Package 'qgraph'

December 9, 2022

Type Package

Title Graph Plotting Methods, Psychometric Data Visualization and Graphical Model Estimation

Version 1.9.3

Maintainer Sacha Epskamp <mail@sachaepskamp.com>

Depends R (>= 3.0.0)

Imports Rcpp (>= 1.0.0), methods, grDevices, psych, lavaan, plyr, Hmisc, igraph, jpeg, png, colorspace, Matrix, corpcor, reshape2, ggplot2, glasso, fdrtool, gtools, parallel, pbapply, abind

ByteCompile yes

Description Fork of qgraph - Weighted network visualization and analysis, as well as Gaussian graphical model computation. See Epskamp et al. (2012) <doi:10.18637/jss.v048.i04>.

BugReports https://github.com/SachaEpskamp/qgraph

License GPL-2

LazyLoad yes

LinkingTo Rcpp

Suggests BDgraph, huge

NeedsCompilation yes

Author Sacha Epskamp [aut, cre], Giulio Costantini [aut], Jonas Haslbeck [aut], Adela Isvoranu [aut], Angelique O. J. Cramer [ctb], Lourens J. Waldorp [ctb], Verena D. Schmittmann [ctb], Denny Borsboom [ctb]

Repository CRAN

Date/Publication 2022-12-09 18:10:02 UTC

R topics documented:

`as.igraph.qgraph` *Converts qgraph object to igraph object.*

Description

This function converts the output of `qgraph` to an 'igraph' object that can be used in the igraph package (Csardi & Nepusz, 2006)

Usage

```
## S3 method for class 'qgraph'
as.igraph(object, attributes = TRUE)
```

Arguments

object	A "qgraph" object
attributes	Logical, should graphical attributes also be transferred?

Author(s)

Sacha Epskamp <mail@sachaepskamp.com>

References

Csardi G, Nepusz T (2006). The igraph software package for complex network research, InterJournal, Complex Systems 1695. http://igraph.sf.net

---

averageLayout *Computes an average layout over several graphs*

Description

This function can be used to compute a joint layout over several graphs.

Usage

```
averageLayout(..., layout = "spring", repulsion = 1, layout.par)
```

Arguments

...	Multiple graph objects such as qgraph objects or weights matrices.
layout	Same as in qgraph
repulsion	The repulsion parameter as used in qgraph.
layout.par	Same as in qgraph

Value

A layout matrix

Author(s)

Sacha Epskamp <mail@sachaepskamp.com>

big5 *Big 5 dataset*

Description

This is a dataset of the Big 5 personality traits that will be used in talks and the paper. It is a measurement of the Dutch translation of the NEO-PI-R on 500 first year psychology students (Dolan, Oort, Stoel, Wicherts, 2009).

Usage

```
data(big5)
```

Format

The format is: num [1:500, 1:240] 2 3 4 4 5 2 2 1 4 2 ... - attr(*, "dimnames")=List of 2 ..$: NULL ..$: chr [1:240] "N1" "E2" "O3" "A4" ...

References

Hoekstra, H. A., Ormel, J., & De Fruyt, F. (2003). NEO-PI-R/NEO-FFI: Big 5 persoonlijkheidsvragenlijst. Handleiding \[Manual of the Dutch version of the NEO-PI-R/NEO-FFI\]. Lisse, The Netherlands: Swets and Zeitlinger.

Dolan, C. V., Oort, F. J., Stoel, R. D., & Wicherts, J. M. (2009). Testing measurement invariance in the target rotates multigroup exploratory factor model. Structural Equation Modeling, 16, 295<96>314.

big5groups *Big 5 groups list*

Description

This is the groups list of the big 5 data. It is a measurement of the Dutch translation of the NEO-PI-R on 500 first year psychology students (Dolan, Oort, Stoel, Wicherts, 2009).

Usage

```
data(big5groups)
```

Format

The format is: List of 5 $ Neuroticism : num [1:48] 1 6 11 16 21 26 31 36 41 46 ... $ Extraversion : num [1:48] 2 7 12 17 22 27 32 37 42 47 ... $ Openness : num [1:48] 3 8 13 18 23 28 33 38 43 48 ... $ Agreeableness : num [1:48] 4 9 14 19 24 29 34 39 44 49 ... $ Conscientiousness: num [1:48] 5 10 15 20 25 30 35 40 45 50 ...

References

Hoekstra, H. A., Ormel, J., & De Fruyt, F. (2003). NEO-PI-R/NEO-FFI: Big 5 persoonlijkheidsvragenlijst. Handleiding [Manual of the Dutch version of the NEO-PI-R/NEO-FFI]. Lisse, The Netherlands: Swets and Zeitlinger.

Dolan, C. V., Oort, F. J., Stoel, R. D., & Wicherts, J. M. (2009). Testing measurement invariance in the target rotates multigroup exploratory factor model. Structural Equation Modeling, 16, 295<96>314.

`centrality` *Centrality statistics of graphs*

Description

This function can be used on the output of `qgraph` to compute the node centrality statistics for weighted graphs proposed by Opsahl, Agneessens and Skvoretz (2010).

Usage

```
centrality(graph, alpha = 1, posfun = abs, pkg, all.shortest.paths = FALSE,
           weighted = TRUE, signed = TRUE, R2 = FALSE)
```

Arguments

`graph`	A "qgraph" object obtained from `qgraph`
`alpha`	The tuning parameter. Defaults to 1.
`posfun`	A function that converts positive and negative values to only positive. Defaults to the absolute value.
`pkg`	Package to use. Either `"qgraph"` or `"igraph"`. Defaults to `"qgraph"` for directed networks and `"igraph"` for undirected networks.
`all.shortest.paths`	Logical if all shortest paths should be returned. Defaults to `FALSE`. Setting this to true can greatly increase computing time if `pkg = "igraph"`.
`weighted`	Logical, set to `FALSE` to set all edge weights to 1 or -1
`signed`	Logical, set to `FALSE` to make all edge weights absolute
`R2`	Logical, should R-squared (predictability) be computed for GGM structures?

Details

This function computes and returns the in and out degrees, closeness and betweenness as well as the shortest path lengths and shortest paths between all pairs of nodes in the graph. For more information on these statistics, see Opsahl, Agneessens and Skvoretz (2010).

Self-loops are ignored in computing centrality indices. These statistics are only defined for positive edge weights, and thus negative edge weights need to be transformed into positive edge weights. By default, this is done by taking the absolute value.

The algorithm used for computing the shortest paths is the well known "Dijkstra's algorithm" (Dijkstra, 1959). The algorithm has been implemented in R, which can make this function take several minutes to run for large graphs (over 100 nodes). A future version of qgraph will include a compiled version to greatly speed up this function.

Value

A list containing:

`OutDegree` A vector containing the outward degree of each node.

`InDegree` A vector containing the inward degree of each node.

`Closeness` A vector containing the closeness of each node.

`Betweenness` A vector containing the betweenness of each node

`InExpectedInfluence`
Expected incoming influence - sum of incomming edge weights connected to a node (not made absolute)

`OutExpectedInfluence`
Expected outgoing influence - sum of outgoing edge weights connected to a node (not made absolute)

`ShortestPathLengths`
A matrix containing the shortest path lengths of each pairs of nodes. These path lenghts are based on the inverse of the absolute edge weights raised to the power alpha.

`ShortestPaths` A matrix of lists containing all shortest path lengths between all pairs of nodes. Use double square brackets to index. E.g., if the list is called 'res', res$ShortestPaths[[i,j]] gives a list containing all shortest paths between node i and j.

Author(s)

Sacha Epskamp (mail@sachaepskamp.com)

References

Opsahl, T., Agneessens, F., Skvoretz, J. (2010). Node centrality in weighted networks: generalizing degree and shortest paths. Soc Netw. 32:245–251.

Dijkstra, E.W. (1959). A note on two problems in connexion with graphs. Numerische Mathematik 1, 269–271.

See Also

qgraph

Examples

```
set.seed(1)
adj <- matrix(sample(0:1,10^2,TRUE,prob=c(0.8,0.2)),nrow=10,ncol=10)
Q <- qgraph(adj)

centrality(Q)
```

centrality and clustering plots
Centrality and Clustering plots and tables

Description

These functions can be used to facilitate interpreting centrality and clustering coefficients. The plot functions use ggplot2 (Wickham, 2009). The table functions create a long format table which can easilly be plotted in ggplot2.

Usage

```
centralityPlot(..., labels, scale = c("raw0", "raw", "z-scores", "relative"),
            include =c("Degree","Strength","OutDegree","InDegree","OutStrength",
                  "InStrength"), theme_bw = TRUE, print = TRUE, verbose = TRUE,
                  standardized, relative, weighted = TRUE,signed = TRUE,
                  orderBy = "default", decreasing = FALSE)
clusteringPlot(..., scale = c("raw0", "raw", "z-scores", "relative"), labels,
                   include , signed = FALSE, theme_bw = TRUE, print = TRUE,
                   verbose = TRUE, standardized, relative,orderBy = "default",
                   decreasing = FALSE)
centralityTable(..., labels, standardized = TRUE,  relative = FALSE, weighted =
                  TRUE, signed = TRUE)
clusteringTable(..., labels,  standardized = TRUE, relative = FALSE,
                 signed = FALSE)
```

Arguments

`...`	Objects usuable in the `getWmat` generic, such as qgraph objects and weights matrices. Can also be lists containing these objects. Graphs in a list will be plotted in the same panel as different lines and graphs in seperate arguments will be plotted in seperate panels.
`scale`	Scale of the x-axis. `"z-scores"` to plot standardized coefficients, `"raw"` to plot raw coefficients, `"raw0"` to plot raw coefficients while including 0 on the x-axis and `"relative"` to show values on a relative scale from 0 (lowest) to 1 (highest).
`labels`	A vector overwriting the labels used. Can be missing.
`include`	A vector of measures to include. if missing all measures available will be included. Not included by default are `"Closeness"`, `"Betweenness"`, `"ExpectedInfluence"`, `"OutExpectedInfluence"`, and `"InExpectedInfluence"`. Can also be `"all"` or `"All"` to include all available centrality measures.
`theme_bw`	Adds the ggplot2 black and white theme to the plot
`print`	If `TRUE`, the plot is sent to the print command and returned invisible, if `FALSE` the plot is returned normally. Needed to include plots in e.g., pdf files.
`verbose`	Should messages be printed to the console?

standardized	Logical, should all measures be standardized? Argument is deprecated and will be removed.
relative	Logical, should all measures be scaled relative to the largest value? Argument is deprecated and will be removed.
weighted	Logical, set to FALSE to set all edge weights to 1 or -1
signed	Logical, set to FALSE to make all edge weights absolute
orderBy	String indicating which measure to order by. Can be default (alphabetical), or one of the measures plotted (e.g., "Strength")
decreasing	Logical indicating if the nodes should be ordered increasing or decreasing

Details

Note that under default setting the plot functions show the standardized centrality indices. That is, z-scores instead of raw centrality indices. This is done to allow easier comparison of multiple networks.

Author(s)

Sacha Epskamp <mail@sachaepskamp.com> and Jolanda Kossakowski

References

H. Wickham. ggplot2: elegant graphics for data analysis. Springer New York, 2009.

---

centrality_auto *Automatic centrality statistics of graphs*

Description

This function can be used on several kinds of graphs to compute several node centrality statistics and the edge-betweenness centrality. The input graph can be an adjacency matrix, a weight matrix, an edgelist (weighted or unweighted), a qgraph object or an igraph object.

Usage

```
centrality_auto(x, weighted = TRUE, signed = TRUE)
```

Arguments

x	A graph. Can be a qgraph object, an igraph object, an adjacency matrix, a weight matrix and an edgelist, or a weighted edgelist.
weighted	Logical, set to FALSE to set all edge weights to 1 or -1
signed	Logical, set to FALSE to make all edge weights absolute

Details

The function recognizes whether the network is unweighted vs. weighted, undirected vs. directed, and connected vs. disconnected and computes a set of centrality indices that is best suited for that particular kind of network. Edge signs are always disregarded, while edge weights and directions, if present, are considered in the computation of the indices. If the network is disconnected, closeness centrality is computed only considering the largest component (notice that this is different from what function `centrality` does).

If `x` is unweighted and directed, then the indegree, the outdegree, the node betweenness centrality, the closenes centrality, and the edge betweenness centrality are computed. If `x` is unweighted and undirected, then the degree, the node betweenness centrality, the closenes centrality, and the edge betweenness centralities are computed. If `x` is weighted and directed, then the instrength and the outstrength (same as indegree and outdegree, but considering weights), the node betweenness centrality, the closeness centrality, and edge betweenness centralities are computed If `x` is weighted and undirected, then the strength, the node betweenness centrality, the closenes centrality, and edge betweenness centralities are computed. Additionally, the shortest path length between each pair of nodes is also computed for all the kinds of networks.

Value

A list containing:

`node.centrality`
: A dataframe that includes the node centrality statistics. A subset of the following centrality indices is included, depending on the input network: `Degree`, `InDegree`, `OutDegree`, `Strength`, `InStrength`, `OutStrength`, `Betweenness`, and `Closeness`.

`ShortestPathLengths`
: A matrix containing the shortest path lengths of each pairs of nodes. These path lenghts are based on the inverse of the absolute edge weights.

`edge.betweenness.centrality`
: The edge betweenness centrality statistic (Newman & Girvan, 2004). Edges are ordered by their decreasing centrality.

Author(s)

Giulio Costantini (giulio.costantini@unimib.it), Sacha Epskamp (mail@sachaepskamp.com)

References

Newman, M. E. J., Girvan, M. (2004). Finding and evaluating community structure in networks. Phisical Review E 69(026113).

Costantini, G., Epskamp, S., Borsboom, D., Perugini, M., Mõttus, R., Waldorp, L., Cramer, A. O. J., State of the aRt personality research: A tutorial on network analysis of personality data in R. Manuscript submitted for publication.

See Also

`qgraph`, `centrality`

Examples

```
set.seed(1)
adj <- matrix(sample(0:1,10^2,TRUE,prob=c(0.8,0.2)),nrow=10,ncol=10)
Q <- qgraph(adj)
centrality_auto(Q)
# notice that a value NA is returned for the closeness centrality of nodes 3 and 9, which are not
# strongly connected to the largest component of the network (3 cannot reach other nodes, 9 cannot
# be reached).
```

---

clustcoef_auto *Local clustering coefficients.*

Description

Compute local clustering coefficients, both signed and unsigned and both for weighted and for unweighted networks.

Usage

```
clustcoef_auto(x, thresholdWS = 0, thresholdON = 0)
clustWS(x, thresholdWS=0)
clustZhang(x)
clustOnnela(x, thresholdON=0)
```

Arguments

x	An undirected graph. Can be a qgraph object, an igraph object, an adjacency matrix, a weight matrix and an edgelist, or a weighted edgelist.
thresholdWS	The threshold used to binarize a weighted network x to compute the binary clustering coefficients clustWS and signed_clustWS. Edges with weights lower than thresholdWS in absolute value are zeroed. For unweighted networks, thresholdWS = 0 is the suggested value.
thresholdON	In the computation of Onnela's clustering coefficient clustOnnela, edge of weights lower than thresholdON in absolute value are excluded. The value thresholdON = 0 (i.e., no edge is excluded) is generally suggested also for weighted networks.

Details

clustWS computes the clustering coefficient for unweighted networks introduced by Watts & Strogatz (1998) and the corresponding signed version (Costantini & Perugini, in press). ClustZhang computes the clustering coefficient for weighted networks introduced by Zhang & Horvath (2005) and the corresponding signed version (Costantini & Perugini, in press). clustOnnela computes the clustering coefficient for weighted networks introduced by Onnela et al. (2005) and the corresponding signed version (Costantini & Perugini, in press). clustering_auto automatically recognizes the kind of the input network x (weighted vs. unweighted, signed vs. unsigned) and computes a subset of indices according to the kind of the network: signed indices are not computed for unsigned

networks and weighted indices are not computed for unweighted networks. However the unsigned indices are computed for signed networks, by considering the absolute value of the weights, and the unweighted indices are computed for weighted networks, after a binarization according to the parameter `thresholdWS`. `clustering_auto` computes also the weighted clustering coefficient by Barrat et al. (2004), relying on function `transitivity` from package `igraph`. For the computation of the local clustering coefficient, a node must have at least two neighbors: for nodes with less than two neighbors `NaN` is returned.

Value

A dataframe that includes one or more of the following indices.

`clustWS`	The Watts & Strogatz's (1998) unweighted clustering coefficient
`signed_clustWS`	The signed version of the Watts & Strogatz's clustering coefficient
`clustZhang`	The Zhang & Horvath's (2005) weighted clustering coefficient
`signed_clustZhang`	The signed version of the Zhang & Horvath's (2005) clustering coefficient
`clustOnnela`	The Onnela et al.'s (2005) clustering coefficient
`signed_clustOnnela`	The signed version of the Onnela et al.'s (2005) clustering coefficient
`clustBarrat`	The Barrat et al.'s (2004) weighted clustering coefficient

Warning

The function requires an undirected network. To convert a directed network to undirected use for instance function `upper.tri` (see examples).

Note

Part of the code has been adapted from package `WGCNA` (Langfelder & Horvath, 2008).

Author(s)

Giulio Costantini (giulio.costantini@unimib.it), Sacha Epskamp (mail@sachaepskamp.com)

References

Barrat, A., Barthelemy, M., Pastor-Satorras, R., & Vespignani, A. (2004). The architecture of complex weighted networks. In Proc. Natl. Acad. Sci. USA 101 (pp. 3747-3752).

Costantini, G., Perugini, M. (in press), Generalization of Clustering Coefficients to Signed Correlation Networks

Langfelder, P., & Horvath, S. (2008). WGCNA: an R package for weighted correlation network analysis. BMC Bioinformatics, 9, 559.

Onnela, J. P., Saramaki, J., Kertesz, J., & Kaski, K. (2005). Intensity and coherence of motifs in weighted complex networks. Physical Review E, 71(6), 065103.

Watts, D. J., & Strogatz, S. H. (1998). Collective dynamics of "small-world" networks. Nature, 393(6684), 440-442.

Zhang, B., & Horvath, S. (2005). A general framework for weighted gene co-expression network analysis. Statistical Applications in Genetics and Molecular Biology, 4(1).

See Also

centrality_auto

Examples

```
set.seed(1)
# generate a random (directed) network:
net_ig <- igraph::erdos.renyi.game(n=8, p.or.m=.4, type="gnp", directed=TRUE)

# convert it to an adjacency matrix:
net <- as.matrix(igraph:::get.adjacency(net_ig, type="both"))

# convert it to a signed and weighted network:
net <- net*matrix(rnorm(ncol(net)^2), ncol=ncol(net))

# make it undirected:
net[upper.tri(net)] <- t(net)[upper.tri(net)]
clustcoef_auto(net)
```

---

cor_auto *Automatically compute an apppropriate correlation matrix*

Description

This is mainly a wrapper around Lavaan function lavCor (Rosseel, 2012) to compute a correlation matrix based on pychoric, polyserial and/or Pearson correlations. The wrapper removes all factors and searches for possible ordinal variabes. A variable is classified as ordinal if it is either `ordered` or if it consist of at most 7 unique integer values. After computing the correlations an additional check will be performed to see if the correlation matrix is positive definite.

Usage

```
cor_auto(data, select, detectOrdinal = TRUE, ordinalLevelMax = 7, npn.SKEPTIC = FALSE,
         forcePD = FALSE, missing = "pairwise", verbose = TRUE)
```

Arguments

`data`	A data frame
`select`	Variables to select from the data frame (as used in subset)
`detectOrdinal`	Logical, should ordinal variables be detected? If `FALSE` only variables that are `ordered` are treated as ordinal variables
`ordinalLevelMax`	Integer specyfying the amount of unique integer values a variable should have to be classified as ordinal

npn.SKEPTIC	Logical, should the Nonparanormal SKEPTIC from the huge package be applied if the data is continuous? See huge.npn (Zhao, Liu, Roeder, Lafferty and Wasserman, 2014)
forcePD	If TRUE the function checks if the correlation matrix is positive definite. If the matrix is not positive definite nearPD from the Matrix package will be used (Bates and Maechler, 2014).
missing	Corresponds to the missing argument in lavCor
verbose	Logical, should information be printed to the console?

Value

A correlation matrix

Author(s)

Sacha Epskamp <mail@sachaepskamp.com>

References

Yves Rosseel (2012). lavaan: An R Package for Structural Equation Modeling. Journal of Statistical Software, 48(2), 1-36. URL http://www.jstatsoft.org/v48/i02/.

Tuo Zhao, Han Liu, Kathryn Roeder, John Lafferty and Larry Wasserman (2014). huge: High-dimensional Undirected Graph Estimation. R package version 1.2.6. http://CRAN.R-project.org/package=huge

Douglas Bates and Martin Maechler (2014). Matrix: Sparse and Dense Matrix Classes and Methods. R package version 1.1-3. http://CRAN.R-project.org/package=Matrix

Examples

```
## Not run:
### Examples from lavCor (lavaan): ###

library("lavaan")

# Holzinger and Swineford (1939) example
HS9 <- HolzingerSwineford1939[,c("x1","x2","x3","x4","x5",
                                 "x6","x7","x8","x9")]

# Pearson correlations
cor_auto(HS9)

# ordinal version, with three categories
HS9ord <- as.data.frame( lapply(HS9, cut, 3, labels=FALSE) )

# polychoric correlations, two-stage estimation
cor_auto(HS9ord)

## End(Not run)
```

EBICglasso *Compute Gaussian graphical model using graphical lasso based on extended BIC criterium.*

Description

This function uses the glasso package (Friedman, Hastie and Tibshirani, 2011) to compute a sparse gaussian graphical model with the graphical lasso (Friedman, Hastie and Tibshirani, 2008). The tuning parameter is chosen using the Extended Bayesian Information criterium (EBIC).

Usage

```
EBICglasso(S, n, gamma = 0.5, penalize.diagonal = FALSE, nlambda = 100,
           lambda.min.ratio = 0.01, returnAllResults = FALSE, checkPD = TRUE,
         penalizeMatrix, countDiagonal = FALSE, refit = FALSE, threshold = FALSE,
           verbose = TRUE, ...)
```

Arguments

S	A covariance or correlation matrix
n	Sample size used in computing S
gamma	EBIC tuning parameter. 0.5 is generally a good choice. Setting to zero will cause regular BIC to be used.
penalize.diagonal	Should the diagonal be penalized?
nlambda	Number of lambda values to test.
lambda.min.ratio	Ratio of lowest lambda value compared to maximal lambda
returnAllResults	If TRUE this function does not return a network but the results of the entire glasso path.
checkPD	If TRUE, the function will check if S is positive definite and return an error if not. It is not advised to use a non-positive definite matrix as input as (a) that can not be a covariance matrix and (b) glasso can hang if the input is not positive definite.
penalizeMatrix	Optional logical matrix to indicate which elements are penalized
countDiagonal	Should diagonal be counted in EBIC computation? Defaults to FALSE. Set to TRUE to mimic qgraph < 1.3 behavior (not recommended!).
refit	Logical, should the optimal graph be refitted without LASSO regularization? Defaults to FALSE.
threshold	Logical, should elements of the precision matrix that are below (log(p*(p-1)/2)) / sqrt(n) be removed (both before EBIC computation and in final model)? Set to TRUE to ensure high specificity.
verbose	Logical, should progress output be printed to the console?
...	Arguments sent to glasso

Details

The glasso is run for 100 values of the tuning parameter logarithmically spaced between the maximal value of the tuning parameter at which all edges are zero, lamba_max, and lambda_max/100. For each of these graphs the EBIC is computed and the graph with the best EBIC is selected. The partial correlation matrix is computed using `wi2net` and returned. When `threshold = TRUE`, elements of the inverse variance-covariance matrix are first thresholded using the theoretical bound (Jankova and van de Geer, 2018).

Value

A partial correlation matrix

Author(s)

Sacha Epskamp <mail@sachaepskamp.com>

References

Friedman, J., Hastie, T., & Tibshirani, R. (2008). Sparse inverse covariance estimation with the graphical lasso. Biostatistics, 9(3), 432-441. Chicago

Jerome Friedman, Trevor Hastie and Rob Tibshirani (2011). glasso: Graphical lasso-estimation of Gaussian graphical models. R package version 1.7. http://CRAN.R-project.org/package=glasso

Foygel, R., & Drton, M. (2010, November). Extended Bayesian Information Criteria for Gaussian Graphical Models. In NIPS (pp. 604-612). Chicago

Revelle, W. (2014) psych: Procedures for Personality and Psychological Research, Northwestern University, Evanston, Illinois, USA, http://CRAN.R-project.org/package=psych Version = 1.4.4.

Bates, D., and Maechler, M. (2014). Matrix: Sparse and Dense Matrix Classes and Methods. R package version 1.1-3. http://CRAN.R-project.org/package=Matrix

Jankova, J., and van de Geer, S. (2018) Inference for high-dimensional graphical models. In: Handbook of graphical models (editors: Drton, M., Maathuis, M., Lauritzen, S., and Wainwright, M.). CRC Press: Boca Raton, Florida, USA.

Examples

```
## Not run:
### Using bfi dataset from psych ###
library("psych")
data(bfi)

# Compute correlations:
CorMat <- cor_auto(bfi[,1:25])

# Compute graph with tuning = 0 (BIC):
BICgraph <- EBICglasso(CorMat, nrow(bfi), 0, threshold = TRUE)

# Compute graph with tuning = 0.5 (EBIC)
EBICgraph <- EBICglasso(CorMat, nrow(bfi), 0.5, threshold = TRUE)
```

```
# Plot both:
layout(t(1:2))
BICgraph <- qgraph(BICgraph, layout = "spring", title = "BIC", details = TRUE)
EBICgraph <- qgraph(EBICgraph, layout = "spring", title = "EBIC")

# Compare centrality and clustering:
layout(1)
centralityPlot(list(BIC = BICgraph, EBIC = EBICgraph))
clusteringPlot(list(BIC = BICgraph, EBIC = EBICgraph))

## End(Not run)
```

FDRnetwork *Model selection using local False Discovery Rate*

Description

This function is a wrapper arounf `fdrtool` to easilly compute a correlation or partial correlation network in which all nonsignificant edges are set to zero.

Usage

```
FDRnetwork(net, cutoff = 0.1, method = c('lfdr', 'pval', 'qval'))
```

Arguments

`net`	A correlation or partial correlation matrix
`cutoff`	The cutoff value to use. The edges of which the value of the first element of `method` are higher than the cutoff are removed. Thus, by default, edges with a local false discovery rate of higher than 0.1 are removed from the graph.
`method`	The method to use with the cutoff. Can be `'lfdr'` for the local false discobvery rate, `'pval'` for the p-value of `'qval'` for the q-value.

Details

`method = 'lfdr'` could result in a very sparse network, so also looking at other values is advisable.

Author(s)

Sacha Epskamp <mail@sachaepskamp.com>

References

Bernd Klaus and Korbinian Strimmer. (2014). fdrtool: Estimation of (Local) False Discovery Rates and Higher Criticism. R package version 1.2.12. http://CRAN.R-project.org/package=fdrtool

Examples

```
## Not run:
### Using bfi dataset from psych ###
library("psych")
data(bfi)

### CORRELATIONS ###
# Compute correlations:
CorMat <- cor_auto(bfi[,1:25])

# Run local FDR:
CorMat_FDR <- FDRnetwork(CorMat)

# Number of edges remaining:
mean(CorMat_FDR[upper.tri(CorMat_FDR,diag=FALSE)]!=0)

# None, so might use different criterion:
CorMat_FDR <- FDRnetwork(CorMat, method = "pval")


# Compare:
L <- averageLayout(CorMat, CorMat_FDR)

layout(t(1:2))
qgraph(CorMat, layout = L, title = "Correlation network",
       maximum = 1, cut = 0.1, minimum = 0, esize = 20)
qgraph(CorMat_FDR, layout = L, title = "Local FDR correlation network",
       maximum = 1, cut = 0.1, minimum = 0, esize = 20)

# Centrality:
centralityPlot(list(cor=CorMat, fdr = CorMat_FDR))


### PARTIAL CORRELATIONS ###
# Partial correlation matrix:
library("parcor")
PCorMat <- cor2pcor(CorMat)

# Run local FDR:
PCorMat_FDR <- FDRnetwork(PCorMat, cutoff = 0.1, method = "pval")

# Number of edges remaining:
mean(PCorMat_FDR[upper.tri(PCorMat_FDR,diag=FALSE)]!=0)

# Compare:
L <- averageLayout(PCorMat, PCorMat_FDR)

layout(t(1:2))
qgraph(PCorMat, layout = L, title = "Partial correlation network",
          maximum = 1, cut = 0.1, minimum = 0, esize = 20)
qgraph(PCorMat_FDR, layout = L, title = "Local FDR partial correlation network",
          maximum = 1, cut = 0.1, minimum = 0, esize = 20)
```

```
# Centrality:
centralityPlot(list(cor=PCorMat, fdr = PCorMat_FDR))

## End(Not run)
```

`flow` *Draws network as a flow diagram showing how one node is connected to all other nodes*

Description

This function will draw one node of interest on the left, then subsequently draw all other nodes in vertical levels to the right, in the order of direct (unweighted) connectiveness to the node of interest. Layout is based on the `layout_as_tree` function from the igraph package. This allows one to see how one node connects to other nodes in the network.

Usage

```
flow(object, from, horizontal = TRUE, equalize = TRUE, minCurve = 1, maxCurve = 4,
      unfadeFirst = FALSE, fade = TRUE, labels, ...)
```

Arguments

`object`	A qgraph object
`from`	Integer or character indicating the (label of the) node of interest.
`horizontal`	Logical, should the flow diagram be plotted horizontally or vertically
`equalize`	Logical, should the placement of nodes be equalized per level.
`minCurve`	Minimum curve of edges on the same level
`maxCurve`	Maximum curve of edges on the same level
`unfadeFirst`	Logical, should edges between the node of interest be unfaded?
`fade`	'fade' argument as used in qgraph
`labels`	'labels' argument as used in qgraph
`...`	Arguments sent to qgraph

Author(s)

Sacha Epskamp

Examples

```
## Not run:
# Load data:
library("psych")
data(bfi)

# Compute polychoric correlations:
corMat <- cor_auto(bfi[,1:25])

# Glasso network:
g2 <- qgraph(corMat,  cut = 0, graph = "glasso", sampleSize = nrow(bfi),
             threshold = TRUE)

# Flow from A2:
flow(g2, "A2", horizontal = TRUE)

## End(Not run)
```

`getWmat` *Obtain a weights matrix*

Description

This function extracts a weights matrix from various kinds of objects.

Usage

```
## S3 method for class 'matrix'
getWmat(x,nNodes,labels, directed = TRUE,...)
## S3 method for class 'data.frame'
getWmat(x,nNodes,labels, directed = TRUE,...)
## S3 method for class 'igraph'
getWmat(x,labels,...)
## S3 method for class 'qgraph'
getWmat(x,directed,...)
```

Arguments

`x`	An input object
`nNodes`	Number of Nodes
`labels`	A vector specifying the labels of each node
`directed`	Logical indicating if the graph should be directed
`...`	Ignored

Value

A weights matrix

Author(s)

Sacha Epskamp <mail@sachaepskamp.com>

`ggmFit` *Obtain fit measures of a Gaussian graphical model*

Description

Obtain fit measures of a given Gaussian graphical model (GGM). Input can be either a partial correlation matrix, inverse covariance matrix or `qgraph` object.

Usage

```
ggmFit(pcor, covMat, sampleSize, refit = TRUE, ebicTuning = 0.5,
        nPar, invSigma, tol = sqrt(.Machine$double.eps), verbose = TRUE,
        countDiagonalPars = TRUE)
```

Arguments

`pcor`	Implied partial correlation matrix or `qgraph` object.
`covMat`	Observed variance-covariance matrix
`sampleSize`	The sample size used in computing the variance-covariance matrix
`refit`	Logical, should the network be refitted using `glasso`?
`ebicTuning`	EBIC tuning parameter.
`invSigma`	Implied inverse variance-covariance matrix. If this object is assigned `pcor` is not used.
`nPar`	Number of parameters, if not specified this is retrieved from the number of zeroes in the inverse variance–covariance matrix. Can be used to compute fit measures of any statistical model (e.g., SEM).
`tol`	Tolerance for setting an edge to zero.
`verbose`	Logical, should progress reports be printed to the console?
`countDiagonalPars`	Logical, should the diagonal of the precision matrix be counted as parameters?

Author(s)

Sacha Epskamp <mail@sachaepskamp.com>

Examples

```
library("psych")

# Load BFI data:
data(bfi)
bfi <- bfi[,1:25]

# Covariance matrix:
CovMat <- cov(bfi[,1:25], use="pairwise.complete.obs")

# Compute network:
EBICgraph <- qgraph(CovMat, graph = "glasso", sampleSize = nrow(bfi),
                    tuning = 0.5, layout = "spring", title = "BIC", details = TRUE)

# Obtain fit measures:
fitNetwork <- ggmFit(EBICgraph, CovMat, nrow(bfi))
fitNetwork
```

`ggmModSelect` *Unregularized GGM model search*

Description

This function will search for an optimal Gaussian graphical model by minimizing the (extended) Bayesian information criterion of unregularized GGM models. Selecting unregularized GGMs according to EBIC has been shown to converge to the true model (Foygel & Drton, 2010). The algorithm starts with refitting models from the glassopath, and subsequently adds and removes edges until EBIC can no longer be improved (see details). Note, contrary to `EBICglasso`, the default for the EBIC hyperparameter gamma is set to 0, indicating BIC model selection.

Usage

```
ggmModSelect(S, n, gamma = 0, start = c("glasso", "empty", "full"), stepwise = TRUE,
  considerPerStep = c("subset", "all"), verbose = TRUE, nCores = 1, checkPD = TRUE,
   criterion = 'ebic', ...)
```

Arguments

`S`	A covariance or correlation matrix
`n`	Sample size used in computing `S`
`gamma`	EBIC tuning parameter. 0 (default) leads to BIC model selection. 0.25 or 0.5 are typical choices for more conservative model selection.
`start`	What model should stepwise search start from? `"glasso"` to first run glasso to obtain the best fitting model, `"empty"` for an empty network, `"full"` for a saturated network, or a matrix encoding the starting network.
`stepwise`	Logical indicating if stepwise model search should be used.

`considerPerStep`
: `"subet"` to only consider changing edges that previously indicated improvement in EBIC, unless changing no edge indicated an improvement to EBIC, in which case all edges are again considered (see details). `"all"` will consider changing all edges at every step.

`verbose`
: Logical, should progress reports be printed to the console?

`nCores`
: The number of cores to use in testing models.

`checkPD`
: If `TRUE`, the function will check if `S` is positive definite and return an error if not. It is not advised to use a non-positive definite matrix as input as (a) that can not be a covariance matrix and (b) glasso can hang if the input is not positive definite.

`criterion`
: String indicating an output of `ggmFit` to be minimized

`...`
: Arguments sent to `glasso`

Details

The full algorithm is as follows:

1. Run glasso to obtain 100 models

2. Refit all models without regularization

3. Choose the best according to EBIC

4. Test all possible models in which one edge is changed (added or removed)

5. If no edge can be added or changed to improve EBIC, stop here

6. Change the edge that best improved EBIC, now test all other edges that would have also lead to an increase in EBIC again

7. If no edge can be added or changed to improve EBIC, go to 4, else, go to 6.

When `stepwise = FALSE`, steps 4 to 7 are ignored. When `considerPerStep = "all"`, all edges are considered at every step. Note that this algorithm is very slow in higher dimensions (e.g., above 30-40 nodes). Note that EBIC computation is slightly different as in `EBICglasso` and instead follows the implementation in Lavaan.

Value

A list with the following elements:

`graph`
: The optimal partial correlation network

`EBIC`
: EBIC corresponding to optimal network.

Author(s)

Sacha Epskamp

References

Foygel, R., & Drton, M. (2010). Extended Bayesian information criteria for Gaussian graphical models. In Advances in neural information processing systems (pp. 604-612).

Examples

```
## Not run:
# Load data:
library("psych")
data(bfi)

# Compute polychoric correlations:
corMat <- cor_auto(bfi[,1:25])

# Optimize network:
Results <- ggmModSelect(corMat, nrow(bfi), gamma = 0.5, nCores = 8)

# Plot results:
qgraph(Results$graph, layout = "spring", cut = 0)

## End(Not run)
```

---

makeBW *A qgraph plot can be understood in black and white*

Description

Plot a qgraph network that can be understood also in black and white or grayscale. Positive lines are full and negative ones are dashed. Nodes colors are associated to unique motifs. Up to 12 different motifs are supported at the moment.

Usage

```
makeBW(x, colorlist = NA,  plot = TRUE)
```

Arguments

x	A qgraph object
colorlist	Optional: a vector of colors. See details.
plot	logical: if FALSE, the network is not plotted.

Details

If no colorlist is specified, each color is randomly associated to one of the motifs. Specifying colorlist serves for (a) assigning colors to a specific motif, because the first color in the vector will always be associated to the first motif (this can be used e.g., for being consistent across plots), or (b) for associating motifs only to some of the colors, but not to others, since only in colors in the colorlist are associated to motifs if a colorlist is specified.

Value

Silently returns a qgraph object "x" in which two new elements are present, "$graphAttributes$Nodes$density" and "$graphAttributes$Nodes$angles", which affect how the nodes are plotted. Can also be further customized and then re-plotted using plot(x).

Author(s)

Giulio Costantini

Examples

```
set.seed(1)
x <- cor(matrix(rnorm(25), nrow = 5))
colors <- c("red", "red", "blue", "blue", "white")

# colored qgraph plot
qg <- qgraph(x, colors = colors)

# randomly assing motifs to colors (notice that white nodes stay white)
makeBW(qg)
# associate a motif only to one of the colors
makeBW(qg, colorlist = c("blue"))
# define an order, which allows to choose motifs
makeBW(qg, colorlist = c("blue", "red"))
makeBW(qg, colorlist = c("red", "blue"))
```

---

mat2vec *Weights matrix to vector*

Description

Converts a weights matrix to a vector of weights. If the matrix is symmetrical only upper triangle values are returned in the vector.

Usage

```
mat2vec(x, diag = FALSE, tol = 1e-10)
```

Arguments

x	A weights matrix
diag	Logical: should diagonal values be included?
tol	Tolerance level

Author(s)

Sacha Epskamp <mail@sachaepskamp.com>

mutualInformation *Computes the mutual information between nodes*

Description

Computes the mutual information from one node to all other nodes, or between sets of nodes.

Usage

```
mutualInformation(ggm, from, to = "all", covMat)
```

Arguments

ggm	Partial correlation network. Can be missing if 'covMat' is supplied.
from	Integer vector corresponding to one set of nodes. Defaults to all nodes.
to	Integer vector corresponding to another set of nodes, or 'all' to compute the mutual information of each node to all other nodes.
covMat	Variance-covariance matrix. Can be missing if 'ggm' is supplied.

Author(s)

Sacha Epskamp

pathways *Highlight shortest pathways in a network*

Description

This function highlights the shortest paths between nodes in a network made by qgraph. Based on Isvoranu et al. (2016).

Usage

```
pathways(graph, from, to, fading = 0.25, lty = 3)
```

Arguments

graph	Output from qgraph.
from	A vector indicating the first set of nodes between which pathways should be highlighted. Can be numeric or characters corresponding to node labels.
to	A vector indicating the second set of nodes between which pathways should be highlighted. Can be numeric or characters corresponding to node labels.
fading	The fading of the edges that are not part of shortest paths between 'from' and 'to'.
lty	The line type of the edges that are not part of shortest paths between 'from' and 'to'.

Author(s)

Sacha Epskamp and Adela M. Isvoranu

References

Isvoranu, A. M., van Borkulo, C. D., Boyette, L. L., Wigman, J. T., Vinkers, C. H., Borsboom, D., & Group Investigators. (2016). A Network Approach to Psychosis: Pathways Between Childhood Trauma and Psychotic Symptoms. Schizophrenia bulletin, sbw055.

See Also

qgraph

Examples

```
library("qgraph")
library("psych")
data(bfi)

# Compute correlations:
CorMat <- cor_auto(bfi[,1:25])

# Compute graph with tuning = 0 (BIC):
BICgraph <- qgraph(CorMat, graph = "glasso", sampleSize = nrow(bfi),
                   tuning = 0, layout = "spring", title = "BIC", details = TRUE)

# All paths between Agreeableness and Neuroticism:
pathways(BICgraph,
         from = c("A1","A2","A3","A4","A5"),
         to = c("N1","N2","N3","N4","N5"))
```

---

plot.qgraph *Plot method for "qgraph"*

Description

Plots an object created by qgraph.

Usage

```
## S3 method for class 'qgraph'
plot(x, ...)
```

Arguments

x	A "qgraph" object
...	Not used

Details

If the result of qgraph is stored, such as `Graph <- qgraph(...)`, the plot can be recreated in two ways. `qgraph(Graph, ...))` reruns qgraph with the same arguments used in the origina call except those restated in the dots. For example `qgraph(Graph, shape = "square")` will recreate the same plot but now use square nodes instead of circular. `plot(Graph)` will NOT rerun qgraph but simply plot the qgraph object. This means that now specific graph attributes can be changed before plotting.

More specific, `qgraph(Graph)` will base the new plot on the `Arguments` element of the qgraph object and `plot(qgraph)` will base the new plot on the `graphAttributes` element of qgraph.

Author(s)

Sacha Epskamp (mail@sachaepskamp.com)

---

print.qgraph *Print edgelist*

Description

This function prints the edgelist of a "qgraph" object

Usage

```
## S3 method for class 'qgraph'
print(x, ...)
```

Arguments

x	A "qgraph" object
...	These arguments are not used

Author(s)

Sacha Epskamp (mail@sachaepskamp.com)

See Also

qgraph

qgraph *qgraph*

Description

This is the main function of qgraph which automatically creates an appropriate network and sends it to the plotting method.

Usage

```
qgraph( input, ... )
```

Arguments

input	Can be either a weights matrix or an edgelist. Can also be an object of class "sem" (sem), "mod" (sem), "lavaan" (lavaan), "principal" (psych), "loadings" (stats), "factanal" (stats), "graphNEL" (Rgraphviz), "pcAlgo" (pcalg), "huge" (huge), "select" (huge) or the output of glasso
...	Any additional arguments described below. Also a list with class "qgraph" can be added that contains any of these arguments (this is returned invisibly by the function)

Details

Because of the amount of arguments the usage of the qgraph function has been reduced by using the ... method for clarity. This does mean that arguments need to be specified by using their exact name. For instance, to specify color="red" you can not use col="red".

Important to note is that qgraph needs to compute in many graphs where the border of nodes are in the plotting area. If the graph is manually rescaled (such as through the "zoom" option in RStudio) the plotting area is changed. This means that the computed location of the border of nodes is no longer valid if the nodes are to remain perfectly square or circular. To overcome this, the usePCH argument can be used. If this argument is set to FALSE nodes will be plotted as polygons meaning they will rescale with rescaling the graph (circles can become ovals) and not have perfect resolution in PDF files. If usePCH is set to TRUE a default plotting symbol is used meaning the graph can not be rescaled but the node will look good in PDF. By defaut, qgraph sets usePCH to TRUE if it detects the graph is stored in a file.

While the usePCH argument makes graphs rescalable it is not a perfect solution. It is highly recommended to NOT RESCALE PLOTTING AREAS when using qgraph, or to rerun qgraph after the plotting area is rescaled. This means using save graph option fro RStudio shoud be avoided in favor of the filetype argument in qgraph

Value

qgraph returns (invisibly) a 'qgraph' object containing:

Edgelist	A list containing for each edge the node of origin, node of destination, weight en wether the edge is directed and bidirectional.

Arguments	A list containing the arguments used in the qgraph call.
plotOptions	A list containing numerous options used in the plotting method.
graphAttributes	A list containing numerous attributes for nodes, edges and the entire graph
layout	A matrix containing the layout used in the plot
layout.orig	A matrix containing the original (unscaled) layout.

Important additional arguments

layout This argument controls the layout of the graph. "circle" places all nodes in a single circle, "groups" gives a circular layout in which each group is put in separate circles and "spring" gives a force embedded layout. It also can be a matrix with a row for each node and x and y coordinates in the first and second column respectively. Defaults to "circular" in weighted graphs without a groups list, "groups" in weighted graphs with a groups list, and "spring" in unweighted graphs. Can also be a function from the igraph package.

groups An object that indicates which nodes belong together. Can be a list in which each element is a vector of integers identifying the numbers of the nodes that belong together, or a factor.

minimum Edges with absolute weights under this value are not shown (but not omitted). Defaults to 0. Can also be set to `"sig"` to only show significant edges for `graph = "cor"` and `graph = "pcor"`). Significance level is controlled by `alpha` and `bonf` arguments

maximum qgraph regards the highest of the maximum or highest absolute edge weight as the highest weight to scale the edge widths too. To compare several graphs, set this argument to a higher value than any edge weight in the graphs (typically 1 for correlations).

cut In weighted graphs, this argument can be used to cut the scaling of edges in width and color saturation. Edges with absolute weights over this value will have the strongest color intensity and become wider the stronger they are, and edges with absolute weights under this value will have the smallest width and become vaguer the weaker the weight. If this is set to `0`, no cutoff is used and all edges vary in width and color. Defaults to 0 for graphs with less then 20 nodes. For larger graphs the `cut` value is automatically chosen to be equal to the maximum of the 75th quantile of absolute edge strengths or the edge strength corresponding to 2n-th edge strength (n being the number of nodes.)

details Logical indicating if minimum, maximum and cutoff score should be printed under the graph. Defaults to FALSE.

threshold A numeric value that defaults to 0. Edges with absolute weight that are not above this value are REMOVED from the network. This differs from `minimum` which simply changes the scaling of width and color so that edges with absolute weight under minimum are not plotted/invisible. Setting a threshold influences the spring layout and centrality measures obtained with the graph whereass setting a minimum does not. In the case of correlation (`graph = "cor"`) or partial correlation (`graph = "pcor"`) networks this argument can also be given a string to omit insignificant edges. See description of this argumet in the next section (Additional options for correlation/covariance matrices).

palette The palette used for coloring nodes when the `groups` argument is used. Can be one of `"rainbow"` (default), `"colorblind"` (making use of http://jfly.iam.u-tokyo.ac.jp/color/), `"pastel"`, `"gray"`, `"R"` and `"ggplot2"`.

theme This argument sets different defaults for various graphical arguments (most notably `posCol`, `negCol` and `palette`). Can be `"classic"`, `"colorblind"`, `"gray"`, `"Hollywood"`, `"Borkulo"`, `"gimme"`, `"TeamFortress"`, `"Reddit"`, `"Leuven"` or `"Fried"`.

Additional options for correlation/covariance matrices

graph Type of graph to be made when a correlation or covariance matrix is used as input. Setting this to other values than `"default"` will check if the matrix is a correlation or covariance matrix; if the matrix is not positive definite `nearPD` from the Matrix package will be used. Options are:

"cor" Plots a correlation network. Runs `cov2cor` if input is detected to be a covariance matrix and plots the input as is

"pcor" Plots a partial correlation network, using `cor2pcor` from the parcor package (Kraemer, Schaefer and Boulesteix, 2009) on the input matrix

"glasso" Will run `EBICglasso` to obtain an optimal sparse estimate of the partial correlation matrix using the glasso package (Friedman, Hastie and Tibshirani, 2011)

Outdated and limited supported options are `"factorial"`, which will create a graph based on an exploratory factor analysis, and `"sig"` will transform all correlations in p-values (using the fdrtool package; Korbinian Strimmer, 2014) and force mode="sig". "sig2" will do the same but show p-values based on negative statistics in shades of orange

threshold In addition to a numeric value to omit edges this argument can also be assigned a string to omit insignficant edges. Note that this REMOVES edges from the network (which influences centrality measures and the spring layout). Can be `"sig"` to compute significance without correction for multiple testing, `"holm"`, `"hochberg"`, `"hommel"`, `"bonferroni"`, `"BH"`, `"BY"`, `"fdr"` or `"none"` which are used directly in the `adjust` argument in `corr.p` of the psych package (Revelle, 2014). In addition, this argument can be assigned `"locfdr"` in which edges are set to zero if the local FDR is below `FDRcutoff`. `fdrtool` from the fdrtool package (Klaus and Strimmer, 2014) is used to compute these measures, which is used inside `FDRnetwork`.

sampleSize The sample-size. Used when `graph = "glasso"` or `minimum = "sig"`

tuning A tuning parameter used in estimation. Currently only used when `graph = "glasso"` and corresponds to the `gamma` argument

lambda.min.ratio The minimal lambda ratio used in `EBICglasso`, defaults to 0.01.

gamma Alias for tuning (overwrites the tuning argument).

refit Logical, should the optimal graph be refitted without LASSO regularization? Defaults to `FALSE` and only used if `graph = "glasso"`.

countDiagonal Should diagonal be counted in EBIC computation? Defaults to `FALSE`. Set to `TRUE` to mimic qgraph < 1.3 behavior (not recommended!).

alpha The significance level (defaults to 0.05) to be used for not showing edges if `minimum = "sig"`.

bonf Logical indicating if a bonferonni correction should be applied if `minimum = "sig"`.

FDRcutoff Cutoff used in which partial correlations should be included if `graph = "fdr"`. Defaults to 0.9

Output arguments

mar A vector of the form c(bottom, left, top, right) which gives the margins. Works similar to the argument in par(). Defaults to c(3,3,3,3)

filetype A character containing the file type to save the output in. "R" outputs in a new R window, "pdf" creates a pdf file. "svg" creates a svg file (requires RSVGTipsDevice). "tex" creates LaTeX code for the graph (requires tikzDevice). 'jpg', 'tiff' and 'png' can also be used. If this is given any other string (e.g. filetype="") no device is opened. Defaults to 'R' if the current device is the NULL-device or no new device if there already is an open device. A function such as `x11` can also be used

filename Name of the file without extension

width Width of the plot, in inches

height Height of the plot, in inches

normalize Logical, should the plot be normalized to the plot size. If TRUE (default) border width, vertex size, edge width and arrow sizes are adjusted to look the same for all sizes of the plot, corresponding to what they would look in a 7 by 7 inches plot if normalize is FALSE.

DoNotPlot Runs qgraph but does not plot. Useful for saving the output (i.e. layout) without plotting

plot Logical. Should a new plot be made? Defaults to TRUE. Set to FALSE to add the graph to the existing plot.

rescale Logical. Defines if the layout should be rescaled to fit the -1 to 1 x and y area. Defaults to TRUE. Can best be used in combination with plot=FALSE.

standAlone Logical. If filetype="tex" this argument can be used to choose between making the output a standalone LaTeX file or only the codes to include the graph.

Graphical arguments

Nodes: These arguments influence the plotting of nodes in qgraph. Most of them can be assigned a single value or a vector with a value for each node.

color A vector with a color for each element in the groups list, or a color for each node. Defaults to the background color ("bg" argument, which defaults to "white") without groups list and rainbow(length(groups)) with a groups list.

vsize A value indicating the size of the nodes (horizontal if shape is "rectangle". Can also be a vector of length 2 (nodes are scaled to degree) or a size for each node. Defaults to 8*exp(-nNodes/80)+1

vsize2 A value indicating the vertical size of the nodes where the shape is "rectangle". Can also be a vector of length 2 (nodes are scaled to degree) or a size for each node. Defaults to the value of 'vsize'. If 'vsize' is not assigned this value is used as a scalar to 'vsize' (e.g., `vsize2 = 1/2` would result in rectangled nodes where the height is half the default width)

node.width Scalar on the default value of 'vsize'. Defaults to 1.

node.height Scalar on the default value of 'vsize2'. Defaults to 1.

borders Logical indicating if borders should be plotted, defaults to TRUE.

border.color Color vector indicating colors of the borders. Is repeated if length is equal to 1. Defaults to "black"

border.width Controls the width of the border. Defaults to 2 and is comparable to 'lwd' argument in 'points'.

shape A character containing the shape of the nodes. `"circle"`, `"square"`, `"triangle"` and `"diamond"` are supported. In addition, can be a name of an element of `polygonList` to plot the corresponding polygon (not reccomended for large graphs), which by default includes

shapes "ellipse" and "heart" Can also be a vector with a shape for each node. Defaults to "circle".

polygonList A list contaning named lists for each element to include polygons to lookup in the shape argument. Each element must be named as they are used in shape and contain a list with elements x and y contaning the coordinates of the polygon. By default ellipse and heart are added to this list. These polygons are scaled according to vsize and vsize2

vTrans Transparency of the nodes, must be an integer between 0 and 255, 255 indicating no transparency. Defaults to 255

subplots A list with as elements R expressions or NULL for each node. If it is an R expression it is evaluated to create a plot for the node.

subpars List of graphical parameters to be used in the subplots

subplotbg Background to be used in the sublots. If missing inherits from 'background' argument.

images A character vector of length 1 or the same length as the number of nodes indicating the file location of PNG or JPEG images to use as nodes. Can be NA to not plot an image as node and overwrites 'subplots'

noPar Set to TRUE to not have qgraph run the par function. Useful when sending qgraph plots as sublots using subplots.

pastel Logical, should default colors (for groups or edge equality constraints) be chosen from pastel colors? If TRUE then rainbow_hcl is used.

rainbowStart A number between 0 and 1 indicating the offset used in rainbow functions for default node coloring.

usePCH Logical indicating if nodes should be drawn using polygons or base R plotting symbols. Defaults to TRUE if more than 50 nodes are used in the graph or if the graph is stored in a file. See details.

node.resolution Resolution of the nodes if usePCH=FALSE. Defaults to 100

title String with a title to be drawn in the topleft of the plot.

title.cex Size of the title, defaults to 1.

preExpression A parsable string containing R codes to be evaluated after opening a plot and before drawing the graph.

postExpression A parsable string containing R codes to be evaluated just before closing the device.

diag Should the diagonal also be plotted as edges? defaults to FALSE. Can also be "col" to plot diagonal values as vertex colors.

Node labels: These arguments influence the plotting of node labels in qgraph. Most of them can be assigned a single value or a vector with a value for each node.

labels If FALSE, no labels are plotted. If TRUE, order in weights matrix is used as labels. This can also be a vector with a label for each node. Defaults for graphs with less than 20 nodes to a 3 character abbreviation of the columnames and rownames if these are identical or else to TRUE. If a label contains an asterisk (e.g. "x1*") then the asterisk will be omitted and the label will be printed in symbol font (use this for Greek letters). Can also be a list with a label as each element, which can be expressions for more advanced mathematical annotation.

label.cex Scalar on the label size.

label.color Character containing the color of the labels, defaults to "black"

label.prop Controls the proportion of the width of the node that the label rescales to. Defaults to 0. 9.

label.norm A single string that is used to normalize label size. If the width of the label is lower than the width of the hypothetical label given by this argument the width of label given by this argument is used instead. Defaults to "OOO" so that every label up to three characters has the same font size.

label.scale Logical indicating if labels should be scaled to fit the node. Defaults to TRUE.

label.scale.equal Logical, set to `TRUE` to make make the font size of all labels equal

label.font Integer specifying the label font of nodes. Can be a vector with value for each node

label.fill.vertical Scalar (0 - 1) indicating the maximum proportion a label may fill a node vertically.

label.fill.horizontal Scalar (0 - 1) indicating the maximum proportion a label may fill a node horizontally.

node.label.offset A vector of length two with the x and y offset coordinates of the node label (e.g., `c(0.5, 0.5)` is the default and centers the label with respect to the node area). The vector is passed to the `adj` argument of `graphics::text` function.

node.label.position Either a numeric vector of length 1 (i.e., it gets recycled) or of length equal to the number of nodes in the network, used to set the positions of the node labels. Takes values between 1 and 4 as follows: 1 - bottom; 2 - left; 3 - top; 4 - right. Overrides the `node.label.offset` argument and values are passed to the `pos` argument of `graphics::text` function. Defaults to `NULL`.

Edges: These arguments influence the plotting of edges qgraph. Most of them can be assigned a single value, a vector with a value per edge when an edgelist is used as input or a matrix containing values for each edge when a wheights matrix is used as input.

esize Size of the largest edge (or what it would be if there was an edge with weight maximum). Defaults to 15*exp(-nNodes/90)+1) for weighted graphs and 2 for unweighted graphs. In directed graphs these values are halved.

edge.width Scalar on 'esize' and 'asize' arguments to make edges wider with a single argument. 'esize' is multiplied with this value and 'asize' with the square root of this value.

edge.color Color of edges. Can be either a single value to make all edges the same color, a matrix with a color for each edge (when using a weights matrix) or a vector with a color for each edge (when using an edgelist). NA indicates that the default color should be used. Note that unless `fade=FALSE` colors still fade to white corresponding to their strength

posCol Color of positive edges. Can be a vector of two to indicate color of edges under 'cut' value and color of edges over 'cut' value. If 'fade is set to TRUE the first color will be faded the weaker the edge weight is. If this is only one element this color will also be used for edges stronger than the 'cut' value. Defaults to c("#009900","darkgreen")

negCol Color of negative edges. Can be a vector of two to indicate color of edges under 'cut' value and color of edges over 'cut' value. If 'fade is set to TRUE the first color will be faded the weaker the edge weight is. If this is only one element this color will also be used for edges stronger than the 'cut' value. Defaults to c("#BF0000","red")

unCol Color to indicate the default edge color of unweighted graphs. Defaults to "#808080".

probCol Color of the probability edges. Defaults to `"blue"`. Only used when `probabilityEdges = TRUE`

negDashed Logical, set to `TRUE` to make negative edges dashed (overwrites `lty`).

probabilityEdges Logical, do edges indicate probabilities? If this is set to `TRUE` `posCol` is overwritten by `probCol`. Mainly implemented for automatic generation of graphs

colFactor Exponent of transformation in color intensity of relative strength. Defaults to 1 for linear behavior.

trans In weighted graphs: logical indicating if the edges should fade to white (FALSE) or become more transparent (TRUE; use this only if you use a background). In directed graphs this is a value between 0 and 1 indicating the level of transparency. (also used as 'transparency')

fade if TRUE (default) and if 'edge.color' is assigned, transparency will be added to edges that are not transparent (or for which no transparency has been assigned) relative to the edge strength, similar if 'trans' is set to TRUE.

loopRotation A vector with an element for each node with either `NA` to let qgraph choose the rotation of the loop, or the rotation of the loop per node in radian

loop If diag=TRUE, this can be used to scale the size of the loop. defaults to 1.

lty Line type, see 'par'

edgeConnectPoints This argument specifies the point for each edge to which it connects to a node, in radians. Can be either a matrix with a row for each edge and two columns: The first column indicates the connection point of the source of the edge and the second column specifies the connection point of the destination of the edge. Can also be an array with a row and column for each node two slices which indicate the source and destination of the edge connecting the two nodes.

Edge Curvature: These arguments control the curvature of edges. Most of them can be assigned a single value, a vector with a value per edge when an edgelist is used as input or a matrix containing values for each edge when a wheights matrix is used as input.

curve A value indicating how strongly edges should be curved. Either a single value, a vector (edgelist input) with a value for each edge or a matrix (weights matrix input). NA indicates default curve behavior should be used, which only curves edges if there are multiple edges between two nodes.

curveAll Logical, indicating if all edges should be curved with the value of the 'curve' or only edges between nodes that have share multiple edges.

curveDefault The default curvature. Defaults to 1.

curveShape The shape of the curve, as used in `xspline`. Defaults to `-1`.

curveScale Logical, should curve scale with distance between nodes. Defaults to `TRUE`. If `FALSE`, the curve can be exactly determined. Recommended to set to `TRUE` for graphs and `FALSE` for diagrams. The curvature is corrected for the number of nodes and will be smaller if there are more nodes.

curveScaleNodeCorrection Logical, set to `FALSE` to disable the node correction in `curveScale`. Defaults to `TRUE`. Not recommended. Set to `FALSE` ONLY if you know what you are doing.

curvePivot Quantile to pivot curves on. This can be used to, rather than round edges, make straight edges as curves with "knicks" in them. Can be logical or numeric. `FALSE` (default) indicates no pivoting in the curved edges, a number indicates the quantile (and one minus this value as quantile) on which to pivot curved edges and `TRUE` indicates a value of 0.1.

curvePivotShape The shape of the curve around the pivots, as used in `xspline`. Defaults to `0.25`.

parallelEdge Logical, set to `TRUE` to draw parallel straight edges rather than curved edges when there are multiple edges between two nodes. Can be a vector with value per edge for edgelists or a matrix with a value per edge for weights marices.

parallelAngle The distance in radians an edge is shifted if `parallel=TRUE`. Can be set to `NA` (default) to determine based on number of edges between two nodes. Can be a vector with value per edge for edgelists or a matrix with a value per edge for weights marices.

parallelAngleDefault The default value for parallelAngle, indicating the angle of the edge furthest from the center. Defaults to `pi/6`

Edge Labels: These arguments influence the plotting of edge labels qgraph. Most of them can be assigned a single value, a vector with a value per edge when an edgelist is used as input or a matrix containing values for each edge when a weights matrix is used as input.

edge.labels If FALSE, no edge labels are plotted. If TRUE, numerical edge weights are printed on the edges. This can also be a vector with a label for each edge. Defaults to FALSE. If a label contains an asterisk (e.g. "y1*") then the asterisk will be omitted and the label will be printed in symbol font (use this for Greek letters). Can also be a list with a label as each element, which can be expressions for more advanced mathematical annotation.

edge.label.cex Either a single number or a number per edge used as a scalar of the edge label size. Defaults to 1.

edge.label.bg Either a logical or character vector/matrix. Indicates the background behind edge labels. If TRUE (default) a white background is plotted behind each edge label. If FALSE no background is plotted behind edge labels. Can also be a single color character, a vector or matrix of color vectors for each edge.

edge.label.margin Margin of the background box around the edge label. Defaults to zero.

edge.label.position Vector of numbers between 0 and 1 controlling the relative position of each edge label. Defaults to 0.5 for placing edge labels at the middle of the edge.

edge.label.font Integer specifying the label font of edges. Can be a vector or matrix with value for each node

edge.label.color Character vector indicating the color of the edge labels. It can be either a vector of length equal to the number of edges in the network or a single character color that will be applied to all edges.

Layout: Arguments concerning the placement of nodes, in combination with 'layout'.

repulsion Scalar on the default repulse radius in the spring layout. Defaults to 1. Setting this argument to lower values (e.g., 0.5) will cause nodes in the spring layout to repulse each other less. This is especially useful if a few unconnected nodes cause the giant component to visually be clustered too much in the same place.

layout.par A list of arguments passed to `qgraph.layout.fruchtermanreingold` when `layout = "spring"` or to an igraph function when such a function is assigned to 'layout'. Defaults to `list(repulse.rad = nNodes^(repulsion * 3))` if `layout = "spring"` and `list()` otherwise.

layoutRound Logical, should weights be rounded (default 10 digits) before computing layouts? This will hopefully make sure different machines result in the same layout. Defaults to `TRUE`

layout.control A scalar on the size of the circles created with the circular layout.

aspect Should the original aspect ratio be maintained if rescale = TRUE? Defaults to FALSE. Set this to TRUE to keep the aspect ratio of the original layout (e.g. result from layout="spring").

rotation A vector that can be used to rotate the circles created with the circular layout. Must contain the rotation in radian for each group of nodes. Defaults to zero for each group.

Legend: Arguments to control the legend placed on the right side of the graph.

legend Logical value indicating if a legend should be plotted. Defaults to TRUE if a groups object or nodeNames is supplied

legend.cex Scalar of the legend. defaults to 1

legend.mode Character string indicating the type of legend to be drawn. `"groups"` indicates the legend should be based on the `groups` object, `"names"` indicates the legend should be based on the `nodeNames` object, and `style1` and `style2` indicate the legend should be based on both. Defaults to `"style1"` if both `"groups"` and `"nodeNames"` arguments are used.

GLratio Relative size of the graph compared to the layout. Defaults to 2.5

layoutScale A vector with a scalar for respectively the x and y coordinates of the layout (which default plotting area is from -1 to 1 on both x and y axis). Setting this to e.g. c(2,2) would make the plot twice as big. Use this in combination with 'layoutOffset' and 'plot' arguments to define the graph placement on an existing plot.

layoutOffset A vector with the offset to the x and coordinates of the center of the graph (defaults to (0,0)). Use this in combination with 'layoutScale' and 'plot' arguments to define the graph placement on an existing plot.

nodeNames Names for each node, can be used to plot a legend next to the plot that links the node labels to node names.

Background: These arguments control the background of the plot

bg If this is TRUE, a background is plotted in which node colors cast a light of that color on a black background. Can also be a character containing the color of the background Defaults to FALSE

bgcontrol The higher this is, the less light each node gives if bg=TRUE. Defaults to 6.

bgres square root of the number of pixels used in bg=TRUE, defaults to 100.

General graphical arguments:

pty See 'par'

gray Logical, set to TRUE to plot the graph in grayscale colors

font Integer specifying the default font for node and edge labels

Arguments for directed graphs

directed Logical indicating if edges are directed or not. Can be TRUE or FALSE to indicate if all edges are directed, a logical vector (when using edgelists) or a logical matrix (when using weights matrix)

arrows A logical indicating if arrows should be drawn, or a number indicating how much arrows should be drawn on each edge. If this is TRUE, a simple arrow is plotted, if this is a number, arrows are put in the middle of the edges.

arrowAngle Angle of the arrowhead, in radians. Defaults to pi/8 for unweighted graphs and pi/4 for weighted graphs.

asize Size of the arrowhead. Defaults to 2*exp(-nNodes/20)+2.

open Logical indicating if open (TRUE) or closed (FALSE) arrowheads should be drawn.

bidirectional If this is TRUE, Then directional edges between nodes that have two edges between them are not curved. Defaults to FALSE. Can also be a logical vector (when using edgelists) or a logical matrix (when using weights matrix)

Arguments for graphs based on significance values

mode This argument defines the mode used for coloring the edges. The default, "strength" assumes each edge weight indicates the strength of connection centered around and makes positive edges green and negative edges red. If this is set to "sig" then the edge weights are assumed to be significance values and colored accordingly. This can also include negative values, which will be interpreted as p-values based on negative statistics.

alpha The significance level (defaults to 0.05) to be used for not showing edges if `minimum = "sig"`, or if `Graph = "sig"` a vector of max 4 elements indicating the alpha level cutoffs. Defaults to c(0.0001,0.001,0.01,0.05)

sigScale The function used to scale the edges if mode="sig". Defaults to $function(x)0.8*(1-x)^(log(0.4/0.8,1-0.05))$

bonf Logical indicating if a bonferonni correction should be applied if `minimum = "sig"` or `mode="sig"`

Arguments for plotting scores on nodes

scores This argument can be used to plot scores of an individual on the test. Should be a vector with the scores for each item. Currently this can only be integer values (e.g.\ LIKERT scales).

scores.range Vector of length two indicating the range of the scores, if scores is assigned.

Arguments for manually defining graphs

mode The mode argument (see section on significance graph arguments) can also be used to make the weights matrix correspond directly to the width of the edges (as in lwd of plot()). To do this, set mode to "direct".

edge.color This argument can be used to overwrite the colors. Can be either a single value to make all edges the same color, a matrix with a color for each edge (when using a weights matrix) or a vector with a color for each edge (when using an edgelist). NA indicates that the default color should be used. Note that unless `fade=FALSE` colors still fade to white corresponding to their strength

Arguments for knots (tying together edges)

knots This argument can be used to tie edges together in their center, which can be useful to, for example, indicate interaction effects. This argument can be assigned a list where each element is a vector containing the edge numbers that should be knotted together. Another option is to assign the argument a integer vector (for edgelists) or a matrix (for weight matrices) with 0 indicating edges that should not be tied together, and increasing numbers indicating each knot.

knot.size The size of the knots. Can be of length one or a vector with the size of each knot. Similar to 'vsize'. Defaults to 1.

knot.color The color of the knots. Can be of length one or a vector with the size of each knot. Defaults to NA, which will result in a mix of the knotted edge colors.

knot.borders Logical indicating if a border should be plotted around the knot. Can be of length one or a vector with the size of each knot. Works similar to 'borders'. Defaults to FALSE

knot.border.color Color of the knot borders. Can be of length one or a vector with the size of each knot. Works similar to 'border.color'. Defaults to "black"

knot.border.width Width of the knot borders. Can be of length one or a vector with the size of each knot. Works similar to 'border.width'. Defaults to 1

Arguments for bars

means A vector with means for every node or NA. Will plot a vertical bar at the location of the mean between meanRange values. NA omits a bar.

SDs A vector with SDs for every node or NA. Will plot an error bar of 2 times this value around the means location. NA to omit.

meanRange The range of the means argument. Default to range(means,na.rm=TRUE)

bars A list with for each node containing either NULL or a vector with values between 0 and 1 indicating where bars should be placed inside the node.

barSide Integer for each node indicating at which side the bars should be drawn. 1, 2, 3 or 4 indicating at bottom, left, top or right respectively.

barColor A vector with for each node indicating the color of bars. Defaults to the border color of the node.

barLength A Vector indicating the relative length of bars of each node compared to the node size. Defaults to 0.5.

barsAtSide Logical, should bars be drawn at the side of a node or at its center? Defaults to FALSE.

Arguments for pies

pie A vector with values between 0 and 1 for each node (or one value for all nodes). Supplying this argument will make the border of nodes a pie chart. Can also be a list with vectors to make pie charts of multiple parts.

pieBorder The size of the pie chart in the border, between 0 and 1. Defaults to 0.15. Set to 1 to make the whole node a pie chart. Can be a vector with a value for each node.

pieColor Colors of the pie plot parts. Can be a vector with a value for each node, or a list with multiple values if there are more parts.

pieColor2 Final color of the pie chart. Only added if the values in the 'pie' argument do not add up to 1. Defaults to 'white'. Can be a vector with a value for each node.

pieStart A vector with values between 0 and 1 for each node (or one value for all nodes), indicating the starting point of the pie chart.

pieDarken A vector with values between 0 and 1 for each node (or one value for all nodes), indicating how much darker the pie border color is made than the node color in the default coloring scheme.

piePastel Should pastel colors be used to fill pie chart parts when more than 2 blocks are used?

pieCImid A vector with values between 0 and 1 for each node (or one value for all nodes), indicating the center point of the confidence region. Overwrites the pie argument

pieCIlower A vector with values between 0 and 1 for each node (or one value for all nodes), indicating the lower bound of the confidence region. Overwrites the pie argument

pieCIupper A vector with values between 0 and 1 for each node (or one value for all nodes), indicating the upper bound of the confidence region. Overwrites the pie argument

pieCIpointcex A vector with values between 0 and 1 for each node (or one value for all nodes), indicating the size of the point estimate of the confidence region. Overwrites the pie argument. Defaults to 0.01.

pieCIpointcex A vector with values between 0 and 1 for each node (or one value for all nodes), indicating the color of the point estimate of the confidence region. Overwrites the pie argument. Defaults to "black".

Additional arguments

edgelist Logical, if TRUE 'input' is assumed to be an edgelist, else if FALSE input is assumed to be a weights matrix. By default this is chosen automatically based on the dimensions of 'input' and this argument is only needed if the dimensions are ambiguous (square matrix with 2 or 3 rows/columns)

weighted Logical that can be used to force either a weighted graph (TRUE) or an unweighted graph(FALSE).

nNodes The number of nodes, only needs to be specified if the first argument is an edge-list and some nodes have no edges

XKCD If set to TRUE the graph is plotted in XKCD style based on http://stackoverflow.com/a/12680841/567015.

Using qgraph to plot graphs

The first argument of qgraph(), 'input', is the input. This can be a number of objects but is mainly either a weights matrix or an edgelist. Here we will assume a graph is made of n nodes connected by m edges. qgraph is mainly aimed at visualizing (statistical) relationships between variables as weighted edges. In these edge weights a zero indicates no connection and negative values are comparable in strength to positive values. Many (standardized) statistics follow these rules, the most important example being correlations. In the special case where all edge weights are either 0 or 1 the weights matrix is interpreted as an adjacency matrix and an unweighted graph is made.

a weights matrix is a square n by n matrix in which each row and column represents a node. The element at row i and column j indicates the connection from node i to node j. If the weights matrix is symmetrical an undirected graph is made and if the matrix is asymmetrical a directed graph is made.

Alternatively an edgelist can be used. This is a m by 2 matrix (not a list!) in which each row indicates an edge. The first column indicates the number of the start of the edge and the second column indicates the number of the end of the edge. The number of each node is a unique integer between 1 and n. The total number of nodes will be estimated by taking the highest value of the edgelist. If this is incorrect (there are nodes with no edges beyond the ones already specified) the 'nNodes' argument can be used. If an integer between 1 and n is missing in the edgelist it is assumed to be a node with no edges. To create a weighted graph edge weights can be added as a third column in the edgelist. By default using an edgelist creates a directed graph, but this can be set with the 'directed' argument.

Interpreting graphs

In weighted graphs green edges indicate positive weights and red edges indicate negative weights. The color saturation and the width of the edges corresponds to the absolute weight and scale relative to the strongest weight in the graph. It is possible to set this strongest edge by using the 'maximum' argument. When 'maximum' is set to a value above any absolute weight in the graph that value is considered the strongest edge (this must be done to compare different graphs; a good value for correlations is 1). Edges with an absolute value under the 'minimum' argument are omitted (useful to keep filesizes from inflating in very large graphs).

In larger graphs the above edge settings can become hard to interpret. With the 'cut' argument a cutoff value can be set which splits scaling of color and width. This makes the graphs much easier to interpret as you can see important edges and general trends in the same picture. Edges with absolute weights under the cutoff score will have the smallest width and become more colorful as

they approach the cutoff score, and edges with absolute weights over the cutoff score will be full red or green and become wider the stronger they are.

Specifying the layout

The placement of the nodes (i.e. the layout) is specified with the 'layout' argument. It can be manually specified by entering a matrix for this argument. The matrix must have a row for each node and two columns indicating its X and Y coordinate respectively. qgraph plots the nodes on a (-1:1)(-1:1) plane, and the given coordinates will be rescaled to fit this plane unless 'rescale' is FALSE (not recommended). Another option to manually specify the layout is by entering a matrix with more then two columns. This matrix must then consist of zeroes and a number (the order in the weights matrix) for each node indicating it's place. For example:

0 0 2 0 0

1 0 3 0 4

will place node 2 at the top in the center, node 1 at the bottom left corner, node 3 at the bottom in the center and node 4 at the bottom right corner. It is recommended however that one of the integrated layouts is used. 'layout' can be given a character as argument to accomplish that. layout="circular" will simply place all nodes in a circle if the groups argument is not used and in separate circles per group if the groups argument is used (see next section).

The circular layout is convenient to see how well the data conforms to a model, but to show how the data clusters another layout is more appropriate. By specifying layout="spring" the Fruchterman-reingold algorithm (Fruchterman & Reingold, 1991), which has been ported from the SNA package (Butts, 2010), can be used to create a force-directed layout. In principle, what this function does is that each node (connected and unconnected) repulse each other, and connected nodes also attract each other. Then after a number of iterations (500 by default) in which the maximum displacement of each node becomes smaller a layout is achieved in which the distance between nodes correspond very well to the absolute edge weight between those nodes.

A solution to use this function for weighted graphs has been taken from the igraph package (Csardi G & Nepusz T, 2006) in which the same function was ported from the SNA package. New in qgraph are the option to include constraints on the nodes by fixing a coordinate for nodes or reducing the maximum allowed displacement per node. This can be done with the 'layout.par' argument. For more information see `qgraph.layout.fruchtermanreingold`.

By default, 'layout' is set to "spring" for unweighted and directed graphs and "circular" otherwise.

Grouping nodes

Grouping nodes (e.g., according to a measurement model) can be specified with the 'groups' argument. This can be a factor or a list in which each element is a vector containing the numbers of nodes that belong together (numbers are taken from the order in the weights matrix). All numbers must be included. If a groups list is specified the "groups" layout can be used to place these nodes together, the nodes in each group will be given a color, and a legend can be plotted (by setting 'legend' to TRUE). The colors will be taken from the 'color' argument, or be generated with the `rainbow` function.

Output

By default qgraph will plot the graph in a new R window. However the graphs are optimized to be plotted in a PDF file. To easily create a pdf file set the 'filetype' argument to "pdf". 'filename' can

be used to specify the filename and folder to output in. 'height' and 'width' can be used to specify the height and width of the image in inches. By default a new R window is opened if the current device is the NULL-device, otherwise the current device is used (note that when doing this 'width' and 'height' still optimize the image for those widths and heights, even though the output screen size isn't affected, this is especially important for directed graphs!).

Furthermore filetype can also be set to numerous other values. Alternatively any output device in R can be used by simply opening the device before calling qgraph and closing it with dev.off() after calling qgraph.

IMPORTANT NOTE: graphs made in qgraph must be exported programatically using device functions such as pdf() and png(). Manually resizing a graph and using export functions such as the one built into RStudio will give UNSTABLE RESULTS.

Manual specification of color and width

In qgraph the widths and colors of each edge can also be manually controlled. To directly specify the width of each edge set the 'mode" argument to "direct". This will then use the absolute edge weights as the width of each edge (negative values can still be used to make red edges). To manually set the color of each edge, set the 'edge.color' argument to a matrix with colors for each edge (when using a weights matrix) or a vector with a color for each edge (when using an edgelist).

Replotting graphs and reusing layouts

If the result of `qgraph` is stored, such as `Graph <- qgraph(...)`, the plot can be recreated in two ways. `qgraph(Graph, ...))` reruns `qgraph` with the same arguments used in the origina call except those restated in the dots. For example `qgraph(Graph, shape = "square")` will recreate the same plot but now use square nodes instead of circular. `plot(Graph)` will NOT rerun `qgraph` but simply plot the qgraph object. This means that now specific graph attributes can be changed before plotting.

More specific, `qgraph(Graph)` will base the new plot only on the `Arguments` element of the `qgraph` object and `plot(qgraph)` will base the new plot on the `graphAttributes` and `plotOptions` elements of the qgraph object.

To reuse a layout, use the `layout` element. e.g., to plot a new graph with the same layout use `qgraph(..., layout = Graph$layout)`

Additional information

By default, edges will be straight between two nodes unless there are two edges between two nodes. To overwrite this behavior the 'bidirectional' argument can be set to TRUE, which will turn two edges between two nodes into one bidirectional edge. 'bidirectional' can also be a vector with TRUE or FALSE for each edge.

To specify the strength of the curve the argument 'curve' can be used (but only in directional graphs). 'curve' must be given a numerical value that represent an offset from the middle of the straight edge through where the curved edge must be drawn. 0 indicates no curve, and any other value indicates a curve of that strength. A value of 0.3 is recommended for nice curves. This can be either one number or a vector with the curve of each edge.

Nodes and edges can be given labels with the 'labels' and the 'edge.labels' arguments. 'labels' can be set to FALSE to omit labels, TRUE (default) to set labels equal to the node number (order in the weights matrix) or it can be a vector with the label for each node. Edge labels can also be set to

FALSE to be omitted (default). If 'edge.labels' is TRUE then the weight of each label is printed. Finally, 'edge.labels' can also be a vector with the label for each edge. If a label (both for edges and nodes) contain an asterisk then the asterisk is omitted and that label is printed in the symbol font (useful to print Greek letters).

A final two things to try: the 'scores' argument can be given a vector with the scores of a person on each variable, which will then be shown using colors of the nodes, And the 'bg' argument can be used to change the background of the graph to another color, or use bg=TRUE for a special background (do set transparency=TRUE when using background colors other then white).

Debugging

If this function crashes for any reason with the filetype argument specified, run:

dev.off()

To shut down the output device!

Author(s)

Sacha Epskamp <mail@sachaepskamp.com>

References

Carter T. Butts <buttsc@uci.edu> (2010). sna: Tools for Social Network Analysis. R package version 2.2-0. http://CRAN.R-project.org/package=sna

Csardi G, Nepusz T (2006). The igraph software package for complex network research, InterJournal, Complex Systems 1695. http://igraph.sf.net

Sacha Epskamp, Angelique O. J. Cramer, Lourens J. Waldorp, Verena D. Schmittmann, Denny Borsboom (2012). qgraph: Network Visualizations of Relationships in Psychometric Data. Journal of Statistical Software, 48(4), 1-18. URL http://www.jstatsoft.org/v48/i04/.

Jerome Friedman, Trevor Hastie and Rob Tibshirani (2011). glasso: Graphical lasso-estimation of Gaussian graphical models. R package version 1.7. http://CRAN.R-project.org/package=glasso

Bernd Klaus and Korbinian Strimmer. (2014). fdrtool: Estimation of (Local) False Discovery Rates and Higher Criticism. R package version 1.2.12. http://CRAN.R-project.org/package=

Fruchterman, T. & Reingold, E. (1991). Graph drawing by force-directed placement. Software - Pract. Exp. 21, 1129-1164.

N. Kraemer, J. Schaefer. A.-L. Boulesteix (2009). Regularized Estimation of Large-Scale Gene Regulatory Networks using Gaussian Graphical Models BMC Bioinformatics 10:384

Plate, T. <tplate@acm.org> and based on RSvgDevice by T Jake Luciani <jakeluciani@yahoo.com> (2009). RSVGTipsDevice: An R SVG graphics device with dynamic tips and hyperlinks. R package version 1.0-1.

Revelle, W. (2014) psych: Procedures for Personality and Psychological Research, Northwestern University, Evanston, Illinois, USA, http://CRAN.R-project.org/package=psych Version = 1.4.4.

See Also

cor_auto qgraph.animate qgraph.loadings

Examples

```
## Not run:
### Correlations ###
# Load big5 dataset:
data(big5)
data(big5groups)

# Compute correlation matrix:
big5_cors <- cor_auto(big5, detectOrdinal = FALSE)

# Correlations:
big5Graph <- qgraph(cor(big5),minimum=0.25,groups=big5groups,
            legend=TRUE,borders=FALSE, title = "Big 5 correlations")


  # Same graph with spring layout:
  qgraph(big5Graph,layout="spring")

  # Same graph with different color scheme:
  qgraph(big5Graph,posCol="blue",negCol="purple")

  ### Network analysis ###
  ### Using bfi dataset from psych ###
  library("psych")
  data(bfi)

  # Compute correlations:
  CorMat <- cor_auto(bfi[,1:25])

  # Compute graph with tuning = 0 (BIC):
  BICgraph <- qgraph(CorMat, graph = "glasso", sampleSize = nrow(bfi),
            tuning = 0, layout = "spring", title = "BIC", details = TRUE)

  # Compute graph with tuning = 0.5 (EBIC)
  EBICgraph <- qgraph(CorMat, graph = "glasso", sampleSize = nrow(bfi),
            tuning = 0.5, layout = "spring", title = "BIC", details = TRUE)

  # Compare centrality and clustering:
  centralityPlot(list(BIC = BICgraph, EBIC = EBICgraph))
  clusteringPlot(list(BIC = BICgraph, EBIC = EBICgraph))

  # Compute centrality and clustering:
  centrality_auto(BICgraph)
  clustcoef_auto(BICgraph)

  ### Directed unweighted graphs ###
  set.seed(1)
  adj=matrix(sample(0:1,10^2,TRUE,prob=c(0.8,0.2)),nrow=10,ncol=10)
  qgraph(adj)
  title("Unweighted and directed graphs",line=2.5)

  # Save plot to nonsquare pdf file:
```

```
qgraph(adj,filetype='pdf',height=5,width=10)

#### EXAMPLES FOR EDGES UNDER DIFFERENT ARGUMENTS ###
# Create edgelist:
dat.3 <- matrix(c(1:15*2-1,1:15*2),,2)
dat.3 <- cbind(dat.3,round(seq(-0.7,0.7,length=15),1))

# Create grid layout:
L.3 <- matrix(1:30,nrow=2)

# Different esize:
qgraph(dat.3,layout=L.3,directed=FALSE,edge.labels=TRUE,esize=14)

# Different esize, strongest edges omitted (note how 0.4 edge is now
# just as wide as 0.7 edge in previous graph):
qgraph(dat.3[-c(1:3,13:15),],layout=L.3,nNodes=30,directed=FALSE,
       edge.labels=TRUE,esize=14)

# Different esize, with maximum:
qgraph(dat.3,layout=L.3,directed=FALSE,edge.labels=TRUE,esize=14,maximum=1)
title("maximum=1",line=2.5)

qgraph(dat.3[-c(1:3,13:15),],layout=L.3,nNodes=30,directed=FALSE,edge.labels=TRUE,
       esize=14,maximum=1)
title("maximum=1",line=2.5)

# Different minimum
qgraph(dat.3,layout=L.3,directed=FALSE,edge.labels=TRUE,esize=14,minimum=0.1)
title("minimum=0.1",line=2.5)

# With cutoff score:
qgraph(dat.3,layout=L.3,directed=FALSE,edge.labels=TRUE,esize=14,cut=0.4)
title("cut=0.4",line=2.5)

# With details:
qgraph(dat.3,layout=L.3,directed=FALSE,edge.labels=TRUE,esize=14,minimum=0.1,
       maximum=1,cut=0.4,details=TRUE)
title("details=TRUE",line=2.5)


# Trivial example of manually specifying edge color and widths:
E <- as.matrix(data.frame(from=rep(1:3,each=3),to=rep(1:3,3),width=1:9))
qgraph(E,mode="direct",edge.color=rainbow(9))


### Input based on other R objects ###


## pcalg
# Example from pcalg vignette:
library("pcalg")
data(gmI)
suffStat <- list(C = cor(gmI$x), n = nrow(gmI$x))
```

```
  pc.fit <- pc(suffStat, indepTest=gaussCItest,
               p = ncol(gmI$x), alpha = 0.01)

  qgraph(pc.fit)

  ## glasso:
  # Using bfi dataset from psych:
  library("psych")
  data(bfi)
  cor_bfi <- cor_auto(bfi[,1:25])

  # Run qgraph:
  library("glasso")
  bfi_glasso <- glasso(cor_bfi, 0.1)

  # Plot:
  qgraph(bfi_glasso, layout = "spring")


## End(Not run)
```

`qgraph.animate` *Animate a growing network*

Description

This function is meant to facilitate the creation of animations based on growing networks. Networks are created based on the Fruchterman Reingold algorithm, which is constraint by limiting the maximum displacement of nodes that are already in the graph.

Usage

```
qgraph.animate(input, ind = NULL, ..., constraint = 10, growth = "order",
  titles = NULL, sleep = 0, smooth = TRUE, plotGraphs = TRUE, progress = TRUE,
  initLayout)
```

Arguments

`input`	A weights matrix of the graph or a list of weigths matrices with different weights of the same graph (see details). See `qgraph`. Edgelists are currently not supported.
`ind`	An object that specifies which nodes ar included or excluded. See details.
`...`	Additional arguments sent to `qgraph`
`constraint`	The constraint factor of included nodes. See details. Defaults to 10 for an soft-constrained animation. Set to Inf for a hard-constrained animation.

growth	The way nodes are added by default. Set to "order" to include nodes in the order they appear in the weigths matrix and to "degree" to include nodes based on their degree (high degree first)
titles	Optional vector with a title for each plot
sleep	Optional value sent to Sys.sleep() for showing the animation in R
smooth	Logical. If set to TRUE smoothing via loess is performed on the layout of all frames.
plotGraphs	Logical. If set to FALSE graphs are not plotted.
progress	Logical. If set to TRUE progress bars are included.
initLayout	An optional n by 2 matrix containing the initial placement of nodes in the animation.

Details

Let n be the number of nodes in total in the graph.

This function is designed to facilitate the production of animations by constraining the Fruchterman Reingold algorithm. Several frames are plotted of (a subset of) the same graph. If a node was already in the graph its maximum displacement per iteration of Fruchterman Reingold is equal to the number of nodes times the inverse of the constraint argument (so by default n/10). The higher this constraint value the stricter nodes stay in the same place between plots.

How many and which plots are made are defined by the 'input' and 'ind' arguments. There are two ways to specify the 'input' argument, either by speficying one weigths matrix or by specifying a list of weights matrices. In the sections below is explained what both of these methods do and how they are used.

This function, since it can be seen as an expression that makes several plots, works well in combination with the animation package for saving the animation to a wide variety of filetypes.

Value

Invisibly returns a list of all graphs.

Single weigths matrix

If 'input' is a single weigths matrix then in each frame a subset of the same graph is plotted. This is especially usefull for animating the growth of a network. Which nodes are in each frame is determined by the 'ind' argument.

If 'int' is not specified an animation is created in which in each frame a single node is added. This node is either in order of apearance in the weigths matrix or by its degree, which is determined with the 'growth' argument.

If 'ind' is a logical vector of length n than the first frame will contain the nodes specified with this vector and all other frames will grow in the same way as explained above (each step one node is added).

If 'ind' is a numeric vector of length n which contains all integers between 1 and n (a single entry per node) then the first frame starts with only the node specified in the first element of the vector and in frame i the ith element is added (each step one node is added).

If 'ind' is a list with numeric vectors as elements containing integers between 1 and n then in frame i the nodes from the ith element of the list will be added. Node numbers that occur multiple times in the list are ignored (they are already added the first time).

Finally, if 'ind' is a logical matrix with n columns and an arbitrary amount of rows, then in frame i only the nodes that are TRUE in row i are included. This is the only way to specify removal of nodes.

List of weigths matrices

The 'input' argument can also be given a list of weigths matrices if all these matrices have the same dimension (i.e.\ only the weights differ). If this is done than in frame i the ith weigths matrix is used. This is especially usefull for animating the change in a graph.

In this case, the 'ind' argument behaves differently. If this argument is not specified then in each frame all nodes are included.

If 'ind' is a logical vector of length n then only one plot is made with the nodes specified with that vector, and only if the length of 'input' is one.

Other methods woth in the same way as above. However, if the 'ind' argument indicates a different number of frames than the 'input' argument the function will stop and give an error.

Author(s)

Sacha Epskamp (mail@sachaepskamp.com)

References

Sacha Epskamp, Angelique O. J. Cramer, Lourens J. Waldorp, Verena D. Schmittmann, Denny Borsboom (2012). qgraph: Network Visualizations of Relationships in Psychometric Data. Journal of Statistical Software, 48(4), 1-18. URL http://www.jstatsoft.org/v48/i04/.

See Also

qgraph

Examples

```
## Not run:

## For these examples, first generate a scale free network using preferential attachment:

# Number of nodes:
n <- 100
# Empty vector with Degrees:
Degs <- rep(0, n)
# Empty Edgelist:
E <- matrix(NA, n - 1, 2)
# Add and connect nodes 1 and 2:
E[1, ] <- 1:2
Degs[1:2] <- 1
# For each node, add it with probability proportional to degree:
for (i in 2:(n - 1))
```

```
{
E[i, 2] <- i + 1
con <- sample(1:i, 1, prob = Degs[1:i]/sum(Degs[1:i]),i)
Degs[c(con,i+1)] <- Degs[c(con,i+1)] + 1
E[i, 1] <- con
}

# Because this is an edgelist we need a function to convert this to an adjacency matrix:
E2adj <- function(E,n)
{
  adj <- matrix(0,n,n)
  for (i in 1:nrow(E))
  {
      adj[E[i,1],E[i,2]]  <- 1
  }
  adj <- adj + t(adj)
  return(adj)
}



### EXAMPLE 1: Animation of construction algorithm: ###
adjs <- lapply(1:nrow(E),function(i) E2adj(E[1:i,,drop=FALSE],n))
qgraph.animate(adjs,color="black",labels=FALSE,sleep=0.1, smooth = FALSE)
rm(adjs)

### EXAMPLE 2: Add nodes by final degree: ###
adj <- E2adj(E,n)
qgraph.animate(E2adj(E,n),color="black",labels=FALSE,constraint=100,sleep=0.1)

### EXAMPLE 3: Changing edge weights: ###
adjW <- adj*rnorm(n^2)
adjW <- (adjW + t(adjW))/2
adjs <- list(adjW)
for (i in 2:100)
{
  adjW <- adj*rnorm(n^2)
  adjW <- (adjW + t(adjW))/2
  adjs[[i]] <- adjs[[i-1]] + adjW
}
qgraph.animate(adjs,color="black",labels=FALSE,constraint=100,sleep=0.1)



## End(Not run)
```

---

qgraph.layout.fruchtermanreingold

qgraph.layout.fruchtermanreingold

Description

This is a wrapper for the function that returns the x and y coordinates of the graph based on the Fruchterman Reingold algorithm (Fruchterman & Reingold, 1991), which was ported from the SNA package (Butts, 2010). This function is used in qgraph and is not designed to be used separately. See details for using constraints in this layout.

Usage

```
qgraph.layout.fruchtermanreingold(edgelist, weights=NULL, vcount=NULL,
niter=NULL, max.delta=NULL, area=NULL, cool.exp=NULL, repulse.rad=NULL,
init=NULL, groups=NULL, rotation=NULL, layout.control=0.5, constraints=NULL,
round = TRUE, digits = NULL)
```

Arguments

edgelist	A matrix with on each row the nodes at the start and the node at the end of each edge.
weights	A vector containing the edge weights.
vcount	The number of nodes.
niter	Number of iterations, default is 500.
max.delta	Maximum displacement, default is equal to the number of nodes.
area	The area of the plot, default is the square of the number of nodes.
cool.exp	Cooling exponent, default is 1.5.
repulse.rad	Repulse radius, defaults to the cube of the number of nodes.
init	Matrix with two columns and a row for each node containing the initial X and Y positions.
groups	See qgraph
rotation	See qgraph
layout.control	See qgraph
constraints	A constraints matrix with two columns and a row for each node containing a NA if the node is free or a fixed value for one of the coordinates.
round	Logical indicating if the initial input should be rounded
digits	Number of digits to round initial input and displacement in the algorithm to. Defaults to 5. This helps prevent floating point disrepancies between different operating systems.

Details

All arguments for this function can be passed from qgraph to this function by using the 'layout.par' argument, which must be a list containing the arguments. This can be used to constrain the layout in two ways:

Hard constraints

By using the 'constraints' argument the X and Y positions of each node can be fixed to a certain value. The 'constraint' argument must be given a matrix with two columns and a row for each node. An NA means that that coordinate for that node is free, and a value means it is fixed to that value.

Soft constraints

Soft constraining can be done by varying the 'max.delta' argument. This can be a single number, but also a vector containing the maximum displacement per step for each node. The default value is the number of nodes, so by setting this to a lower value for some nodes the node won't move so much. Use this in combination with the 'init' argument to make sure nodes don't move too much from their initial setup. This can be useful when adding a new node to an existing network and if you don't want the network to completely change.

Author(s)

Sacha Epskamp (mail@sachaepskamp.com)

References

Sacha Epskamp, Angelique O. J. Cramer, Lourens J. Waldorp, Verena D. Schmittmann, Denny Borsboom (2012). qgraph: Network Visualizations of Relationships in Psychometric Data. Journal of Statistical Software, 48(4), 1-18. URL http://www.jstatsoft.org/v48/i04/.

Carter T. Butts <buttsc@uci.edu> (2010). sna: Tools for Social Network Analysis. R package version 2.2-0. http://CRAN.R-project.org/package=sna

Fruchterman, T. & Reingold, E. (1991). Graph drawing by force-directed placement. Software - Pract. Exp. 21, 1129?1164.

See Also

qgraph

Examples

```
## Not run:
# This example makes a multipage PDF that contains images
# Of a building network using soft constraints.

# Each step one node is added with one edge. The max.delta
# decreases the longer nodes are present in the network.

pdf("Soft Constraints.pdf",width=10,height=5)

adj=adjO=matrix(0,nrow=3,ncol=3)
adj[upper.tri(adj)]=1
Q=qgraph(adj,vsize=3,height=5,width=10,layout="spring",
esize=1,filetype='',directed=T)
cons=Q$layout
for (i in 1:20)
{
```

```
x=nrow(adj)
adjN=matrix(0,nrow=x+1,ncol=x+1)
adjN[1:x,1:x]=adj
consN=matrix(NA,nrow=x+1,ncol=2)
consN[1:x,]=cons[1:x,]
layout.par=list(init=rbind(cons,c(0,0)),
max.delta=10/(x+1):1,area=10^2,repulse.rad=10^3)
y=sample(c(x,sample(1:(x),1)),1)
adjN[y,x+1]=1
Q=qgraph(adjN,Q,layout="spring",layout.par=layout.par)
cons=Q$layout
adj=adjN
}
dev.off()

## End(Not run)
```

`qgraph.loadings` *qgraph.loadings*

Description

This function is a wrapper function for `qgraph` designed to visualize factor loadings.

Usage

```
qgraph.loadings( fact, ...)
```

Arguments

`fact`	A matrix containing factor loadings (items per row, factors per column) or an "loadings" object
`...`	Additional optional arguments passed to `qgraph` and special arguments used in this function (described below).

Additional optional arguments

layout If "default" a standard layout for factor models will be made. If this is "circle" the default layout is circled (factors in the centre, items at the edge). No other layouts are currently supported.

vsize A vector where the first value indicates the size of manifest variables and the second value indicates the size of latent variables.

model "reflective" to have arrows go to manifest variables, "formative" to have arrows go to latent variables or "none" (default) for no arrows

crossloadings Logical, if TRUE then for each manifest variable the strongest loading is omitted (default to FALSE).

groups An optional list containing the measurement model, see `qgraph`

Fname When there is only one factor, this is it's name. If there are more factors, the names in the groups list are used only if the factors can be identified.

resid Values for the residuals

residSize Size of the residuals, defaults to 0.1

factorCors Correlation matrix of the factors

Author(s)

Sacha Epskamp (mail@sachaepskamp.com)

References

Sacha Epskamp, Angelique O. J. Cramer, Lourens J. Waldorp, Verena D. Schmittmann, Denny Borsboom (2012). qgraph: Network Visualizations of Relationships in Psychometric Data. Journal of Statistical Software, 48(4), 1-18. URL http://www.jstatsoft.org/v48/i04/.

See Also

qgraph

Examples

```
## Not run:
# Load big5 dataset:
data(big5)
data(big5groups)

big5efa <- factanal(big5,factors=5,rotation="promax",scores="regression")
big5loadings <- loadings(big5efa)
qgraph.loadings(big5loadings,groups=big5groups,minimum=0.2,
cut=0.4,vsize=c(1.5,15),borders=FALSE,vTrans=200,
model = "reflective", resid = big5efa$uniquenesses)

# Tree layout:
qgraph.loadings(big5loadings,groups=big5groups,minimum=0.2,
cut=0.4,vsize=c(1.5,15),borders=FALSE,vTrans=200,
layout="tree",width=20,model = "reflective",
resid = big5efa$uniquenesses)

## End(Not run)
```

---

qgraphMixed *Plots a mixed graph with both directed and undirected edges.*

Description

This function can be used to plot a network in which each node is connected by at most 3 edges; one undirected edge and two directed edges.

Usage

```
qgraphMixed(undirected, directed, parallel = TRUE, parallelAngle = pi/6,
     diagUndirected = FALSE, diagDirected = TRUE, ltyUndirected = 1, ltyDirected = 1,
        curve = 1, ...)
```

Arguments

undirected	The undirected network weights matrix.
directed	The directed network weights matrix.
parallel	Logical indicating if edges should be plotted parallel or curved.
parallelAngle	See qgraph
diagUndirected	Logical indicating if the diagonal of the undirected edges should be included.
diagDirected	Logical indicating if the diagonal of the directed edges should be included.
ltyUndirected	lty of undirected edges
ltyDirected	lty of directed edges
curve	Curvature of directed edges
...	Arguments sent to qgraph

Author(s)

Sacha Epskamp <mail@sachaepskamp.com>

---

smallworldIndex *Small-world index of unweighted graph*

Description

Computes the small-world index of an unweighted graph. When the graph is weighted, weights are removed and every nonzero edge weight is set to 1.

Usage

```
smallworldIndex(x)
```

Arguments

x	A qgraph object.

Author(s)

Sacha Epskamp <mail@sachaepskamp.com>

References

Watts, D. J., & Strogatz, S. H. (1998). Collective dynamics of 'small-world' networks. nature, 393(6684), 440-442.

`smallworldness` *Compute the small-worldness index.*

Description

Compute the small-worldness index (Humphries & Gurney, 2008) relying on the global transitity of the network (Newman, 2003) and on its average shortest path length.

Usage

```
smallworldness(x, B = 1000, up = 0.995, lo = 0.005)
```

Arguments

`x`	A graph. Can be a `qgraph` object object, an `igraph` object, an adjacency matrix, a weight matrix and an edgelist, or a weighted edgelist.
`B`	The number of random networks.
`up`	The upper quantile.
`lo`	the lower quantile.

Details

The function computes the transitivity of the target network and the average shortest path length. Then it computes the average of the same indices on `B` random networks. The small-worldness index is then computed as the transitivity (normalized by the random transitivity) over the average shortest path length (normalized by the random average shortest path length). The `lo` and `up` quantiles of the distribution of the random networks are also returned for both the transitivity and the average shortest path length.

A network can be said "smallworld" if its `smallworldness` is higher than one (a stricter rule is `smallworldness>=3`; Humphries & Gurney, 2008). To consider a network as "smallworld", it is also suggested to inspect that the network has a transitivity substantially higher than comparable random networks and that its average shortest path length is similar or higher (but not many times higher) than that computed on random networks. Edge weights, signs and directions are ignored in the computation of the indices.

Value

`smallworldness`	the "small-worldness" index proposed by Humphries & Gurney (2008)
`trans_target`	the global transitivity of the target network (Newman, 2003)
`averagelength_target`	the average shortest path length in the target network
`trans_rnd_M`	the average transitivity in the `B` random networks
`trans_rnd_lo`	the `lo` quantile of the transitivity in the `B` random networks
`trans_rnd_up`	the `up` quantile of the transitivity in the `B` random networks

averagelength_rnd_M
: the average shortest path length in the B random networks

averagelength_rnd_lo
: the lo quantile of the shortest path length in the B random networks

averagelength_rnd_up
: the up quantile of the shortest path length in the B random networks

Note

If a directed network is given as input, an edge between every two nodes i and j is considered present if there is an arrow either from i to j or from j to i or both.

Author(s)

Giulio Costantini (giulio.costantini@unimib.it), Sacha Epskamp (mail@sachaepskamp.com)

References

Costantini, G., Epskamp, S., Borsboom, D., Perugini, M., Mottus, R., Waldorp, L., Cramer, A. O. J., State of the aRt personality research: A tutorial on network analysis of personality data in R. Manuscript submitted for publication.

Humphries, M. D., & Gurney, K. (2008). Network "small-world-ness": a quantitative method for determining canonical network equivalence. PLoS One, 3(4), e0002051.

Newman, M. E. J. (2003). The structure and function of complex networks*. SIAM Review, 45(3), 167–256.

Examples

```
set.seed(1)
# a regular lattice. Even if the small-worldness is higher than three, the average path length is
# much higher than that of random networks.
regnet<-igraph::watts.strogatz.game(dim=1, size=1000, nei=10, p=0, loops=FALSE, multiple=FALSE)
smallworldness(regnet, B=10)

## Not run:
# a small-world network: the transitivity is much higher than random, the average path length is
# close to that of random networks
swnet<-igraph::watts.strogatz.game(dim=1, size=1000, nei=10, p=.1, loops=FALSE, multiple=FALSE)
smallworldness(swnet, B=10)

# a pseudorandom network: both the average path length and the transitivity are similar to random
# networks.
rndnet<-igraph::watts.strogatz.game(dim=1, size=1000, nei=10, p=1, loops=FALSE, multiple=FALSE)
smallworldness(rndnet, B=10)

## End(Not run)
```

summary.qgraph *Summary method for "qgraph"*

Description

This function creates a brief summary based on a "qgraph" object.

Usage

```
## S3 method for class 'qgraph'
summary(object, ...)
```

Arguments

object	A "qgraph" object
...	These arguments are not used

Author(s)

Sacha Epskamp (mail@sachaepskamp.com)

See Also

qgraph

VARglm *Computes a vector autoregressive lag-1 model using GLM*

Description

This function computes a VAR model using glm.

Usage

```
VARglm(x, family, vars, adjacency, icfun = BIC, ...)
```

Arguments

x	A data frame
family	The family to be used. Defaults to gaussian if data is continuous or binomial if data is binary
vars	Vector of variables to predict. If missing all variables are predicted.
adjacency	Adjacency matrix. If missing full network is estimated
icfun	Information criterium function to be included in the output
...	Arguments used in the icfun

Value

A list containing:

graph	The estimated graph
IC	The information criterium

Author(s)

Sacha Epskamp <mail@sachaepskamp.com

---

wi2net *Converts precision matrix to partial correlation matrix*

Description

A small function that converts a precision matrix (inverse of covariance matrix) to a partial correlatin matrix. This can be done by standardizing the precision matrix and changing the sign of the offdiagonal entries. Many methods exist for obtaining a precision matrix (Such as the glasso package; Friedman, Hastie and Tibshirani, 2011) but the partial correlation matrix is easier interpretable and better usuable in qgraph.

Usage

```
wi2net(x)
```

Arguments

x	A precision matrix

Value

A partial correlation matrix

Author(s)

Sacha Epskamp <mail@sachaepskamp.com>

References

Jerome Friedman, Trevor Hastie and Rob Tibshirani (2011). glasso: Graphical lasso-estimation of Gaussian graphical models. R package version 1.7. http://CRAN.R-project.org/package=glasso

Index