdj2estimateTime=10)
## S4 method for signature data.frame
qpNrr(X, q=1, I=NULL, restrict.Q=NULL, fix.Q=NULL, nTests=100,
                               alpha=0.05, pairup.i=NULL, pairup.j=NULL,
                               long.dim.are.variables=TRUE, verbose=TRUE,
                    identicalQs=TRUE, exact.test=TRUE, use=c("complete.obs", "em"),
```

```
                              tol=0.01, R.code.only=FALSE, clusterSize=1,
                              estimateTime=FALSE, nAdj2estimateTime=10)
## S4 method for signature matrix
qpNrr(X, q=1, I=NULL, restrict.Q=NULL, fix.Q=NULL, nTests=100,
                          alpha=0.05, pairup.i=NULL, pairup.j=NULL,
                    long.dim.are.variables=TRUE, verbose=TRUE, identicalQs=TRUE,
                         exact.test=TRUE, use=c("complete.obs", "em"), tol=0.01,
                          R.code.only=FALSE, clusterSize=1, estimateTime=FALSE,
                          nAdj2estimateTime=10)
```

Arguments

`X` data set from where to estimate the non-rejection rates. It can be an `ExpressionSet` object, a `qtl/cross` object, a `data.frame` object or a `matrix` object.

`q` partial-correlation order to be employed.

`I` indexes or names of the variables in `X` that are discrete. See details below regarding this argument.

`restrict.Q` indexes or names of the variables in `X` that restrict the sample space of conditioning subsets Q.

`fix.Q` indexes or names of the variables in `X` that should be fixed within every conditioning conditioning subsets Q.

`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.

`exact.test` logical; if `FALSE` an asymptotic conditional independence test is employed with mixed (i.e., continuous and discrete) data; if `TRUE` (default) then an exact conditional independence test with mixed data is employed. See details below regarding this argument.

`use` a character string defining the way in which calculations are done in the presence of missing values. It can be either `"complete.obs"` (default) or `"em"`.

`tol` maximum tolerance controlling the convergence of the EM algorithm employed when the argument `use="em"`.

`R.code.only` logical; if `FALSE` then the faster C implementation is used (default); if `TRUE` then only R code is executed.

clusterSize
: size of the cluster of processors to employ if we wish to speed-up the calculations by performing them in parallel. A value of 1 (default) implies a single-processor execution. The use of a cluster of processors requires having previously loaded the packages `snow` and `rlecuyer`.

estimateTime
: logical; if `TRUE` then the time for carrying out the calculations with the given parameters is estimated by calculating for a limited number of adjacencies, specified by `nAdj2estimateTime`, and extrapolating the elapsed time; if `FALSE` (default) calculations are performed normally till they finish.

nAdj2estimateTime
: number of adjacencies to employ when estimating the time of calculations (`estimateTime=TRUE`). By default this has a default value of 10 adjacencies and larger values should provide more accurate estimates. This might be relevant when using a cluster facility.

Details

Note that for pure continuous data 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. Full details on the calculation of the non-rejection rate can be found in Castelo and Roverato (2006).

When `I` is set different to `NULL` then mixed graphical model theory is employed and, concretely, it is assumed that the data comes from an homogeneous conditional Gaussian distribution. In this setting further restrictions to the maximum value of `q` apply, concretely, it cannot be smaller than `p` plus the number of levels of the discrete variables involved in the marginal distributions employed by the algorithm. By default, with `exact.test=TRUE`, an exact test for conditional independence is employed, otherwise an asymptotic one will be used. Full details on these features can be found in Tur, Roverato and Castelo (2014).

The argument `I` specifying what variables are discrete actually applies only when `X` is a matrix object since in the other cases data types are specified for each data columns or slot.

In the case that `X` is a `qtl/cross` object, the default `NULL` values in arguments `pairup.i` and `pairup.j` actually imply pairing all markers and phenotypes with numerical phenotypes only (including integer phenotypes). Likewise, the default argument `restrict.Q=NULL` implies setting `restrict.Q` to all numeric phenotypes. Setting these arguments to values other than `NULL` allows the user to use those particular values being set.

Value

A `dspMatrix-class` symmetric matrix of estimated non-rejection rates with the diagonal set to `NA` values. If arguments `pairup.i` and `pairup.j` are employed, those cells outside the constrained pairs will get also a `NA` value.

Note, however, that when `estimateTime=TRUE`, then instead of the matrix of estimated non-rejection rates, a vector specifying the estimated number of days, hours, minutes and seconds for completion of the calculations is returned.

Author(s)

R. Castelo, A. Roverato and I. Tur

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.

Tur, I., Roverato, A. and Castelo, R. Mapping eQTL networks with mixed graphical models. *Submitted*, http://arxiv.org/abs/1402.4547, 2014.

See Also

qpAvgNrr qpEdgeNrr qpHist qpGraphDensity qpClique

Examples

```
nVar <- 50  ## number of variables
maxCon <- 3 ## maximum connectivity per variable
nObs <- 30  ## number of observations to simulate

set.seed(123)

## simulate an undirected Gaussian graphical model
## determined by some random undirected d-regular graph
model <- rUGgmm(dRegularGraphParam(p=nVar, d=maxCon), rho=0.5)
model

## simulate data from this model
X <- rmvnorm(nObs, model)
dim(X)

## estimate non-rejection rates with q=3
nrr.estimates <- qpNrr(X, q=3, verbose=FALSE)

## create an adjacency matrix of the undirected graph
## determining the undirected Gaussian graphical model
A <- as(model$g, "matrix") == 1

## distribution of non-rejection rates for the present edges
summary(nrr.estimates[upper.tri(nrr.estimates) & A])

## distribution of non-rejection rates for the missing edges
summary(nrr.estimates[upper.tri(nrr.estimates) & !A])

## Not run:
## using R code only this would take much more time
qpNrr(X, q=3, R.code.only=TRUE, estimateTime=TRUE)

## only for moderate and large numbers of variables the
## use of a cluster of processors speeds up the calculations
```

```
library(snow)
library(rlecuyer)

nVar <- 500
maxCon <- 3
model <- rUGgmm(dRegularGraphParam(p=nVar, d=maxCon), rho=0.5)
X <- rmvnorm(nObs, model)

system.time(nrr.estimates <- qpNrr(X, q=10, verbose=TRUE))
system.time(nrr.estimates <- qpNrr(X, q=10, verbose=TRUE, clusterSize=4))

## End(Not run)
```

---

qpPAC *Estimation of partial correlation coefficients*

Description

Estimates partial correlation coefficients (PACs) for a Gaussian graphical model with undirected graph G and their corresponding P-values for the hypothesis of zero partial correlations.

Usage

```
## S4 method for signature ExpressionSet
qpPAC(X, g, return.K=FALSE, tol=0.001,
                               matrix.completion=c("HTF", "IPF"), verbose=TRUE,
                                R.code.only=FALSE)
## S4 method for signature data.frame
qpPAC(X, g, return.K=FALSE, long.dim.are.variables=TRUE,
                             tol=0.001, matrix.completion=c("HTF", "IPF"),
                             verbose=TRUE, R.code.only=FALSE)
## S4 method for signature matrix
qpPAC(X, g, return.K=FALSE, long.dim.are.variables=TRUE,
                         tol=0.001, matrix.completion=c("HTF", "IPF"),
                         verbose=TRUE, R.code.only=FALSE)
```

Arguments

`X` 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 adjacency 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 `X` is a data frame or 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.

`tol`	maximum tolerance in the application of the IPF algorithm.
`matrix.completion`	algorithm to employ in the matrix completion operations employed to construct a positive definite matrix with the zero pattern specified in g
`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

In the context of maximum likelihood estimation (MLE) of PACs it is a necessary condition for the existence of MLEs that the sample size `n` is larger than the clique number `w(G)` of the graph `G`.

The PAC estimation is done by first obtaining a MLE of the covariance matrix using the `qpIPF` function and the P-values are calculated based on the estimation of the standard errors (see 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

```
require(mvtnorm)

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

set.seed(123)

A <- qpRndGraph(p=nVar, d=maxCon)
Sigma <- qpG2Sigma(A, rho=0.5)
```

```
X <- rmvnorm(nObs, sigma=as.matrix(Sigma))

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

g <- qpGraph(nrr.estimates, 0.5)

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

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

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

---

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

Arguments

X	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 X is a data frame or 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

```
require(graph)
require(mvtnorm)

nVar <- 50 ## number of variables
nObs <- 10 ## number of observations to simulate

set.seed(123)

g <- randomEGraph(as.character(1:nVar), p=0.15)

Sigma <- qpG2Sigma(g, rho=0.5)
X <- rmvnorm(nObs, sigma=as.matrix(Sigma))

pcc.estimates <- qpPCC(X)

## get the corresponding boolean adjacency matrix
A <- as(g, "matrix") == 1

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

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

---

qpPlotMap *Plots a map of associated pairs*

Description

Plots a map of associated pairs defined by adjusted p-values

Usage

```
qpPlotMap(p.valueMatrix, markerPos, genePos, chrLen,
          p.value=0.05, adjust.method="holm",
          xlab="Ordered Markers", ylab="Ordered Genes",
          main="", ...)
```

Arguments

`p.valueMatrix`	squared symmetric matrix with raw p-values for all pairs.
`markerPos`	two-column matrix containing chromosome and position of each genetic marker.
`genePos`	two-column matrix containing chromosome and position of each gene.
`chrLen`	named vector with chromosome lengths. Vector names should correspond to chromosome names, which are displayed in the axes of the plot. This vector should be ordered following the same convention for chromosomes in arguments `markerPos` and `genePos`.
`p.value`	adjusted p-value cutoff.
`adjust.method`	method employed to adjust the raw p-values. It is passed in a call to `p.adjust()` in its `method` argument.
`xlab`	label for the x-axis.
`ylab`	label for the y-axis.
`main`	main title of the plot, set to the empty string by default.
`...`	further arguments passed to the `plot()` function.

Details

This function plots a map of present associations, typically between genetic markers and gene expression profiles (i.e., eQTL associations), according to the chromosomal locations of both the genetic markers and the genes. The input argument `p.valueMatrix` should contain the raw p-values of these associations. Present associations are selected by a cutoff given in the `p.value` argument applied to the adjusted p-values.

The input raw p-values can be obtained with the function `qpAllCItests`.

Value

The selected present associations are invisibly returned.

Author(s)

R. Castelo

See Also

`qpAllCItests`

Examples

```
## generate uniformly random p-values for synthetic associations
## between m genetic markers and g genes into a symmetric matrix
m <- 100
g <- 100
p <- m + g
markerids <- paste0("m", 1:m)
geneids <- paste0("g", 1:g)
rndpvalues <- matrix(0, nrow=p, ncol=p,
                     dimnames=list(c(markerids, geneids), c(markerids, geneids)))
rndpvalues[1:m,(m+1):p] <- runif(m*g)

## put significant cis associations
rndpvalues[cbind(1:m, (m+1):p)] <- rnorm(m, mean=1e-4, sd=1e-2)^2

## put one hotspot locus with significant, but somehat weaker, trans associations
hotspotmarker <- sample(1:m, size=1)
rndpvalues[cbind(hotspotmarker, (m+1):p)] <- rnorm(g, mean=1e-2, sd=1e-2)^2

## make matrix symmetric
rndpvalues <- rndpvalues + t(rndpvalues)
stopifnot(isSymmetric(rndpvalues))
rndpvalues[1:m, 1:m] <- rndpvalues[(m+1):p,(m+1):p] <- NA

## create chromosomal map
chrlen <- c("chr1"=1000)
posmarkers <- matrix(c(rep(1, m), seq(1, chrlen, length.out=m)), nrow=m)
posgenes <- matrix(c(rep(1, g), seq(1, chrlen, length.out=g)), nrow=g)
rownames(posmarkers) <- paste0("m", 1:m)
rownames(posgenes) <- paste0("g", 1:g)

qpPlotMap(rndpvalues, posmarkers, posgenes, chrlen, cex=3)
```

`qpPlotNetwork` *Plots a graph*

Description

Plots a graph using the `Rgraphviz` library

Usage

```
qpPlotNetwork(g, vertexSubset=graph::nodes(g), boundary=FALSE,
               minimumSizeConnComp=2, pairup.i=NULL, pairup.j=NULL,
          highlight=NULL, annotation=NULL, layout=c("twopi", "dot", "neato", "circo", "fdp"))
```

Arguments

g	graph to plot provided as a graphNEL-class object.
vertexSubset	subset of vertices that define the induced subgraph to be plotted.
boundary	flag set to TRUE when we wish that the subset specified in vertexSubset also includes the vertices connected to them; FALSE otherwise.
minimumSizeConnComp	minimum size of the connected components to be plotted.
pairup.i	subset of vertices to pair up with subset pairup.j.
pairup.j	subset of vertices to pair up with subset pairup.i.
highlight	subset of vertices to highlight by setting the color font to red.
annotation	name of an annotation package to transform gene identifiers into gene symbols when vertices correspond to genes.
layout	layout argument for the Rgraphviz library that plots the network. Possible values are twopi (default), dot, neato, circo, fdp.

Details

This function acts as a wrapper for the functionality provided by the Rgraphviz package to plot graphs in R. It should help to plot networks obtained with methods from theqpgraph package.

Value

The plotted graph is invisibly returned as a graphNEL-class object.

Author(s)

R. Castelo

See Also

qpGraph qpAnyGraph

Examples

```
require(Rgraphviz)

rndassociations <- qpUnifRndAssociation(10)
g <- qpAnyGraph(abs(rndassociations), threshold=0.7, remove="below", return.type="graphNEL")
qpPlotNetwork(g)
```

`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, refGraph, decreasing=TRUE, pairup.i=NULL,
                  pairup.j=NULL, recallSteps=seq(0, 1, by=0.1))
```

Arguments

`measurementsMatrix`
: matrix containing the measure of association between all pairs of variables.

`refGraph`
: a reference graph from which to calculate the precision-recall curve provided either as an adjacency matrix, a two-column matrix of edges, a `graphNEL-class` object or a `graphAM-class` object.

`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

```
require(mvtnorm)

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

set.seed(123)

A <- qpRndGraph(p=nVar, d=maxCon)
Sigma <- qpG2Sigma(A, rho=0.5)
X <- rmvnorm(nObs, sigma=as.matrix(Sigma))

## estimate non-rejection rates
nrr.estimates <- qpNrr(X, q=5, verbose=FALSE)

## estimate Pearson correlation coefficients
pcc.estimates <- qpPCC(X)

## calculate area under the precision-recall curve
## for both sets of estimated values of association
nrr.prerec <- qpPrecisionRecall(nrr.estimates, refGraph=A, 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$R), refGraph=A,
                                recallSteps=seq(0, 1, 0.1))
f <- approxfun(pcc.prerec[, c("Recall", "Precision")])
integrate(f, 0, 1)$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

```
require(mvtnorm)

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

set.seed(123)

A <- qpRndGraph(p=nVar, d=maxCon)
Sigma <- qpG2Sigma(A, rho=0.5)
X <- rmvnorm(nObs, sigma=as.matrix(Sigma))

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

nrr.prerec <- qpPrecisionRecall(nrr.estimates, A, decreasing=FALSE,
                                recallSteps=seq(0, 1, by=0.1))

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 *Undirected random d-regular graphs*

Description

Samples an undirected d-regular graph approximately uniformly at random.

Usage

```
qpRndGraph(p=6, d=2, labels=1:p, exclude=NULL, verbose=FALSE,
           return.type=c("adjacency.matrix", "edge.list", "graphBAM", "graphNEL"),
             R.code.only=FALSE)
```

Arguments

`p`	number of vertices.
`d`	degree of every vertex.
`labels`	vertex labels.
`exclude`	vector of vertices inducing edges that should be excluded from the sampled d-regular graph.
`verbose`	show progress on the calculations.
`return.type`	class of object to be returned by the function
`R.code.only`	logical; if `FALSE` then the faster C implementation is used (default); if `TRUE` then only R code is executed.

Details

This function implements the algorithm from Steger and Wormald (1999) for sampling undirected d-regular graphs from a probability distribution of all d-regular graphs on p vertices which is approximately uniform. More concretely, for all vertex degree values d that grow as a small power of p, all d-regular graphs on p vertices will have in the limit the same probability as p grows large. Steger and Wormald (1999, pg. 396) believe that for d » sqrt(p) the resulting probability distribution will no longer be approximately uniform.

This function is provided in order to generate a random undirected graph as input to the function `qpG2Sigma` which samples a random covariance matrix whose inverse (aka, precision matrix) has zeroes on those cells corresponding to the missing edges in the input graph. d-regular graphs are useful for working with synthetic graphical models for two reasons: one is that d-regular graph density is a linear function of d and the other is that the minimum connectivity degree of two disconnected vertices is an upper bound of their outer connectivity (see Castelo and Roverato, 2006, pg. 2646).

Value

The adjacency 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.

Steger, A. and Wormald, N.C. Generating random regular graphs quickly, *Combinatorics, Probab. and Comput.*, 8:377-396.

See Also

qpG2Sigma

Examples

```
set.seed(123)

A <- qpRndGraph(p=50, d=3)

summary(apply(A, 1, sum))
```

---

qpRndHMGM *Random homogeneous mixed graphical Markov model*

Description

Builds a random homogeneous mixed graphical Markov model. This function has been defunct, the user should use rHMgmm.

Usage

```
qpRndHMGM(nDiscrete=1, nContinuous=3, d=2, mixedIntStrength=5, rho=0.5, G=NULL)
```

Arguments

nDiscrete	number of discrete variables.
nContinuous	number of continuous variables.
d	degree of every vertex.
mixedIntStrength	strength of the mixed interactions.
rho	marginal correlation of the quadratic interactions.
G	input graph, if we don't want the function to simulate one.

Details

This function builds a random homogeneous mixed graphical model. It uses qpRndGraph to simulate a random d-regular graph and then builds a set of parameters that encode the conditional independencies encoded by the graph and the given number of discrete and continuous vertices. This is still an experimental feature and by now it generates only models where the discrete variables are marginally independent.

Value

A list with the graph and the parameters of the homogeneous mixed graphical model, ready to be used with the function qpSampleFromHMGM for sampling synthetic data using this model.

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

qpRndGraph qpSampleFromHMGM

Examples

```
## Not run:
qpRndHMGM()

## End(Not run)
```

---

qpRndWishart *Random Wishart distribution*

Description

Random generation for the (`n.var * n.var`) Wishart distribution (see Press, 1972) with matrix parameter `A=diag(delta)%*%P%*%diag(delta)` and degrees of freedom `df`.

Usage

```
qpRndWishart(delta=1, P=0, df=NULL, n.var=NULL)
```

Arguments

- `delta` — a numeric vector of `n.var` positive values. If a scalar is provided then this is extended to form a vector.
- `P` — a (`n.var * n.var`) positive definite matrix with unit diagonal. If a scalar is provided then this number is used as constant off-diagonal entry for P.
- `df` — degrees of freedom.
- `n.var` — dimension of the Wishart matrix. It is required only when both delata and P are scalar.

Details

The degrees of freedom are `df > n.var-1` and the expected value of the distribution is equal to `df * A`. The random generator is based on the algorithm of Odell and Feiveson (1966).

Value

A list of two `n.var * n.var` matrices `rW` and `meanW` where `rW` is a random value from the Wishart and `meanW` is the expected value of the distribution.

Author(s)

A. Roverato

References

Odell, P.L. and Feiveson, A.G. A numerical procedure to generate a sample covariance matrix. *J. Am. Statist. Assoc.* 61, 199-203, 1966.

Press, S.J. *Applied Multivariate Analysis: Using Bayesian and Frequentist Methods of Inference*. New York: Holt, Rinehalt and Winston, 1972.

See Also

`qpG2Sigma`

Examples

```
## Construct an adjacency matrix for a graph on 6 vertices

nVar <- 6
A <- matrix(0, nVar, nVar)
A[1,2] <- A[2,3] <- A[3,4] <- A[3,5] <- A[4,6] <- A[5,6] <- 1
A=A + t(A)
A
set.seed(123)
M <- qpRndWishart(delta=sqrt(1/nVar), P=0.5, n.var=nVar)
M
set.seed(123)
d=1:6
M <- qpRndWishart(delta=d, P=0.7, df=20)
M
```

qpSampleFromHMGM *Sample from homogeneous mixed graphical Markov models*

Description

Samples synthetic data from homogeneous mixed graphical Markov models. This function has been defunct, the user should use `rcmvnorm`.

Usage

```
qpSampleFromHMGM(n=10, hmgm=qpRndHMGM())
```

Arguments

`n`	number of observations to sample.
`hmgm`	homogeneous mixed graphical Markov model as generated by the function `qpRndHMGM`.

Details

This function samples synthetic data from a random homogeneous mixed graphical model build with the function `qpRndHMGM`. This is still an experimental feature.

Value

The sampled synthetic data.

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

`qpRndGraph` `qpSampleFromHMGM`

Examples

```
## Not run:
qpSampleFromHMGM()

## End(Not run)
```

qpTopPairs *Report pairs of variables*

Description

Report a top number of pairs of variables according to either some association measure and/or occurring in a given reference graph.

Usage

```
qpTopPairs(measurementsMatrix=NULL, refGraph=NULL, n=6L, file=NULL,
           decreasing=FALSE, pairup.i=NULL, pairup.j=NULL,
           annotation=NULL, fcOutput=NULL, fcOutput.na.rm=FALSE,
           digits=NULL)
```

Arguments

`measurementsMatrix` matrix containing the measure of association between all pairs of variables.

`refGraph` a reference graph containing the pairs that should be reported and provided either as an adjacency matrix, a `graphNEL-class` object or a `graphAM-class` object.

`n` number of pairs to report, 6 by default, use `Inf` for reporting all of them.

`file` file name to dump the pairs information as tab-separated column text.

`decreasing` logical; if TRUE then the measurements are employed to be 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`.

`annotation` name of an annotation package to transform gene identifiers into gene symbols when variables correspond to genes.

`fcOutput` output of `qpFunctionalCoherence`.

`fcOutput.na.rm` flag set to TRUE when pairs with `NA` values from `fcOutput` should not be reported; `FALSE` (default) otherwise.

`digits` number of decimal digits reported in the values of `measurementsMatrix` and functional coherence values. By default `digits=NULL`, and therefore, no rounding is performed.

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`. The same holds for `refGraph`. One of these two, should be specified.

Value

The ranking of pairs is invisibly returned.

Author(s)

R. Castelo

See Also

qpGraph qpPrecisionRecall qpFunctionalCoherence

Examples

```
qpTopPairs(matrix(runif(100), nrow=10, dimnames=list(1:10,1:10)))
```

---

qpUnifRndAssociation *Uniformly random association values*

Description

Builds a matrix of uniformly random association values between -1 and +1 for all pairs of variables that follow from the number of variables given as input argument.

Usage

```
qpUnifRndAssociation(n.var, var.names=1:n.var)
```

Arguments

n.var	number of variables.
var.names	names of the variables to use as row and column names in the resulting matrix.

Details

This function simply generates uniformly random association values with no independence pattern associated to them. For generating a random covariance matrix that reflects such a pattern use the function qpG2Sigma.

Value

A symmetric matrix of uniformly random association values between -1 and +1.

Author(s)

R. Castelo

See Also

qpG2Sigma

Examples

```
rndassociation <- qpUnifRndAssociation(100)
summary(rndassociation[upper.tri(rndassociation)])
```

---

qpUpdateCliquesRemoving

Update clique list when removing one edge

Description

Updates the set of (maximal) cliques of a given undirected graph when removing one edge.

Usage

```
qpUpdateCliquesRemoving(g, clqlst, v, w, verbose=TRUE)
```

Arguments

g	either a graphNEL object or an adjacency matrix of the given undirected graph.
clqlst	list of cliques of the graph encoded in g. this list should start on element n+1 (for n vertices) while between elements 1 to n there should be references to the cliques to which each of the 1 to n vertices belong to (i.e., the output of qpGetCliques) with parameter clqspervtx=TRUE.
v	vertex of the edge being removed.
w	vertex of the edge being removed.
verbose	show progress on calculations.

Details

To find the list of all (maximal) cliques in an undirected graph is an NP-hard problem which means that its computational cost is bounded by an exponential running time (Garey and Johnson, 1979). For this reason, this is an extremely time and memory consuming computation for large dense graphs. If we spend the time to obtain one such list of cliques and we remove one edge of the graph with this function we may be able to update the set of maximal cliques instead of having to generate it again entirely with qpGetCliques but it requires that in the first call to qpGetCliques we set clqspervtx=TRUE. It calls a C implementation of the algorithm from Stix (2004).

Value

The updated list of maximal cliques after removing one edge from the input graph. Note that because the corresponding input clique list had to be generated with the argument `clqspervtx=TRUE` in the call to `qpGetCliques`, the resulting updated list of cliques also includes in its first p entries (p=number of variables) the indices of the cliques where that particular vertex belongs to. Notice also that although this strategy might be in general more efficient than generating again the entire list of cliques, when removing one edge from the graph, the clique enumeration problem remains NP-hard (see Garey and Johnson, 1979) and therefore depending on the input graph its computation may become unfeasible.

Author(s)

R. Castelo

References

Garey, M.R. and Johnson D.S. *Computers and intractability: a guide to the theory of NP-completeness*. W.H. Freeman, San Francisco, 1979.

Stix, V. Finding all maximal cliques in dynamic graphs *Comput. Optimization and Appl.*, 27:173-186, 2004.

See Also

`qpCliqueNumber` `qpGetCliques` `qpIPF`

Examples

```
## the example below takes about 30 seconds to execute and for that reason
## it is not executed by default
## Not run:
require(graph)

set.seed(123)
nVar <- 1000
g1 <- randomEGraph(V=as.character(1:nVar), p=0.1)
g1
clqs1 <- qpGetCliques(g1, clqspervtx=TRUE, verbose=FALSE)

length(clqs1)

g2 <- removeEdge(from="1", to=edges(g1)[["1"]][1], g1)
g2

system.time(clqs2a <- qpGetCliques(g2, verbose=FALSE))

system.time(clqs2b <- qpUpdateCliquesRemoving(g1, clqs1, "1", edges(g1)[["1"]][1], verbose=FALSE))

length(clqs2a)

length(clqs2b)-nVar
```

```
## End(Not run)
```

`SsdMatrix-class` *Sum of squares and deviations Matrices*

Description

The "SsdMatrix" class is the class of symmetric, dense matrices in packed storage (just as a dspMatrix-class, i.e., only the upper triangle is stored) defined within the qpgraph package to store corrected, or uncorrected, matrices of the sum of squares and deviations (SSD) of pairs of random variables. A corrected SSD matrix corresponds to a sample covariance matrix.

Objects from the Class

Objects can be created by calls of the form `new("SsdMatrix", ...)` or by using `qpCov()` which estimates a sample covariance matrix from data returning an object of this class.

Slots

`ssd`: Object of class dspMatrix-class storing the SSD matrix.

`n`: Object of class "numeric" storing the sample size employed to estimate the SSD matrix stored in the slot `ssd`. This is specially relevant when the SSD matrix was estimated from data with missing values by using complete observations only, which is the default mode of operation of `qpCov()`.

Extends

"SsdMatrix" extends class "dspMatrix", directly.

Methods

dim `signature(x = "SsdMatrix")`

dimnames `signature(x = "SsdMatrix")`

show `signature(object = "SsdMatrix")`

determinant `signature(object = "SsdMatrix", logarithm = "missing")`

UGgmm-class *Undirected Gaussian graphical Markov model*

Description

The "UGgmm" class is the class of undirected Gaussian graphical Markov models defined within the qpgraph package to store simulate and manipulate this type of graphical Markov models (GMMs).

An undirected Gaussian GMM is a family of multivariate normal distributions sharing a set of conditional independences encoded by means of an undirected graph. Further details can be found in the book of Lauritzen (1996).

Objects from the Class

Objects can be created by calls of the form `UGgmm(g, ...)` corresponding to constructor methods or `rUGgmm(n, g, ...)` corresponding to random simulation methods.

Slots

`p`: Object of class "integer" storing the dimension of the undirected Gaussian GMM corresponding to the number of random variables.

`g`: Object of class graphBAM-class storing the associated undirected labeled graph.

`mean`: Object of class "numeric" storing the mean vector.

`sigma`: Object of class dspMatrix-class storing the covariance matrix.

Methods

`UGgmm(g)` Constructor method where `g` can be either an adjacency matrix or a graphBAM-class object.

`rUGgmm(n, g)` Constructor simulation method that allows one to simulate undirected Gaussian GMMs where `n` is the number of GMMs to simulate and `g` can be either a graphParam object, an adjacency matrix or a graphBAM-class object.

`names(x)` Accessor method to obtain the names of the elements in the object `x` that can be retrieved with the `$` accessor operator.

`$` Accessor operator to retrieve elements of the object in an analogous way to a `list`.

`dim(x)` Dimension of the undirected Gaussian GMM corresponding to the total number of random variables.

`dimnames(x)` Names of the random variables in the undirected Gaussian GMM.

`show(object)` Method to display some bits of information about the input undirected Gaussian GMM specified in `object`.

`summary(object)` Method to display a sumamry of the main features of the input undirected Gaussian GMM specified in `object`.

`plot(x, ...)` Method to plot the undirected graph associated to the the input undirected Gaussian GMM specified in `x`. It uses the plotting capabilities from the `Rgraphviz` library to which further arguments specified in `...` are further passed.

Author(s)

R. Castelo

References

Lauritzen, S.L. *Graphical models*. Oxford University Press, 1996.

See Also

HMgmm

Index