Package 'NetworkToolbox'

March 8, 2020

Title Methods and Measures for Brain, Cognitive, and Psychometric Network Analysis

Version 1.4.0

Date 2020-03-08

Maintainer Alexander Christensen <alexpaulchristensen@gmail.com>

Description Implements network analysis and graph theory measures used in neuroscience, cognitive science, and psychology. Methods include various filtering methods and approaches such as threshold, dependency (Kenett, Tumminello, Madi, Gur-Gershgoren, Mantegna, & Ben-Jacob, 2010 <doi:10.1371/journal.pone.0015032>), Information Filtering Networks (Barfuss, Massara, Di Matteo, & Aste, 2016 <doi:10.1103/PhysRevE.94.062306>), and Efficiency-Cost Optimization (Fallani, Latora, & Chavez, 2017 <doi:10.1371/journal.pcbi.1005305>). Brain methods include the recently developed Connectome Predictive Modeling (see references in package). Also implements several network measures including local network characteristics (e.g., centrality), community-level network characteristics (e.g., community centrality), global network characteristics (e.g., clustering coefficient), and various other measures associated with the reliability and reproducibility of network analysis.

Depends R (>= 3.3.0)

License GPL (>= 3.0)

Encoding UTF-8

LazyData true

Imports Matrix, psych, corrplot, fdrtool, R.matlab, igraph, qgraph, ppcor, parallel, foreach, doParallel, MASS, pwr, IsingFit, pbapply

RoxygenNote 7.0.2

NeedsCompilation no

Author Alexander Christensen [aut, cre] (<https://orcid.org/0000-0002-9798-7037>), Guido Previde Massara [ctb] (<https://orcid.org/0000-0003-0502-2789>)

Repository CRAN

Date/Publication 2020-03-08 17:00:06 UTC

R topics documented:

`NetworkToolbox-package`

NetworkToolbox–package

Description

Implements network analysis and graph theory measures used in neuroscience, cognitive science, and psychology. Methods include various filtering methods and approaches such as threshold, dependency, Information Filtering Networks, and Efficiency-Cost Optimization. Brain methods include the recently developed Connectome Predictive Modeling. Also implements several network measures including local network characteristics (e.g., centrality), global network characteristics (e.g., clustering coefficient), and various other measures associated with the reliability and reproducibility of network analysis.

Author(s)

Alexander Christensen <alexpaulchristensen@gmail.com>

References

Christensen, A. P. (in press). NetworkToolbox: Methods and measures for brain, cognitive, and psychometric network analysis in R. *The R Journal*, *10*, 422-439.

`adapt.a` *Adaptive Alpha*

Description

Compute an alpha value adjusted for sample size. The adjusted value is based on Perez and Pericchi's (2014) formula (equation 11, see below) using a reference sample, which can be defined a priori or estimated using the sample size calculation from power.

$$\frac{\alpha * \sqrt{n_0 \times (log(n_0) + \chi^2_\alpha(1))}}{\sqrt{n^* \times (log(n^*) + \chi^2_\alpha(1))}}$$

Usage

```
adapt.a(
  test = c("anova", "chisq", "cor", "one.sample", "two.sample", "paired"),
  ref.n = NULL,
  n = NULL,
  alpha = 0.05,
  power = 0.8,
  efxize = c("small", "medium", "large"),
  groups = NULL,
  df = NULL
)
```

Arguments

`test`	Type of statistical test being used. Can be any of the tests listed
`ref.n`	*n0* in the above equation. Reference sample size. If sample size was determined a priori, then the reference number of participants can be set. This removes the calculation of sample size based on power
`n`	*n** in the above equation. Number of participants in the experiment sample (or per group)
`alpha`	α in the above equation. Alpha value to adjust. Defaults to `.05`
`power`	Power $(1 - \beta)$ value. Used to estimate the reference sample size (n0). Defaults to `.80`
`efxize`	Effect size to be used to estimate the reference sample size. Effect sizes are based on Cohen (1992). Numeric values can be used. Defaults to `"medium"`
`groups`	Number of groups (only for `test = "anova"`)
`df`	Number of degrees of freedom (only for `test = "chisq"`)

Value

A list containing the following objects:

`adapt.a`	The adapted alpha value
`crit.value`	The critical value associated with the adapted alpha value
`orig.a`	The original alpha value
`ref.n`	The reference sample size based on alpha, power, effect size, and test
`exp.n`	The sample size of the experimental sample
`power`	The power used to determine the reference sample size
`test`	The type of statistical test used

Author(s)

Alexander Christensen <alexpaulchristensen@gmail.com>

References

Cohen, J. (1992). A power primer. *Psychological Bulletin*, *112*, 155-159.

Perez, M. E., & Pericchi, L. R. (2014). Changing statistical significance with the amount of information: The adaptive *a* significance level. *Statistics & Probability Letters*, *85*, 20-24.

Examples

```
#ANOVA
adapt.anova <- adapt.a(test = "anova", n = 200, alpha = .05, power = .80, groups = 3)

#Chi-square
adapt.chisq <- adapt.a(test = "chisq", n = 200, alpha = .05, power = .80, df = 3)

#Correlation
adapt.cor <- adapt.a(test = "cor", n = 200, alpha = .05, power = .80)

#One-sample t-test
adapt.one <- adapt.a(test = "one.sample", n = 200, alpha = .05, power = .80)

#Two-sample t-test
adapt.two <- adapt.a(test = "two.sample", n = 200, alpha = .05, power = .80)

#Paired sample t-test
adapt.paired <- adapt.a(test = "paired", n = 200, alpha = .05, power = .80, efxize = "medium")
```

`behavOpen` *NEO-PI-3 for Resting-state Data*

Description

NEO-PI-3 Openness to Experience associated with resting-state data (n = 144).

Usage

```
data(behavOpen)
```

Format

behavOpen (vector, length = 144)

Details

Behavioral data of NEO-PI-3 associated with each connectivity matrix (open).

To access the resting-state brain data, please go to `https://drive.google.com/file/d/1ugwi7nRrlHQYuGPzEB4wYzsizFr`
`view`

References

Beaty, R. E., Chen, Q., Christensen, A. P., Qiu, J., Silvia, P. J., & Schacter, D. L. (2018). Brain networks of the imaginative mind: Dynamic functional connectivity of default and cognitive control networks relates to Openness to Experience. *Human Brain Mapping*, *39*, 811-821.

Beaty, R. E., Kenett, Y. N., Christensen, A. P., Rosenberg, M. D., Benedek, M., Chen, Q., ... & Silvia, P. J. (2018). Robust prediction of individual creative ability from brain functional connectivity. *Proceedings of the National Academy of Sciences*, 201713532.

Examples

```
data("behavOpen")
```

`betweenness` *Betweenness Centrality*

Description

Computes betweenness centrality of each node in a network

Usage

```
betweenness(A, weighted = TRUE)
```

Arguments

`A`	An adjacency matrix of network data
`weighted`	Is the network weighted? Defaults to `TRUE`. Set to `FALSE` for unweighted measure of betweenness centrality

Value

A vector of betweenness centrality values for each node in the network

Author(s)

Alexander Christensen <alexpaulchristensen@gmail.com>

References

Rubinov, M., & Sporns, O. (2010). Complex network measures of brain connectivity: Uses and interpretations. *NeuroImage*, *52*, 1059-1069.

Examples

```
# Pearson's correlation only for CRAN checks
A <- TMFG(neoOpen, normal = FALSE)$A

#Weighted BC
BCw <- betweenness(A)

#Unweighted BC
BC <- betweenness(A, weighted = FALSE)
```

`binarize` *Binarize Network*

Description

Converts weighted adjacency matrix to a binarized adjacency matrix

Usage

```
binarize(A)
```

Arguments

`A`	An adjacency matrix of network data (or an array of matrices)

Value

Returns an adjacency matrix of 1's and 0's

Author(s)

Alexander Christensen <alexpaulchristensen@gmail.com>

Examples

```
# Pearson's correlation only for CRAN checks
A <- TMFG(neoOpen, normal = FALSE)$A

neoB <- binarize(A)
```

`closeness` *Closeness Centrality*

Description

Computes closeness centrality of each node in a network

Usage

```
closeness(A, weighted = TRUE)
```

Arguments

`A`	An adjacency matrix of network data
`weighted`	Is the network weighted? Defaults to `TRUE`. Set to `FALSE` for unweighted measure of closeness centrality

Value

A vector of closeness centrality values for each node in the network

Author(s)

Alexander Christensen <alexpaulchristensen@gmail.com>

References

Rubinov, M., & Sporns, O. (2010). Complex network measures of brain connectivity: Uses and interpretations. *NeuroImage*, *52*, 1059-1069.

Examples

```
# Pearson's correlation only for CRAN checks
A <- TMFG(neoOpen, normal = FALSE)$A

#Weighted LC
LC <- closeness(A)

#Unweighted LC
LC <- closeness(A, weighted = FALSE)
```

`clustcoeff` *Clustering Coefficient*

Description

Computes global clustering coefficient (CC) and local clustering coefficient (CCi)

Usage

```
clustcoeff(A, weighted = FALSE)
```

Arguments

`A`	An adjacency matrix of network data
`weighted`	Is the network weighted? Defaults to `FALSE`. Set to `TRUE` for weighted measures of CC and CCi

Value

Returns a list containing:

`CC`	Global clustering coefficient. The average clustering coefficient for each node in the network
`CCi`	Local clustering coefficient. The clustering coefficient for each node in the network

Author(s)

Alexander Christensen <alexpaulchristensen@gmail.com>

References

Rubinov, M., & Sporns, O. (2010). Complex network measures of brain connectivity: Uses and interpretations. *NeuroImage*, *52*, 1059-1069. doi: 10.1016/j.neuroimage.2009.10.003

Examples

```
# Pearson's correlation only for CRAN checks
A <- TMFG(neoOpen, normal = FALSE)$A

#Unweighted CC
CCu <- clustcoeff(A)

#Weighted CC
CCw <- clustcoeff(A, weighted=TRUE)
```

`comcat` *Communicating Nodes*

Description

Computes the between-community strength for each node in the network

Usage

```
comcat(
  A,
  comm = c("walktrap", "louvain"),
  cent = c("strength", "degree"),
  absolute = TRUE,
  metric = c("across", "each"),
  diagonal = 0,
  ...
)
```

Arguments

`A`	An adjacency matrix of network data
`comm`	Can be a vector of community assignments or community detection algorithms (`"walktrap"` or `"louvain"`) can be used to determine the number of factors. Defaults to `"walktrap"`. Set to `"louvain"` for `louvain` community detection
`cent`	Centrality measure to be used. Defaults to `"strength"`.
`absolute`	Should network use absolute weights? Defaults to `TRUE`. Set to `FALSE` for signed weights
`metric`	Whether the metric should be compute for across all of the communities (a single value) or for each community (a value for each community). Defaults to `"across"`. Set to `"each"` for values for each community
`diagonal`	Sets the diagonal values of the `A` input. Defaults to `0`
`...`	Additional arguments for `cluster_walktrap` and `louvain` community detection algorithms

Value

A vector containing the between-community strength value for each node

Author(s)

Alexander Christensen <alexpaulchristensen@gmail.com>

References

Blanken, T. F., Deserno, M. K., Dalege, J., Borsboom, D., Blanken, P., Kerkhof, G. A., & Cramer, A. O. (2018). The role of stabilizing and communicating symptoms given overlapping communities in psychopathology networks. *Scientific Reports*, *8*, 5854.

Examples

```
# Pearson's correlation only for CRAN checks
A <- TMFG(neoOpen, normal = FALSE)$A

communicating <- comcat(A, comm = "walktrap", cent = "strength", metric = "across")
```

`comm.close` *Community Closeness Centrality*

Description

Computes the community closeness centrality measure of each community in a network

Usage

```
comm.close(A, comm, weighted = FALSE)
```

Arguments

`A`	An adjacency matrix of network data
`comm`	A vector or matrix corresponding to the community each node belongs to
`weighted`	Is the network weighted? Defaults to `FALSE`. Set to `TRUE` for weighted measures

Value

A vector of community closeness centrality values for each specified community in the network (larger values suggest more central positioning)

Author(s)

Alexander Christensen <alexpaulchristensen@gmail.com>

References

Christensen, A. P. (in press). NetworkToolbox: Methods and measures for brain, cognitive, and psychometric network analysis in R. *The R Journal*, *10*, 422-439.

Examples

```
# Pearson's correlation only for CRAN checks
A <- TMFG(neoOpen, normal = FALSE)$A

comm <- igraph::walktrap.community(convert2igraph(abs(A)))$membership

#Weighted
result <- comm.close(A, comm)

#Unweighted
result <- comm.close(A, comm, weighted = FALSE)
```

`comm.eigen` *Community Eigenvector Centrality*

Description

Computes the flow.frac for each community in the network. The values are equivalent to the community's eigenvector centrality

Usage

```
comm.eigen(A, comm, weighted = TRUE)
```

Arguments

`A`	An adjacency matrix
`comm`	A vector or matrix corresponding to the community each node belongs to
`weighted`	Is the network weighted? Defaults to `TRUE`. Set to `FALSE` for weighted measures

Value

A vector of community eigenvector centrality values for each specified community in the network (larger values suggest more central positioning)

Author(s)

Alexander Christensen <alexpaulchristensen@gmail.com>

References

Giscard, P. L., & Wilson, R. C. (2018). A centrality measure for cycles and subgraphs II. *Applied Network Science*, *3*, 9.

Examples

```
# Pearson's correlation only for CRAN checks
A <- TMFG(neoOpen, normal = FALSE)$A

comm <- igraph::walktrap.community(convert2igraph(abs(A)))$membership

result <- comm.eigen(A, comm)
```

`comm.str` *Community Strength/Degree Centrality*

Description

Computes the community `strength`/`degree` centrality measure of each community in a network or computes the `strength`/`degree` centrality measure of each community's connections to the other communities

Usage

```
comm.str(A, comm, weighted = TRUE, measure = c("within", "between"))
```

Arguments

`A`	An adjacency matrix of network data
`comm`	A vector corresponding to the community each node belongs to
`weighted`	Is the network weighted? Defaults to `TRUE`. Set to `FALSE` for weighted measures
`measure`	Type of measure to compute: • `"within"` Computes the community strength or degree of nodes within its own community • `"between"` Computes the community strength or degree of nodes outside of its own community

Value

A vector of community strength/degree centrality values for each specified community in the network (larger values suggest more central positioning)

Author(s)

Alexander Christensen <alexpaulchristensen@gmail.com>

Examples

```
# Pearson's correlation only for CRAN checks
A <- TMFG(neoOpen, normal = FALSE)$A

comm <- igraph::walktrap.community(convert2igraph(abs(A)))$membership

#Strength
within.ns <- comm.str(A, comm, measure = "within")
between.ns <- comm.str(A, comm, measure = "between")

#Degree
within.deg <- comm.str(A, comm, weighted = FALSE, measure = "within")
between.deg <- comm.str(A, comm, weighted = FALSE, measure = "between")
```

`conn` *Network Connectivity*

Description

Computes the average and standard deviation of the weights in the network

Usage

```
conn(A)
```

Arguments

`A`	An adjacency matrix of a network

Value

Returns a list containing:

`weights`	Each edge weight in the network
`mean`	The mean of the edge weights in the network
`sd`	The standard deviation of the edge weights in the network
`total`	The sum total of the edge weights in the network

Author(s)

Alexander Christensen <alexpaulchristensen@gmail.com>

Examples

```
# Pearson's correlation only for CRAN checks
A <- TMFG(neoOpen, normal = FALSE)$A

connectivity <- conn(A)
```

`convert2igraph` *Convert Network(s) to igraph's Format*

Description

Converts single or multiple networks into `igraph`'s format for network analysis

Usage

```
convert2igraph(A, neural = FALSE)
```

Arguments

`A`	Adjacency matrix (network matrix) or brain connectivity array (from `convertConnBrainMat`)
`neural`	Is input a brain connectivity array (i.e., m x m x n)? Defaults to `FALSE`. Set to `TRUE` to convert each brain connectivity matrix

Value

Returns a network matrix in `igraph`'s format or returns a list of brain connectivity matrices each of which have been convert to `igraph`'s format

Author(s)

Alexander Christensen <alexpaulchristensen@gmail.com>

Examples

```
# Pearson's correlation only for CRAN checks
A <- TMFG(neoOpen, normal = FALSE)$A

igraphNetwork <- convert2igraph(A)

## Not run:
neuralarray <- convertConnBrainMat()

igraphNeuralList <- convert2igraph(neuralarray, neural = TRUE)

## End(Not run)
```

`convertConnBrainMat` *Import CONN Toolbox Brain Matrices to R format*

Description

Converts a Matlab brain z-score connectivity array (n x n x m) where **n** is the n x n connectivity matrices and **m** is the participant. If you would like to simply import a connectivity array from Matlab, then see the examples

Usage

```
convertConnBrainMat(MatlabData, progBar = TRUE)
```

Arguments

`MatlabData`	Input for Matlab data file. Defaults to interactive file choice
`progBar`	Should progress bar be displayed? Defaults to `TRUE`. Set `FALSE` for no progress bar

Value

Returns a list containing:

`rmat`	Correlation matrices for each participant (m) in an array (n x n x m)
`zmat`	Z-score matrices for each participant (m) in an array (n x n x m)

Author(s)

Alexander Christensen <alexpaulchristensen@gmail.com>

Examples

```
## Not run:
neuralarray <- convertConnBrainMat()

#Import correlation connectivity array from Matlab
library(R.matlab)
neuralarray<-readMat(file.choose())

## End(Not run)
```

`cor2cov` *Convert Correlation Matrix to Covariance Matrix*

Description

Converts a correlation matrix to a covariance matrix

Usage

```
cor2cov(cormat, data)
```

Arguments

`cormat`	A correlation matrix
`data`	The dataset the correlation matrix is from

Value

Returns a covariance matrix

Author(s)

Alexander Christensen <alexpaulchristensen@gmail.com>

Examples

```
cormat <- cor(neoOpen)

covmat <- cor2cov(cormat,neoOpen)
```

`core.items` *Core Items*

Description

Automatically determines core, intermediary, and peripheral items in the network. The entire network or within-community gradations can be determined. Based on the `hybrid` centrality

Usage

```
core.items(A, comm, by = c("network", "communities"))
```

Arguments

`A`	An adjacency matrix of network data
`comm`	A vector or matrix corresponding to the community each node belongs to
`by`	Should the core items be defined by network or communities? Defaults to `"network"`. Set to `"communities"` to define core items within communities

Value

Returns a list containing:

`core`	Core items for each community
`inter`	Intermediate items for each community
`peri`	Peripheral items for each community

Author(s)

Alexander Christensen <alexpaulchristensen@gmail.com>

Examples

```
#network
# Pearson's correlation only for CRAN checks
A <- TMFG(neoOpen, normal = FALSE)$A

#core items by network
coreBYnetwork <- core.items(A, by = "network")

#theoretical factors
comm <- c(rep(1,8),rep(2,8),rep(3,8),rep(4,8),rep(5,8),rep(6,8))

#core items by communities
coreBYcomm <- core.items(A, comm, by = "communities")
```

`cpm` *Connectome Predictive Modeling*

Description

Suite of functions for Connectome Predictive Modeling (CPM). **See and cite Finn et al., 2015; Rosenberg et al., 2016; Shen et al., 2017**

- `cpmIV`

  Internal Validation method (Rosenberg et al., 2016; Shen et al., 2017). Using a leave-one-out approach, this method correlates a behavioral statistic `bstat` with each edge of a whole-brain network across participants. Using the significant edges in the network `thresh`, a connectome model is built (without the participant's network). A linear regression model is fit, with the behavioral statistic being regressed on the connectome model. The left out participants connectome model is then used with the linear regression weights to compute their predicted behavioral score. This is repeated for every participant. The predicted scores are correlated with their observed score. Significant values suggest that the connectome is related to the behavioral statistic

- `cpmEV`

  UNDER DEVELOPMENT. External Validation method (Beaty et al., 2018). Performs similar function as `cpmIV` but uses data to train `train_na` the connectome model using a behavioral statistic `train_b`. This training connectome model is then used to predict another dataset `valid_na`, using the same behavioral statistic `valid_b`. The full training dataset `FALSE` or the leave-one-out `overlap` = `TRUE` approach can be used

- `cpmFP`

  Fingerprinting method (Finn et al., 2015). Uses CPM approach to identify participants across two sessions

- `cpmFPperm`

  Fingerprinting method (Finn et al., 2015). Uses permutation method to estimate the significance of of the `cpmFP` results

- `cpmPlot`

  Plots the CPM results

Usage

```
cpmIV(neuralarray, bstat, covar, thresh = .01, groups = NULL,
      method = c("mean", "sum"), model = c("linear","quadratic","cubic"),
      corr = c("pearson","spearman"), nEdges,
      standardize = FALSE, cores, progBar = TRUE)

cpmEV(train_na, train_b, valid_na, valid_b, thresh = .01,
      overlap = FALSE, progBar = TRUE)

cpmFP(session1, session2, progBar = TRUE)

cpmFPperm(session1, session2, iter = 1000, progBar = TRUE)

cpmPlot(cpm.obj, visual.nets = FALSE)
```

Arguments

`neuralarray`	Array from `convertConnBrainMat` function
`bstat`	Behavioral statistic for each participant with neural data (a vector)
`covar`	Covariates to be included in predicting relevant edges (**time consuming**). **Must** be input as a `list()` (see examples)
`thresh`	Sets an α threshold for edge weights to be retained. Defaults to `.01`
`groups`	Allows grouping variables to be used for plotting points. **Must** be a vector. Defaults to `NULL`

`method`	Use `"mean"` or `"sum"` of edge strengths in the positive and negative connectomes. Defaults to `"mean"`
`model`	Regression model to use for fitting the data. Defaults to `"linear"`
`corr`	Correlation method for assessing the relationship between the behavioral measure and edges between ROIs. Defaults to `"pearson"`. Set to `"spearman"` for non-linear or monotonic associations
`nEdges`	Number of participants that are required to have an edge to appear in the plots. Defaults to 10 percent of edges in participants
`standardize`	Should the behavioral statistic (`bstat`) be standardized? Defaults to `FALSE`
`cores`	Number of computer processing cores to use when performing covariate analyses. Defaults to *n* - 1 total number of cores. Set to any number between 1 and maximum amount of cores on your computer
`progBar`	Should progress bar be displayed? Defaults to `TRUE`. Set to `FALSE` for no progress bar
`train_na`	Training dataset (an array from `convertConnBrainMat` function)
`train_b`	Behavioral statistic for each participant for the **training** neural data (a vector)
`valid_na`	Validation dataset (an array from `convertConnBrainMat` function)
`valid_b`	Behavioral statistic for each participant for the **validation** neural data (a vector)
`overlap`	Should leave-one-out cross-validation be used? Defaults to `FALSE` (use full dataset, no leave-one-out). Set to `TRUE` to select edges that appear in every leave-one-out cross-validation network (*time consuming*)
`session1`	Array from `convertConnBrainMat` function (first session)
`session2`	Array from `convertConnBrainMat` function (second session)
`iter`	Number of iterations to perform. Defaults to `1000`
`cpm.obj`	`cpm` object
`visual.nets`	Boolean. Uses `qgraph` to plot connectivity between the networks as a network. Defaults to `FALSE`. Set to `TRUE` to visualize the networks

Value

`cpmIV` and `cpmEV`:

Returns a list containing:

`results`	A matrix containing: r coefficient (`r`), p-value (`p-value`), mean absolute error (`mae`), root mean square error (`rmse`)
`posMask`	Positive connectivity for input in BioImage Suite Connectivity Viewer
`negMask`	Negative connectivity for input in BioImage Suite Connectivity Viewer

`cpmFP`:

Returns a matrix containing the percentage and number of correctly identified subjects for sessions 1 and 2

`cpmPlot`:

Returns plot of connectivity differences between the positive and negative masks

Author(s)

Alexander Christensen <alexpaulchristensen@gmail.com>

References

Beaty, R. E., Kenett, Y. N., Christensen, A. P., Rosenberg, M. D., Benedek, M., Chen, Q., Fink, A., Qiu, J., Kwapil, T. R., Kane, M. J., & Silvia, P. J. (2018). Robust prediction of individual creative ability from brain functional connectivity. *Proceedings of the National Academy of Sciences*, *115*, 1087-1092.

Finn, E. S., Shen, X., Scheinost, D., Rosenberg, M. D., Huang, J., Chun, M. M., Papademetris, X., Constable, R. T. (2015). Functional connectome fingerprinting: Identifying individuals using patterns of brain connectivity. *Nature Neuroscience*, *18*, 1664-1671.

Rosenberg, M. D., Finn, E. S., Scheinost, D., Papademetris, X., Shen, X., Constable, R. T., Chun, M. M. (2016). A neuromarker of sustained attention from whole-brain functional connectivity. *Nature Neuroscience*, *19*, 165-171.

Shen, X. Finn, E. S., Scheinost, D., Rosenberg, M. D., Chun, M. M., Papademetris, X., Constable, R. T. (2017). Using connectome-based predictive modeling to predict individual behavior from brain connectivity. *Nature Protocols*, *12*, 506-518.

Wei, T. & Simko, V.(2017). R package "corrplot": Visualization of a correlation matrix (Version 0.84).

Examples

```
# Load data
behav <- behavOpen

## Not run:

# Download associated brain data
https://drive.google.com/file/d/1ugwi7nRrlHQYuGPzEB4wYzsizFrIMvKR/view

# Load brain data
load(”restOpen.rda”)

# Run cpmIV
res <- cpmIV(neuralarray = restOpen, bstat = behav, cores = 4)

# Plot cpmIV results
cpmPlot(res)


## End(Not run)
```

`dCor` *Distance Correlation for ROI Time Series*

Description

Computes the distance correlation (Yoo et al., 2019) for ROI time series data. This function is mainly a subroutine for the `dCor.parallel` function

Usage

```
dCor(neurallist, centering = c("U", "double"))
```

Arguments

`neurallist`	List. A time series list from `convertConnBrainMat` function
`centering`	Character. Options for centering the Euclidean distances. • `"U"` Uses number of time points minus 2 in the computation of the mean • `"double"` Uses the mean

Value

Returns a m x m matrix corresponding to distance correlations between ROIs

Author(s)

Alexander Christensen <alexpaulchristensen@gmail.com>

References

Yoo, K., Rosenberg, M. D., Noble, S., Scheinost, D., Constable, R. T., & Chun, M. M. (2019). Multivariate approaches improve the reliability and validity of functional connectivity and prediction of individual behaviors. *NeuroImage*, *197*, 212-223.

Examples

```
## Not run:
# Import time series data
neurallist <- convertConnBrainMat()

# Run distance correlation
dCor(neurallist)


## End(Not run)
```

`dCor.parallel` *Parallelization of Distance Correlation for ROI Time Series*

Description

Parallelizes the `dCor` function for faster computation times

Usage

```
dCor.parallel(neurallist, cores)
```

Arguments

`neurallist`	List of lists. A list containing the time series list from all participants imported from the `convertConnBrainMat` function
`cores`	Number of computer processing cores to use when performing covariate analyses. Defaults to *n* - 1 total number of cores. Set to any number between 1 and maximum amount of cores on your computer

Value

Returns a m x m x n array corresponding to distance correlations between ROIs (m x m matrix) for n participants

Author(s)

Alexander Christensen <alexpaulchristensen@gmail.com>

References

Yoo, K., Rosenberg, M. D., Noble, S., Scheinost, D., Constable, R. T., & Chun, M. M. (2019). Multivariate approaches improve the reliability and validity of functional connectivity and prediction of individual behaviors. *NeuroImage*, *197*, 212-223.

Examples

```
## Not run:
# Import time series data
for(i in 1:5)

# Run distance correlation
dCor.parallel(mat.list, cores = 2)


## End(Not run)
```

`degree` *Degree*

Description

Computes degree of each node in a network

Usage

```
degree(A)
```

Arguments

`A`	An adjacency matrix of network data

Value

A vector of degree values for each node in the network.

If directed network, returns a list containing:

`inDegree`	Degree of incoming edges (pointing to the node)
`outDegree`	Degree of outgoing edges (pointing away from the node)
`relInf`	Relative degree of incoming and outgoing edges. Positive values indicate more outgoing degree relative to incoming degree. Negative values indicate more incoming degree relative to outgoing degree

Author(s)

Alexander Christensen <alexpaulchristensen@gmail.com>

References

Rubinov, M., & Sporns, O. (2010). Complex network measures of brain connectivity: Uses and interpretations. *NeuroImage*, *52*, 1059-1069.

Examples

```
#Undirected network
## Pearson's correlation only for CRAN checks
A <- TMFG(neoOpen, normal = FALSE)$A

deg <- degree(A)

#Directed network
## Not run:
dep <- depend(neoOpen)

Adep <- TMFG(dep, depend = TRUE)$A
```

```
deg <- degree(Adep)

## End(Not run)
```

`depend` *Dependency Network Approach*

Description

Generates a dependency matrix of the data (index argument is still in testing phase)

Usage

```
depend(
  data,
  normal = FALSE,
  na.data = c("pairwise", "listwise", "fiml", "none"),
  index = FALSE,
  fisher = FALSE,
  progBar = TRUE
)
```

Arguments

`data`	A set of data
`normal`	Should data be transformed to a normal distribution? Defaults to `FALSE`. Data is not transformed to be normal. Set to `TRUE` if data should be transformed to be normal (computes correlations using the cor_auto function)
`na.data`	How should missing data be handled? For `"listwise"` deletion the `na.omit` function is applied. Set to `"fiml"` for Full Information Maximum Likelihood (corFiml). Full Information Maximum Likelihood is **recommended** but time consuming
`index`	Should correlation with the latent variable (i.e., weighted average of all variables) be removed? Defaults to `FALSE`. Set to `TRUE` to remove common latent factor
`fisher`	Should Fisher's Z-test be used to keep significantly higher influences (index only)? Defaults to `FALSE`. Set to `TRUE` to remove non-significant influences
`progBar`	Should progress bar be displayed? Defaults to `TRUE`. Set to `FALSE` for no progress bar

Value

Returns an adjacency matrix of dependencies

Author(s)

Alexander Christensen <alexpaulchristensen@gmail.com>

References

Kenett, D. Y., Tumminello, M., Madi, A., Gur-Gershgoren, G., Mantegna, R. N., & Ben-Jacob, E. (2010). Dominating clasp of the financial sector revealed by partial correlation analysis of the stock market. *PLoS one*, *5*, e15032.

Kenett, D. Y., Huang, X., Vodenska, I., Havlin, S., & Stanley, H. E. (2015). Partial correlation analysis: Applications for financial markets. *Quantitative Finance*, *15*, 569-578.

Examples

```
## Not run:
D <- depend(neoOpen)

Dindex <- depend(neoOpen, index = TRUE)

## End(Not run)
```

`depna` *Dependency Neural Networks*

Description

Applies the dependency network approach to neural network array

Usage

```
depna(neuralarray, pB = TRUE, ...)
```

Arguments

`neuralarray`	Array from `convertConnBrainMat` function
`pB`	Should progress bar be displayed? Defaults to `TRUE`. Set `FALSE` for no progress bar
`...`	Additional arguments from `depend` function

Value

Returns an array of n x n x m dependency matrices

Author(s)

Alexander Christensen <alexpaulchristensen@gmail.com>

References

Jacob, Y., Winetraub, Y., Raz, G., Ben-Simon, E., Okon-Singer, H., Rosenberg-Katz, K., ... & Ben-Jacob, E. (2016). Dependency Network Analysis (DEPNA) reveals context related influence of brain network nodes. *Scientific Reports*, *6*, 27444.

Kenett, D. Y., Tumminello, M., Madi, A., Gur-Gershgoren, G., Mantegna, R. N., & Ben-Jacob, E. (2010). Dominating clasp of the financial sector revealed by partial correlation analysis of the stock market. *PLoS one*, *5*, e15032.

Examples

```
## Not run:
neuralarray <- convertConnBrainMat()

dependencyneuralarray <- depna(neuralarray)

## End(Not run)
```

`desc` *Variable Descriptive Statistics*

Description

Computes `mean`, standard deviation (`sd`), minimum value (`min`), maximum value (`max`), and univariate normal statistics (`normal?`) for a variable

Usage

```
desc(data, column, histplot = TRUE)
```

Arguments

`data`	A matrix or data frame
`column`	Column name or number in `data`
`histplot`	A histogram plot of the variable

Value

A data frame containing values for `n` (number of cases), `missing` (number of missing cases), `mean`, `sd`, `min`, and `max`. `normal?` will contain yes/no for whether the variable is normally distributed based on the `shapiro.test` for a variable

Author(s)

Alexander Christensen <alexpaulchristensen@gmail.com>

Examples

```
desc(neoOpen,1)
```

`desc.all` *Dataset Descriptive Statistics*

Description

Computes `mean`, standard deviation (`sd`), minimum value (`min`), maximum value (`max`), and univariate normal statistics (`normal?`) for the entire dataset

Usage

```
desc.all(data)
```

Arguments

`data`	A matrix or data frame

Value

A data frame containing values for `n` (number of cases), `missing` (number of missing cases), `mean`, `sd`, `min`, and `max`. `normal?` will contain yes/no for whether the variable is normally distributed based on the `shapiro.test` for the entire dataset

Author(s)

Alexander Christensen <alexpaulchristensen@gmail.com>

Examples

```
desc.all(neoOpen)
```

`distance` *Distance*

Description

Computes distance matrix of the network

Usage

```
distance(A, weighted = FALSE)
```

Arguments

`A`	An adjacency matrix of network data
`weighted`	Is the network weighted? Defaults to `FALSE`. Set to `TRUE` for weighted measure of distance

Value

A distance matrix of the network

Author(s)

Alexander Christensen <alexpaulchristensen@gmail.com>

References

Rubinov, M., & Sporns, O. (2010). Complex network measures of brain connectivity: Uses and interpretations. *NeuroImage*, *52*, 1059-1069.

Examples

```
# Pearson's correlation only for CRAN checks
A <- TMFG(neoOpen, normal = FALSE)$A

#Unweighted
Du <- distance(A)

#Weighted
Dw <- distance(A, weighted = TRUE)
```

`diversity` *Diversity Coefficient*

Description

Computes the diversity coefficient for each node. The diversity coefficient measures a node's connections to communitites outside of its own community. Nodes that have many connections to other communities will have higher diversity coefficient values. Positive and negative signed weights for diversity coefficients are computed separately.

Usage

```
diversity(A, comm = c("walktrap", "louvain"))
```

Arguments

`A`	Network adjacency matrix
`comm`	A vector of corresponding to each item's community. Defaults to `"walktrap"` for the `cluster_walktrap` community detection algorithm. Set to `"louvain"` for the `louvain` community detection algorithm. Can also be set to user-specified communities (see examples)

Details

Values closer to 1 suggest greater between-community connectivity and values closer to 0 suggest greater within-community connectivity

Value

Returns a list containing:

`overall`	Diversity coefficient without signs considered
`positive`	Diversity coefficient with only positive sign
`negative`	Diversity coefficient with only negative sign

Author(s)

Alexander Christensen <alexpaulchristensen@gmail.com>

References

Rubinov, M., & Sporns, O. (2010). Complex network measures of brain connectivity: Uses and interpretations. *NeuroImage*, *52*, 1059-1069.

Examples

```
# Pearson's correlation only for CRAN checks
A <- TMFG(neoOpen, normal = FALSE)$A

#theoretical communities
comm <- rep(1:8, each = 6)

gdiv <- diversity(A, comm = comm)

#walktrap communities
wdiv <- diversity(A, comm = "walktrap")
```

`ECO` *ECO Neural Network Filter*

Description

Applies the ECO neural network filtering method

Usage

```
ECO(data, directed = FALSE)
```

Arguments

`data`	Can be a dataset or a correlation matrix
`directed`	Is the network directed? Defaults to `FALSE`. Set `TRUE` if the network is directed

Value

A sparse association matrix

Author(s)

Alexander Christensen <alexpaulchristensen@gmail.com>

References

Fallani, F. D. V., Latora, V., & Chavez, M. (2017). A topological criterion for filtering information in complex brain networks. *PLoS Computational Biology*, *13*, e1005305.

Examples

```
eco.net <- ECO(neoOpen)
```

`ECOplusMaST` *ECO+MaST Network Filter*

Description

Applies the `ECO` neural network filtering method combined with the `MaST` filtering method

Usage

```
ECOplusMaST(data)
```

Arguments

`data`	Can be a dataset or a correlation matrix

Value

A sparse association matrix

Author(s)

Alexander Christensen <alexpaulchristensen@gmail.com>

References

Fallani, F. D. V., Latora, V., & Chavez, M. (2017). A topological criterion for filtering information in complex brain networks. *PLoS Computational Biology*, *13*, e1005305.

Examples

```
# half the variables for CRAN checks
ECOplusMaST.net <- ECOplusMaST(neoOpen[,c(1:24)])
```

`edgerep` *Edge Replication*

Description

Computes the number of edges that replicate between two cross-sectional networks

Usage

```
edgerep(A, B, corr = c("pearson", "spearman", "kendall"))
```

Arguments

`A`	An adjacency matrix of network A
`B`	An adjacency matrix of network B
`corr`	Correlation method for assessing the relationship between the replicated edge weights. Defaults to `"pearson"`. Set to `"spearman"` for non-linear or monotonic associations. Set to `"kendall"` for rank-order correlations

Value

Returns a list containing:

`replicatedEdges`	The edges that replicated and their weights
`replicated`	Number of edges that replicated
`meanDiff`	The average edge weight difference between the edges that replicated
`sdDiff`	The standard deviation edge weight difference between the edges that replicated
`cor`	The correlation between the edges that replicated

Lists for each network contain:

`totalEdges`	Total possible number of edges to be replicated
`percentage`	Percentage of edges that replicated relative to total possible
`density`	The density of the network

Author(s)

Alexander Christensen <alexpaulchristensen@gmail.com>

Examples

```
# normal set to FALSE for CRAN tests
tmfg <- TMFG(neoOpen, normal = FALSE)$A

# normal set to FALSE for CRAN tests
mast <- MaST(neoOpen, normal = FALSE)

edges <- edgerep(tmfg, mast)
```

`eigenvector` *Eigenvector Centrality*

Description

Computes eigenvector centrality of each node in a network

Usage

```
eigenvector(A, weighted = TRUE)
```

Arguments

`A`	An adjacency matrix of network data
`weighted`	Is the network weighted? Defaults to `TRUE`. Set to `FALSE` for unweighted measure of eigenvector centrality

Value

A vector of eigenvector centrality values for each node in the network

Author(s)

Alexander Christensen <alexpaulchristensen@gmail.com>

References

Rubinov, M., & Sporns, O. (2010). Complex network measures of brain connectivity: Uses and interpretations. *NeuroImage*, *52*, 1059-1069.

Examples

```
# Pearson's correlation only for CRAN checks
A <- TMFG(neoOpen, normal = FALSE)$A

#Weighted
EC <- eigenvector(A)

#Unweighted
EC <- eigenvector(A, weighted = FALSE)
```

`flow.frac` *Flow Fraction*

Description

Computes `eigenvector` centrality over nodes in a subset of nodes in the network. This measure generalizes across any subset of nodes and is not specific to communities

Usage

```
flow.frac(A, nodes)
```

Arguments

`A`	An adjacency matrix
`nodes`	A subset of nodes in the network

Value

Returns a flow fraction value

Author(s)

Alexander Christensen <alexpaulchristensen@gmail.com>

References

Giscard, P. L., & Wilson, R. C. (2018). A centrality measure for cycles and subgraphs II. *Applied Network Science*, *3*, 9.

Examples

```
# Pearson's correlation only for CRAN checks
A <- TMFG(neoOpen, normal = FALSE)$A

nodes <- seq(1,48,2)

result <- flow.frac(A, nodes)
```

`gain.functions` *MFCF Gain Functions*

Description

These functions maximize a gain criterion for adding a node to a clique (and the larger network). The flexibility of `MFCF` allows for any multivariate function to be used as a scoring function.

- `"logLik"`

  The log determinant of the matrix restricted to the separator minus the log determinant of the matrix restricted to the clique.
- `"logLik.val"`

  `"logLik"` with a further validation based on the likelihood ratio. If the increase in gain is not significant the routine stops adding nodes to the separator.
- `"rSquared.val"`

  The R squared from the regression of the node against the clique. Only the clique nodes with a regression coefficient significantly different from zero are added to the separator / new clique. The gain is different from zero only if the F-values is significant, It assumed that the `data` matrix is a dataset of realizations (i.e., `p` variables and `n` observations).

Usage

```
"logLik"
gfcnv_logdet(data, clique_id, cl, excl_nodes, ctreeControl)

"logLik.val"
gfcnv_logdet_val(data, clique_id, cl, excl_nodes, ctreeControl)

"rSquared.val"
gdcnv_lmfit(data, clique_id, cl, excl_nodes, ctreeControl)
```

Arguments

`data`	Matrix or data frame. Can be a dataset or a correlation matrix
`clique_id`	Numeric. Number corresponding to clique to add another node to
`cl`	List. List of cliques already assembled in the network
`excl_nodes`	Numeric vector. A vector of numbers corresponding to nodes not already included in the network
`ctreeControl`	List (length = 5). A list containing several parameters for controlling the clique tree sizes: • `min_size` Numeric. Minimum number of nodes allowed per clique. Defaults to 1 • `max_size` Numeric. Maximum number of nodes allowed per clique. Defaults to 8

- `pval` Numeric. *p*-value used to determine cut-offs for nodes to include in a clique. Defaults to `.05`
- `pen` Numeric. Multiplies the number of edges added to penalize complex models. Similar to the penalty term in AIC
- `drop_sep` Boolean. This parameter influences the MFCF only. If TRUE any separator can be used only once, as in the TMFG.
- `use_returns` Boolean. Only used in rSquared.val. If set to TRUE the regression is performed on log-returns. Defaults to `FALSE`

Value

Returns the value with the maximum gain

Author(s)

Guido Previde Massara <gprevide@gmail.com> and Alexander Christensen <alexpaulchristensen@gmail.com>

References

Massara, G. P. & Aste, T. (2019). Learning clique forests. *ArXiv*.

`gateway` *Gateway Coefficient*

Description

Computes the gateway coefficient for each node. The gateway coefficient measures a node's connections between its community and other communities. Nodes that are solely responsible for inter-community connectivity will have higher gateway coefficient values. Positive and negative signed weights for gateway coefficients are computed separately.

Usage

```
gateway(
  A,
  comm = c("walktrap", "louvain"),
  cent = c("strength", "betweenness")
)
```

Arguments

`A`	Network adjacency matrix
`comm`	A vector of corresponding to each item's community. Defaults to `"walktrap"` for the `cluster_walktrap` community detection algorithm. Set to `"louvain"` for the `louvain` community detection algorithm. Can also be set to user-specified communities (see examples)
`cent`	Centrality to community gateway coefficient. Defaults to `"strength"`. Set to `"betweenness"` to use the `betweenness` centrality

Value

Returns a list containing:

`overall`	Gateway coefficient without signs considered
`positive`	Gateway coefficient with only positive sign
`negative`	Gateway coefficient with only negative sign

Author(s)

Alexander Christensen <alexpaulchristensen@gmail.com>

References

Rubinov, M., & Sporns, O. (2010). Complex network measures of brain connectivity: Uses and interpretations. *NeuroImage*, *52*, 1059-1069.

Vargas, E. R., & Wahl, L. M. (2014). The gateway coefficient: A novel metric for identifying critical connections in modular networks. *The European Physical Journal B*, *87*, 1-10.

Examples

```
#theoretical communities
comm <- rep(1:8, each = 6)

# Pearson's correlation only for CRAN checks
A <- TMFG(neoOpen, normal = FALSE)$A

gw <- gateway(A, comm = comm)

#walktrap communities
wgw <- gateway(A, comm = "walktrap")
```

`hybrid` *Hybrid Centrality*

Description

Computes hybrid centrality of each node in a network

Usage

```
hybrid(A, BC = c("standard", "random"), beta)
```

Arguments

`A`	An adjacency matrix of network data
`BC`	How should the betweenness centrality be computed? Defaults to `"random"`. Set to `"standard"` for standard `betweenness`.
`beta`	Beta parameter to be passed to the `rspbc` function Defaults to .01

Value

A vector of hybrid centrality values for each node in the network (higher values are more central, lower values are more peripheral)

Author(s)

Alexander Christensen <alexpaulchristensen@gmail.com>

References

Christensen, A. P., Kenett, Y. N., Aste, T., Silvia, P. J., & Kwapil, T. R. (2018). Network structure of the Wisconsin Schizotypy Scales-Short Forms: Examining psychometric network filtering approaches. *Behavior Research Methods*, *50*, 2531-2550.

Pozzi, F., Di Matteo, T., & Aste, T. (2013). Spread of risk across financial markets: Better to invest in the peripheries. *Scientific Reports*, *3*, 1655.

Examples

```
# Pearson's correlation only for CRAN checks
A <- TMFG(neoOpen, normal = FALSE)$A

HC <- hybrid(A)
```

`impact` *Node Impact*

Description

Computes impact measure or how much the average distance in the network changes with that node removed of each node in a network (**Please see and cite Kenett et al., 2011**)

Usage

```
impact(A)
```

Arguments

`A`	An adjacency matrix of network data

Value

A vector of node impact values for each node in the network (impact > 0, greater ASPL when node is removed; impact < 0, lower ASPL when node is removed)

Author(s)

Alexander Christensen <alexpaulchristensen@gmail.com>

References

Cotter, K. N., Christensen, A. P., & Silvia, P. J. (in press). Understanding inner music: A dimensional approach to musical imagery. *Psychology of Aesthetics, Creativity, and the Arts*.

Kenett, Y. N., Kenett, D. Y., Ben-Jacob, E., & Faust, M. (2011). Global and local features of semantic networks: Evidence from the Hebrew mental lexicon. *PLoS one*, *6*, e23912.

Examples

```
# normal set to FALSE for CRAN tests
A <- TMFG(neoOpen, normal = FALSE)$A

nodeimpact <- impact(A)
```

`is.graphical` *Determines if Network is Graphical*

Description

Tests for whether the network is graphical. Input must be a partial correlation network. Function assumes that partial correlations were computed from a multivariate normal distribution

Usage

```
is.graphical(A)
```

Arguments

`A`	A partial correlation network (adjacency matrix)

Value

Returns a TRUE/FALSE for whether network is graphical

Author(s)

Alexander Christensen <alexpaulchristensen@gmail.com>

Examples

```
## Not run:
A <- LoGo(neoOpen, normal = TRUE, partial = TRUE)

is.graphical(A)

## End(Not run)
```

`kld` *Kullback-Leibler Divergence*

Description

Estimates the Kullback-Leibler Divergence which measures how one probability distribution diverges from the original distribution (equivalent means are assumed) Matrices **must** be positive definite inverse covariance matrix for accurate measurement. This is a **relative** metric

Usage

```
kld(base, test)
```

Arguments

`base`	Full or base model
`test`	Reduced or testing model

Value

A value greater than 0. Smaller values suggest the probability distribution of the reduced model is near the full model

Author(s)

Alexander Christensen <alexpaulchristensen@gmail.com>

References

Kullback, S., & Leibler, R. A. (1951). On information and sufficiency. *The Annals of Mathematical Statistics*, *22*, 79-86. doi: 10.1214/aoms/1177729694

Examples

```
A1 <- solve(cov(neoOpen))

## Not run:
A2 <- LoGo(neoOpen)

kld_value <- kld(A1, A2)

## End(Not run)
```

`lattnet` *Generates a Lattice Network*

Description

Generates a lattice network

Usage

```
lattnet(nodes, edges)
```

Arguments

`nodes`	Number of nodes in lattice network
`edges`	Number of edges in lattice network

Value

Returns an adjacency matrix of a lattice network

Author(s)

Alexander Christensen <alexpaulchristensen@gmail.com>

References

Rubinov, M., & Sporns, O. (2010). Complex network measures of brain connectivity: Uses and interpretations. *NeuroImage*, *52*, 1059-1069.

Examples

```
latt <- lattnet(10, 27)
```

`leverage` *Leverage Centrality*

Description

Computes leverage centrality of each node in a network (the degree of connected neighbors; **Please see and cite Joyce et al., 2010**)

Usage

```
leverage(A, weighted = TRUE)
```

Arguments

`A`	An adjacency matrix of network data
`weighted`	Is the network weighted? Defaults to `TRUE`. Set to `FALSE` for unweighted measure of leverage centrality

Value

A vector of leverage centrality values for each node in the network

Author(s)

Alexander Christensen <alexpaulchristensen@gmail.com>

References

Joyce, K. E., Laurienti, P. J., Burdette, J. H., & Hayasaka, S. (2010). A new measure of centrality for brain networks. *PLoS One*, *5* e12200.

Examples

```
# Pearson's correlation only for CRAN checks
A <- TMFG(neoOpen, normal = FALSE)$A

#Weighted
levW <- leverage(A)

#Unweighted
levU <- leverage(A, weighted = FALSE)
```

`LoGo` *Local/Global Inversion Method*

Description

Applies the Local/Global method to estimate a Gaussian Graphical Model (GGM) using a `TMFG`-filtered network (**see and cite Barfuss et al., 2016**). Also used to convert clique and separator structure from `MFCF` into partial correlation and precision matrices

Usage

```
LoGo(
  data,
  cliques,
  separators,
  normal = TRUE,
  na.data = c("pairwise", "listwise", "fiml", "none"),
  partial = TRUE,
  ...
)
```

Arguments

`data`	Must be a dataset
`cliques`	Cliques defined in the network. Input can be a list or matrix
`separators`	Separators defined in the network. Input can be a list or matrix
`normal`	Should data be transformed to a normal distribution? Defaults to `TRUE` (computes correlations using the `cor_auto` function). Set to `FALSE` for Pearson's correlations
`na.data`	How should missing data be handled? For `"listwise"` deletion the `na.omit` function is applied. Set to `"fiml"` for Full Information Maximum Likelihood (`corFiml`). Full Information Maximum Likelihood is **recommended** but time consuming
`partial`	Should the output network's connections be the partial correlation between two nodes given all other nodes? Defaults to `TRUE`, which returns a partial correlation matrix. Set to `FALSE` for a sparse inverse covariance matrix
`...`	Additional arguments (deprecated arguments)

Value

Returns the sparse LoGo-filtered inverse covariance matrix (`partial = FALSE`) or LoGo-filtered partial correlation matrix (`partial = TRUE`)

Author(s)

Alexander Christensen <alexpaulchristensen@gmail.com>

References

Barfuss, W., Massara, G. P., Di Matteo, T., & Aste, T. (2016). Parsimonious modeling with information filtering networks. *Physical Review E*, *94*, 062306.

Examples

```
# normal set to FALSE for CRAN tests
LoGonet <- LoGo(neoOpen, normal = FALSE, partial = TRUE)
```

`louvain` *Louvain Community Detection Algorithm*

Description

Computes a vector of communities (community) and a global modularity measure (Q)

Usage

```
louvain(A, gamma, M0)
```

Arguments

`A`	An adjacency matrix of network data
`gamma`	Defaults to `1`. Set to `gamma` > 1 to detect smaller modules and `gamma` < 1 for larger modules
`M0`	Input can be an initial community vector. Defaults to `NULL`

Value

Returns a list containing:

`community`	A community vector corresponding to each node's community
`Q`	Modularity statistic. A measure of how well the communities are compartmentalized

Author(s)

Alexander Christensen <alexpaulchristensen@gmail.com>

References

Blondel, V. D., Guillaume, J. L., Lambiotte, R., & Lefebvre, E. (2008). Fast unfolding of communities in large networks. *Journal of Statistical Mechanics: Theory and Experiment*, *2008*, P10008.

Rubinov, M., & Sporns, O. (2010). Complex network measures of brain connectivity: Uses and interpretations. *NeuroImage*, *52*, 1059-1069.

Examples

```
# Pearson's correlation only for CRAN checks
A <- TMFG(neoOpen, normal = FALSE)$A

modularity <- louvain(A)
```

`MaST` *Maximum Spanning Tree*

Description

Applies the Maximum Spanning Tree (MaST) filtering method

Usage

```
MaST(
  data,
  normal = TRUE,
  na.data = c("pairwise", "listwise", "fiml", "none"),
  depend = FALSE
)
```

Arguments

`data`	Can be a dataset or a correlation matrix
`normal`	Should data be transformed to a normal distribution? Input must be a dataset. Defaults to `TRUE`. Computes correlations using the `cor_auto` function. Set to `FALSE` for Pearson's correlation
`na.data`	How should missing data be handled? For `"listwise"` deletion the `na.omit` function is applied. Set to `"fiml"` for Full Information Maximum Likelihood (`corFiml`). Full Information Maximum Likelihood is **recommended** but time consuming
`depend`	Is network a dependency (or directed) network? Defaults to `FALSE`. Set `TRUE` to generate a MaST-filtered dependency network (output obtained from the `depend` function)

Value

A sparse association matrix

Author(s)

Alexander Christensen <alexpaulchristensen@gmail.com>

Examples

```
# Pearson's correlation only for CRAN checks
MaST.net <- MaST(neoOpen, normal = FALSE)
```

`MFCF` *Maximally Filtered Clique Forest*

Description

Applies the Maximally Filtered Clique Forest (MFCF) filtering method (**Please see and cite Massara & Aste**).

Usage

```
MFCF(
  data,
  cases = NULL,
  na.data = c("pairwise", "listwise", "fiml", "none"),
  time.series = FALSE,
  gain.fxn = c("logLik", "logLik.val", "rSquared.val"),
  min_size = 0,
  max_size = 8,
  pval = 0.05,
```

```
  pen = 0,
  drop_sep = FALSE,
  use_returns = FALSE
)
```

Arguments

`data`	Matrix (n x n or p x n) or data frame. Can be a dataset or a correlation matrix
`cases`	Numeric. If `data` is a (partial) correlation matrix, then number of cases must be input. Defaults to `NULL`
`na.data`	Character. How should missing data be handled? • `"listwise"` Removes case if **any** missing data exists. Applies `na.omit` • `"pairwise"` Estimates correlations using the available data for each variable • `"fiml"` Estimates correlations using the Full Information Maximum Likelihood. Recommended and most robust but time consuming • `"none"` Default. No missing data or missing data has been handled by the user
`time.series`	Boolean. Is `data` a time-series dataset? Defaults to `FALSE`. Set to `TRUE` to handle time-series data (n x p)
`gain.fxn`	Character. Gain function to be used for inclusion of nodes in cliques. There are several options available (see `gain.functions` for more details): `"logLik"`, `"logLik.val"`, `"rSquared.val"`. Defaults to `"rSquared.val"`
`min_size`	Numeric. Minimum number of nodes allowed per clique. Defaults to `0`
`max_size`	Numeric. Maximum number of nodes allowed per clique. Defaults to `8`
`pval`	Numeric. *p*-value used to determine cut-offs for nodes to include in a clique
`pen`	Numeric. Multiplies the number of edges added to penalise complex models. Similar to the penalty term in AIC
`drop_sep`	Boolean. This parameter influences the MFCF only. Defaults to `FALSE`. If `TRUE`, then any separator can be used only once (similar to the `TMFG`)
`use_returns`	Boolean. Only used in `"gain.fxn = rSquared.val"`. If set to `TRUE` the regression is performed on log-returns. Defaults to `FALSE`

Value

Returns a list containing:

`A`	MFCF filtered partial correlation network (adjacency matrix)
`J`	MFCF filtered inverse covariance matrix (precision matrix)
`cliques`	Cliques in the network (output for `LoGo`)
`separators`	Separators in the network (output for `LoGo`)

Author(s)

Guido Previde Massara <gprevide@gmail.com> and Alexander Christensen <alexpaulchristensen@gmail.com>

References

Massara, G. P. & Aste, T. (2019). Learning clique forests. *ArXiv*.

Examples

```
# Load data
data <- neoOpen

## Not run:
# Use polychoric correlations and R-squared method
MFCF.net <- MFCF(qgraph::cor_auto(data), cases = nrow(neoOpen))$A


## End(Not run)
```

`neoOpen` *NEO-PI-3 Openness to Experience Data*

Description

A response matrix (*n* = 802) of NEO-PI-3's Openness to Experience from Christensen, Cotter, & Silvia (2019).

Usage

```
data(neoOpen)
```

Format

A 802x48 response matrix

References

Christensen, A. P., Cotter, K. N., & Silvia, P. J. (2019). Reopening openness to experience: A network analysis of four openness to experience inventories. *Journal of Personality Assessment*, *101*, 574-588.

Examples

```
data("neoOpen")
```

`network.coverage` *Network Coverage*

Description

Computes the mean distance across a subset of nodes in a network. This measure can be used to identify the effectiveness of a subset of nodes' coverage of the network space

Usage

```
network.coverage(A, nodes, weighted = FALSE)
```

Arguments

`A`	An adjacency matrix
`nodes`	Subset of nodes to examine the coverage of the network
`weighted`	Is the network weighted? Defaults to `FALSE`. Set to `TRUE` for weighted measures

Value

Returns a list containing:

`mean`	The average distance from the subset of nodes to all other nodes in the network
`sd`	The standard deviation of distance from the subset of nodes to all other nodes in the network
`range`	The range of distance from the subset of nodes to all other nodes in the network

Author(s)

Alexander Christensen <alexpaulchristensen@gmail.com> and Mathias Benedek <mathias.benedek@uni-graz.at>

References

Christensen, A. P., Cotter, K. N., Silvia, P. J., & Benedek, M. (2018) Scale development via network analysis: A comprehensive and concise measure of Openness to Experience *PsyArXiv*, 1-40.

Examples

```
# Pearson's correlation only for CRAN checks
A <- TMFG(neoOpen, normal = FALSE)$A

nodes <- seq(1,48,2)

result <- network.coverage(A, nodes)
```

`network.permutation` *Permutation Test for Network Measures*

Description

Computes a permutation test to determine whether there are difference in centrality and global network measures

Usage

```
network.permutation(
  sample1 = NULL,
  sample2 = NULL,
  iter,
  network = c("glasso", "ising", "TMFG", "LoGo"),
 measure = c("betweenness", "closeness", "strength", "rspbc", "hybrid", "ASPL", "CC",
    "S", "Q"),
  alternative = c("less", "greater", "two.tailed"),
  ncores,
  prev.perm = NULL
)
```

Arguments

`sample1`	Matrix or data frame. Sample to be compared with `sample2`
`sample2`	Matrix or data frame. Sample to be compared with `sample1`
`iter`	Numeric. Number of iterations to perform. Defaults to `1000`
`network`	Character. Network estimation method to apply to the datasets. Defaults to `"glasso"`
`measure`	Character. Network measure to be compared in the permutation test
`alternative`	Character. Alternative hypothesis test to perform. Defaults to `"two.tailed"`
`ncores`	Numeric. Number of computer processing cores to use for bootstrapping samples. Defaults to *n* - 1 total number of cores. Set to any number between 1 and maximum amount of cores on your computer (see `parellel::detectCores()`)
`prev.perm`	`network.permutation` class object. An object of previously performed permutation test. The networks generated in the previous permutation will be used to compute other network measures. This saves time when computing multiple permutation tests

Value

Returns a list containing two objects:

`result`	The results of the permutation test. For centrality measures, this is a matrix where the rows represent each node and the columns are the observed values of the centrality measure for `sample1`, `sample2`, and the *p*-value from the permutation test. For global network measures, this is a vector with the observed values of the global network measure for `sample1`, `sample2`, and the *p*-value from the permutation test.
`networks`	A list containing two lists: `network1` and `network2`. The network lists correspond to the networks generated in the permutation test for `sample1` and `sample2`, respectively. This output is used primarily for the computation of other network measures using the same datasets (see `prev.perm` explanation)

Author(s)

Alexander Christensen <alexpaulchristensen@gmail.com>

Examples

```
# Split data (only for example)
split1 <- neoOpen[c(1:401),]
split2 <- neoOpen[c(402:802),]


# Perform permutation test
perm.str <- network.permutation(split1, split2, iter = 1000, network = "glasso",
measure = "strength", alternative = "two.tailed", ncores = 2)

# Check results
perm.str$result

# Permutation to check other measures (using networks from previous result)
perm.aspl <- network.permutation(prev.perm = perm.str, measure = "ASPL", ncores = 2)

# Check results
perm.aspl$result
```

`neuralnetfilter` *Neural Network Filter*

Description

Applies a network filtering methodology to neural network array. Removes edges from the neural network output from `convertConnBrainMat` using a network filtering approach

Usage

```
neuralnetfilter(
  neuralarray,
  method = c("TMFG", "MaST", "ECOplusMaST", "ECO", "threshold"),
  progBar = TRUE,
  ...
)
```

Arguments

`neuralarray`	Array from `convertConnBrainMat` function
`method`	Filtering method to be applied
`progBar`	Should progress bar be displayed? Defaults to `TRUE`. Set `FALSE` for no progress bar
`...`	Additional arguments from network filtering methods

Value

Returns an array of n x n x m filtered matrices

Author(s)

Alexander Christensen <alexpaulchristensen@gmail.com>

Examples

```
## Not run: neuralarray <- convertConnBrainMat()

filteredneuralarray <- neuralnetfilter(neuralarray, method = "threshold", thresh = .50)

dependencyarray <- depna(neuralarray)

filtereddependencyarray <- neuralnetfilter(dependencyarray, method = "TMFG", depend = TRUE)

## End(Not run)
```

`node.multidimensional` *Detects Node Crossings in a Network*

Description

UNDER DEVELOPMENT. Computes `rspbc` for connections between dimensions in a network. Multidimensional nodes can be detected

Usage

```
node.multidimensional(A, comm, plot = FALSE)
```

Arguments

`A`	An adjacency matrix of network data
`comm`	A vector or matrix corresponding to the community each node belongs to
`plot`	Should a plot be produced?

Value

Produces a list containing:

Author(s)

Alexander Christensen <alexpaulchristensen@gmail.com>

Examples

```
# Pearson's correlation only for CRAN checks
A <- TMFG(neoOpen, normal = FALSE)$A

# Obtain communities via walktrap algorithm
comm <- igraph::walktrap.community(convert2igraph(abs(A)))$membership

## Not run:
result <- node.multidimensional(A = A, comm = comm$wc, plot = FALSE)

## End(Not run)
```

`participation` *Participation Coefficient*

Description

Computes the participation coefficient for each node. The participation coefficient measures the strength of a node's connections within its community. Positive and negative signed weights for participation coefficients are computed separately.

Usage

```
participation(A, comm = c("walktrap", "louvain"))
```

Arguments

`A`	Network adjacency matrix
`comm`	A vector of corresponding to each item's community. Defaults to `"walktrap"` for the `cluster_walktrap` community detection algorithm. Set to `"louvain"` for the `louvain` community detection algorithm. Can also be set to user-specified communities (see examples)

Details

Values closer to 1 suggest greater within-community connectivity and values closer to 0 suggest greater between-community connectivity

Value

Returns a list containing:

`overall`	Participation coefficient without signs considered
`positive`	Participation coefficient with only positive sign
`negative`	Participation coefficient with only negative sign

Author(s)

Alexander Christensen <alexpaulchristensen@gmail.com>

References

Guimera, R., & Amaral, L. A. N. (2005). Functional cartography of complex metabolic networks. *Nature*, *433*, 895-900.

Rubinov, M., & Sporns, O. (2010). Complex network measures of brain connectivity: Uses and interpretations. *NeuroImage*, *52*, 1059-1069.

Examples

```
#theoretical factors
comm <- rep(1:8, each = 6)

# Pearson's correlation only for CRAN checks
A <- TMFG(neoOpen, normal = FALSE)$A

pc <- participation(A, comm = comm)

# Walktrap factors
wpc <- participation(A, comm = "walktrap")
```

`pathlengths` *Characteristic Path Lengths*

Description

Computes global average shortest path length, local average shortest path length, eccentricity, and diameter of a network

Usage

```
pathlengths(A, weighted = FALSE)
```

Arguments

`A`	An adjacency matrix of network data
`weighted`	Is the network weighted? Defaults to `FALSE`. Set to `TRUE` for weighted measures

Value

Returns a list containing:

`ASPL`	Global average shortest path length
`ASPLi`	Local average shortest path length
`ecc`	Eccentricity (i.e., maximal shortest path length between a node and any other node)
`D`	Diameter of the network (i.e., the maximum of eccentricity)

Author(s)

Alexander Christensen <alexpaulchristensen@gmail.com>

References

Rubinov, M., & Sporns, O. (2010). Complex network measures of brain connectivity: Uses and interpretations. *NeuroImage*, *52*, 1059-1069.

Examples

```
# Pearson's correlation only for CRAN checks
A <- TMFG(neoOpen, normal = FALSE)$A

#Unweighted
PL <- pathlengths(A)

#Weighted
PL <- pathlengths(A, weighted = TRUE)
```

`randnet` *Generates a Random Network*

Description

Generates a random network

Usage

```
randnet(nodes, edges)
```

Arguments

`nodes`	Number of nodes in random network
`edges`	Number of edges in random network

Value

Returns an adjacency matrix of a random network

Author(s)

Alexander Christensen <alexpaulchristensen@gmail.com>

References

Rubinov, M., & Sporns, O. (2010). Complex network measures of brain connectivity: Uses and interpretations. *NeuroImage*, *52*, 1059-1069.

Examples

```
rand <- randnet(10, 27)
```

`reg` *Regression Matrix*

Description

Computes regression such that one variable is regressed over all other variables

Usage

```
reg(
  data,
  family = c("binomial", "gaussian", "Gamma", "poisson"),
  symmetric = TRUE
)
```

Arguments

`data`	A dataset
`family`	Error distribution to be used in the regression model. Defaults to `"logistic"`. Set to any family used in function `family`
`symmetric`	Should matrix be symmetric? Defaults to `TRUE`, taking the mean of the two edge weights (i.e., `[i,j]` and `[j,i]`) Set to `FALSE` for asymmetric weights (i.e., `[i,j]` does not equal `[j,i]`)

Value

A matrix of fully regressed coefficients where one variable is regressed over all others

Author(s)

Alexander Christensen <alexpaulchristensen@gmail.com>

Examples

```
#binarize responses
psyb <- ifelse(neoOpen>=4, 1, 0)

#perform logistic regression
mat <- reg(psyb)
```

`resp.rep` *Repeated Responses Check*

Description

Screens data to identify potential cases of repeated responding. The function is based on two criteria: no variance (i.e., a standard deviation of zero for given responses)and frequency proportion of the response values (which is set by `freq.prop`). Note that these criteria are highly related. Additional criteria will be added in the future.

Usage

```
resp.rep(data, scale.lens = NULL, max.val, reverse = NULL, freq.prop = 0.8)
```

Arguments

`data`	A dataset
`scale.lens`	The number of items for each scale in the data. A vector indicating the length for each scale to be checked in the data
`max.val`	Maximum value for data (or scales). If scales have different maximum values, then a vector must be input with each scale's maximum value (see examples)
`reverse`	Reverse scored responses. If responses have not yet reversed, then do not reverse them. If responses have been reversed, then a vector indicating which responses have been reverse-scored should be input (see examples). Can be TRUE/FALSE or 1/0 (reversed/not reversed)
`freq.prop`	Frequency proportion of the response values. Allows the researcher to determine the maximum frequency proportion of a certain response value is suspicious. The default is set to `.80` (or 80 percent responses are a single value)

Details

If a case is returned, then it does not mean that it is a bad case. Researchers should thoroughly inspect each case that is returned. A general guideline is that if a participant responded with all middle values (e.g., all 3's on a 5-point Likert scale), then they should be dropped. Note that a participant who responds with all maximum or minimum values may be a real case or a bad case. It is up to the researcher to decide and justify why or why not a case is kept.

Value

Returns a matrix when `scale.lens = NULL` and a list with elements corresponding to the order of scales. In general, the output contains potential bad cases that should be further inspected by the researcher.

Author(s)

Alexander Christensen <alexpaulchristensen@gmail.com>

Examples

```
#Re-reverse responses
rev.vec <- c(TRUE,FALSE,TRUE,FALSE,TRUE,TRUE,TRUE,FALSE,TRUE,FALSE,
TRUE,FALSE,FALSE,TRUE,FALSE,TRUE,FALSE,TRUE,TRUE,FALSE,TRUE,FALSE,TRUE,
FALSE,FALSE,TRUE,FALSE,TRUE,FALSE,TRUE,TRUE,FALSE,TRUE,FALSE,TRUE,
FALSE,TRUE,FALSE,FALSE,TRUE,FALSE,TRUE,TRUE,FALSE,FALSE,TRUE,FALSE,TRUE)

#Maximum value (5-point Likert scale)
mv.vec <- 5

#Repeated responses check
resp.rep(neoOpen, reverse = rev.vec, max.val = mv.vec)

#Example with multiple scales

#Facet scale lengths of NEO-PI-3 Openness to Experience
s.len <- c(8, 8, 8, 8, 8, 8)

#Maximum values
mv.vec <- c(5, 5, 5, 5, 5, 5)

#Re-reverse responses
rev.vec <- c(TRUE,FALSE,TRUE,FALSE,TRUE,TRUE,TRUE,FALSE,TRUE,FALSE,
TRUE,FALSE,FALSE,TRUE,FALSE,TRUE,FALSE,TRUE,TRUE,FALSE,TRUE,FALSE,TRUE,
FALSE,FALSE,TRUE,FALSE,TRUE,FALSE,TRUE,TRUE,FALSE,TRUE,FALSE,TRUE,
FALSE,TRUE,FALSE,FALSE,TRUE,FALSE,TRUE,TRUE,FALSE,FALSE,TRUE,FALSE,TRUE)

#Repeated responses check
resp.rep(neoOpen, scale.lens = s.len, max.val = mv.vec, reverse = rev.vec)
```

`rmse` *Root Mean Square Error*

Description

Computes the root mean square error (RMSE) of a sparse model to a full model

Usage

```
rmse(base, test)
```

Arguments

`base`	Base (or full) model to be evaluated against
`test`	Reduced (or testing) model (e.g., a sparse correlation or covariance matrix)

Value

RMSE value (lower values suggest more similarity between the full and sparse model)

Author(s)

Alexander Christensen <alexpaulchristensen@gmail.com>

Examples

```
A1 <- solve(cov(neoOpen))

## Not run:
A2 <- LoGo(neoOpen)

root <- rmse(A1, A2)

## End(Not run)
```

`rspbc` *Randomized Shortest Paths Betweenness Centrality*

Description

Computes betweenness centrality based on randomized shortest paths of each node in a network (**Please see and cite Kivimaki et al., 2016**)

Usage

```
rspbc(A, beta = 0.01, comm = NULL)
```

Arguments

`A`	An adjacency matrix of network data
`beta`	Sets the beta parameter. Defaults to `0.01` (recommended). Beta > 0.01 measure gets closer to weighted betweenness centrality (10) and beta < 0.01 measure gets closer to degree (.0001)
`comm`	Vector. Community vector containing a value for each node. Computes "bridge" RSPBC, where the number of times a node is used on a random path between to another community

Value

A vector of randomized shortest paths betweenness centrality values for each node in the network

Author(s)

Alexander Christensen <alexpaulchristensen@gmail.com>

References

Kivimaki, I., Lebichot, B., Saramaki, J., & Saerens, M. (2016). Two betweenness centrality measures based on Randomized Shortest Paths. *Scientific Reports*, *6*, 19668.

Examples

```
# Pearson's correlation only for CRAN checks
A <- TMFG(neoOpen, normal = FALSE)$A

rspbc <- rspbc(A, beta=0.01)
```

`sim.chordal` *Simulate Chordal Network*

Description

Simulates a chordal network based on number of nodes. Data will also be simulated based on the true network structure

Usage

```
sim.chordal(
  nodes,
  inverse = c("cases", "matrix"),
  n = NULL,
  ordinal = FALSE,
  ordLevels = NULL,
  idio = NULL,
  eps = NULL
)
```

Arguments

`nodes`	Numeric. Number of nodes in the simulated network
`inverse`	Character. Method to produce inverse covariance matrix. • `"cases"`Estimates inverse covariance matrix based on `n` number of cases and `nodes` number of variables, which are drawn from a random normal distribution `rnorm`. Data generated will be continuous unless `ordinal` is set to `TRUE` • `"matrix"`Estimates inverse covariance matrix based on ??eps??
`n`	Numeric. Number of cases in the simulated dataset
`ordinal`	Boolean. Should simulated continuous data be converted to ordinal? Defaults to `FALSE`. Set to `TRUE` for simulated ordinal data
`ordLevels`	Numeric. If `ordinal` = `TRUE`, then how many levels should be used? Defaults to 5. Set to desired number of intervals
`idio`	Numeric. DESCRIPTION. Defaults to `0.10`
`eps`	Numeric. DESCRIPTION. Defaults to `2`

Value

Returns a list containing:

`cliques`	The cliques in the network
`separators`	The separators in the network
`inverse`	Simulated inverse covariance matrix of the network
`data`	Simulated data from sim.correlation in the `psych` package based on the simulated network

Author(s)

Guido Previde Massara <gprevide@gmail.com>

References

Massara, G. P. & Aste, T. (2019). Learning clique forests. *ArXiv*.

Examples

```
#Continuous data
sim.Norm <- sim.chordal(nodes = 20, inverse = "cases", n = 1000)

#Ordinal data
sim.Likert <- sim.chordal(nodes = 20, inverse = "cases", n = 1000, ordinal = TRUE)

#Dichotomous data
sim.Binary <- sim.chordal(nodes = 20, inverse = "cases", n = 1000, ordinal = TRUE, ordLevels = 5)
```

`sim.swn` *Simulate Small-world Network*

Description

Simulates a small-world network based on specified topological properties. Data will also be simulated based on the true network structure

Usage

```
sim.swn(
  nodes,
  n,
  pos = 0.8,
  ran = c(0.3, 0.7),
  nei = 1,
  p = 0.5,
  corr = FALSE,
  replace = NULL,
  ordinal = FALSE,
  ordLevels = NULL
)
```

Arguments

`nodes`	Number of nodes in the simulated network
`n`	Number of cases in the simulated dataset
`pos`	Proportion of positive correlations in the simulated network
`ran`	Range of correlations in the simulated network
`nei`	Adjusts the number of connections each node has to neighboring nodes (see `sample_smallworld`)
`p`	Adjusts the rewiring probability (default is .5). p > .5 rewires the simulated network closer to a random network. p < .5 rewires the simulated network closer to a lattice network
`corr`	Should the simulated network be a correlation network? Defaults to FALSE. Set to TRUE for a simulated correlation network
`replace`	If noise > 0, then should participants be sampled with replacement? Defaults to TRUE. Set to FALSE to not allow the potential for participants to be consecutively entered into the simulated dataset.
`ordinal`	Should simulated continuous data be converted to ordinal? Defaults to FALSE. Set to TRUE for simulated ordinal data
`ordLevels`	If ordinal = TRUE, then how many levels should be used? Defaults to NULL. Set to desired number of intervals (defaults to 5)

Value

Returns a list containing:

`simNetwork`	Adjacency matrix of the simulated network
`simData`	Simulated data from sim.correlation in the `psych` package based on the simulated network
`simRho`	Simulated correlation from sim.correlation in the `psych` package

Author(s)

Alexander Christensen <alexpaulchristensen@gmail.com>

References

Csardi, G., & Nepusz, T. (2006). The *igraph* software package for complex network research. *InterJournal, Complex Systems*, *1695*, 1-9.

Examples

```
#Continuous data
sim.Norm <- sim.swn(25, 500, nei = 3)

#Ordinal data
sim.Likert <- sim.swn(25, 500, nei = 3, replace = TRUE, ordinal = TRUE, ordLevels = 5)

#Dichotomous data
sim.Binary <- sim.swn(25, 500, nei = 3, replace = TRUE, ordinal = TRUE, ordLevels = 2)
```

`smallworldness` *Small-worldness Measure*

Description

Computes the small-worldness measure of a network

Usage

```
smallworldness(
  A,
  iter = 100,
  progBar = FALSE,
  method = c("HG", "rand", "TJHBL")
)
```

Arguments

`A`	An adjacency matrix of network data
`iter`	Number of random (or lattice) networks to generate, which are used to calculate the mean random ASPL and CC (or lattice)
`progBar`	Defaults to `FALSE`. Set to `TRUE` to see progress bar
`method`	Defaults to `"HG"` (Humphries & Gurney, 2008). Set to `"rand"` for the CC to be calculated using a random network or set to `"TJHBL"` for (Telesford et al., 2011) where CC is calculated from a lattice network

Details

For `"rand"`, values > 1 indicate a small-world network. For `"HG"`, values > 3 indicate a small-world network. For `"TJHBL"`, values near 0 indicate a small-world network, while < 0 indicates a more regular network and > 0 indicates a more random network

Value

Returns a list containing:

`swm`	Small-worldness value
`rASPL`	Global average shortest path length from random network
`lrCCt`	When `"rand"`, clustering coefficient from a random network. When `"HG"`, transitivity from a random network. When `"TJHBL"`, clustering coefficient from a lattice network

Author(s)

Alexander Christensen <alexpaulchristensen@gmail.com>

References

Humphries, M. D., & Gurney, K. (2008). Network 'small-world-ness': A quantitative method for determining canonical network equivalence. *PLoS one*, *3*, e0002051.

Telesford, Q. K., Joyce, K. E., Hayasaka, S., Burdette, J. H., & Laurienti, P. J. (2011). The ubiquity of small-world networks. *Brain Connectivity*, *1*(5), 367-375.

Examples

```
# Pearson's correlation only for CRAN checks
A <- TMFG(neoOpen, normal = FALSE)$A

swmHG <- smallworldness(A, method="HG")

swmRand <- smallworldness(A, method="rand")

swmTJHBL <- smallworldness(A, method="TJHBL")
```

`stable` *Stabilizing Nodes*

Description

Computes the within-community centrality for each node in the network

Usage

```
stable(
  A,
  comm = c("walktrap", "louvain"),
  cent = c("betweenness", "rspbc", "closeness", "strength", "degree", "hybrid"),
  absolute = TRUE,
  diagonal = 0,
  ...
)
```

Arguments

`A`	An adjacency matrix of network data
`comm`	Can be a vector of community assignments or community detection algorithms (`"walktrap"` or `"louvain"`) can be used to determine the number of factors. Defaults to `"walktrap"`. Set to `"louvain"` for `louvain` community detection
`cent`	Centrality measure to be used. Defaults to `"strength"`.
`absolute`	Should network use absolute weights? Defaults to `TRUE`. Set to `FALSE` for signed weights
`diagonal`	Sets the diagonal values of the `A` input. Defaults to `0`
`...`	Additional arguments for `cluster_walktrap` and `louvain` community detection algorithms

Value

A matrix containing the within-community centrality value for each node

Author(s)

Alexander Christensen <alexpaulchristensen@gmail.com>

References

Blanken, T. F., Deserno, M. K., Dalege, J., Borsboom, D., Blanken, P., Kerkhof, G. A., & Cramer, A. O. (2018). The role of stabilizing and communicating symptoms given overlapping communities in psychopathology networks. *Scientific Reports*, *8*, 5854.

Examples

```
# Pearson's correlation only for CRAN checks
A <- TMFG(neoOpen, normal = FALSE)$A

stabilizing <- stable(A, comm = "walktrap")
```

`strength` *Node Strength*

Description

Computes strength of each node in a network

Usage

```
strength(A, absolute = TRUE)
```

Arguments

`A`	An adjacency matrix of network data
`absolute`	Should network use absolute weights? Defaults to `TRUE`. Set to `FALSE` for signed weights

Value

A vector of strength values for each node in the network.

If directed network, returns a list containing:

`inStrength`	Strength of incoming edges (pointing to the node)
`outStrength`	Strength of outgoing edges (pointing away from the node)
`relInf`	Relative degree of incoming and outgoing edges. Positive values indicate more outgoing strength relative to incoming strength. Negative values indicate more incoming strength relative to outgoing strength

Author(s)

Alexander Christensen <alexpaulchristensen@gmail.com>

References

Rubinov, M., & Sporns, O. (2010). Complex network measures of brain connectivity: Uses and interpretations. *NeuroImage*, *52* 1059-1069.

Examples

```
# Pearson's correlation only for CRAN checks
A <- TMFG(neoOpen, normal = FALSE)$A

str <- strength(A)

#Directed network
## Not run:
dep <- depend(neoOpen)

Adep <- TMFG(dep, depend = TRUE)$A

str <- strength(Adep)

## End(Not run)
```

`threshold` *Threshold Network Estimation Methods*

Description

Filters the network based on an r-value, alpha, adaptive alpha, bonferroni, false-discovery rate (FDR), or proportional density (fixed number of edges) value

Usage

```
threshold(
  data,
  a,
  thresh = c("alpha", "adaptive", "bonferroni", "FDR", "proportional"),
  normal = FALSE,
  na.data = c("pairwise", "listwise", "fiml", "none"),
  ...
)
```

Arguments

`data`	Can be a dataset or a correlation matrix
`a`	When `thresh` = `"alpha"`, `"adaptive"`, and `"bonferroni"` an α threshold is applied (defaults to `.05`). For `"adaptive"`, beta (Type II error) is set to $\alpha * 5$ for a medium effect size (*r* = .3). When `thresh` = `"FDR"`, a q-value threshold is applied (defaults to `.10`). When `thresh` = `"proportional"`, a density threshold is applied (defaults to `.15`)
`thresh`	Sets threshold. Defaults to `"alpha"`. Set to any value 0> *r* >1 to retain values greater than set value, `"adaptive"` for an `adapt.a` based on sample size (Perez & Pericchi, 2014), `"bonferroni"` for the bonferroni correction on alpha, `"FDR"` for local false discovery rate, and `"proportional"` for a fixed density of edges (keeps strongest correlations within density)

`normal`	Should data be transformed to a normal distribution? Defaults to FALSE. Data is not transformed to be normal. Set to TRUE if data should be transformed to be normal (computes correlations using the cor_auto function)
`na.data`	How should missing data be handled? For `"listwise"` deletion the `na.omit` function is applied. Set to `"fiml"` for Full Information Maximum Likelihood (`corFiml`). Full Information Maximum Likelihood is **recommended** but time consuming
`...`	Additional arguments for `fdrtool` and `adapt.a`

Value

Returns a list containing:

`A`	The filtered adjacency matrix
`r.cv`	The critical correlation value used to filter the network

Author(s)

Alexander Christensen <alexpaulchristensen@gmail.com>

References

Strimmer, K. (2008). fdrtool: A versatile R package for estimating local and tail area-based false discovery rates. *Bioinformatics*, *24*, 1461-1462.

Examples

```
threshnet<-threshold(neoOpen)

alphanet<-threshold(neoOpen, thresh = "alpha", a = .05)

bonnet<-threshold(neoOpen, thresh = "bonferroni", a = .05)

FDRnet<-threshold(neoOpen, thresh = "FDR", a = .10)

propnet<-threshold(neoOpen, thresh = "proportional", a = .15)
```

`TMFG` *Triangulated Maximally Filtered Graph*

Description

Applies the Triangulated Maximally Filtered Graph (TMFG) filtering method (**Please see and cite Massara et al., 2016**). The TMFG method uses a structural constraint that limits the number of zero-order correlations included in the network (3n - 6; where n is the number of variables). The TMFG algorithm begins by identifying four variables which have the largest sum of correlations to all other variables. Then, it iteratively adds each variable with the largest sum of three correlations to nodes already in the network until all variables have been added to the network. This structure can be associated with the inverse correlation matrix (i.e., precision matrix) to be turned into a GGM (i.e., partial correlation network) by using `LoGo`. See Details for more information on this network estimation method.

Usage

```
TMFG(
  data,
  normal = TRUE,
  na.data = c("pairwise", "listwise", "fiml", "none"),
  depend = FALSE
)
```

Arguments

`data`	Can be a dataset or a correlation matrix
`normal`	Should data be transformed to a normal distribution? Input must be a dataset. Defaults to `TRUE`. Computes correlations using the `cor_auto` function. Set to `FALSE` for Pearson's correlation
`na.data`	How should missing data be handled? For `"listwise"` deletion the `na.omit` function is applied. Set to `"fiml"` for Full Information Maximum Likelihood (`corFiml`). Full Information Maximum Likelihood is **recommended** but time consuming
`depend`	Is network a dependency (or directed) network? Defaults to `FALSE`. Set to `TRUE` to generate a TMFG-filtered dependency network (output obtained from the `depend` function)

Details

The TMFG method applies a structural constraint on the network, which restrains the network to retain a certain number of edges ($3n$-6, where n is the number of nodes; Massara et al., 2016). The network is also composed of 3- and 4-node cliques (i.e., sets of connected nodes; a triangle and tetrahedron, respectively). The TMFG method constructs a network using zero-order correlations and the resulting network can be associated with the inverse covariance matrix (yielding a GGM; Barfuss, Massara, Di Matteo, & Aste, 2016). Notably, the TMFG can use any association measure and thus does not assume the data is multivariate normal.

Construction begins by forming a tetrahedron of the four nodes that have the highest sum of correlations that are greater than the average correlation in the correlation matrix. Next, the algorithm iteratively identifies the node that maximizes its sum of correlations to a connected set of three nodes (triangles) already included in the network and then adds that node to the network. The process is completed once every node is connected in the network. In this process, the network automatically generates what's called a planar network. A planar network is a network that could be drawn on a sphere with no edges crossing (often, however, the networks are depicted with edges crossing; Tumminello, Aste, Di Matteo, & Mantegna, 2005).

Value

Returns a list containing:

`A`	The filtered adjacency matrix
`separators`	The separators (3-cliques) in the network (wrapper output for `LoGo`)
`cliques`	The cliques (4-cliques) in the network (wrapper output for `LoGo`)

Author(s)

Alexander Christensen <alexpaulchristensen@gmail.com>

References

Christensen, A. P., Kenett, Y. N., Aste, T., Silvia, P. J., & Kwapil, T. R. (2018). Network structure of the Wisconsin Schizotypy Scales-Short Forms: Examining psychometric network filtering approaches. *Behavior Research Methods*, *50*, 2531-2550.

Massara, G. P., Di Matteo, T., & Aste, T. (2016). Network filtering for big data: Triangulated maximally filtered graph. *Journal of Complex Networks*, *5*, 161-178.

Examples

```
# Pearson's correlation only for CRAN checks
A <- TMFG(neoOpen, normal = FALSE)$A
```

`transitivity` *Transitivity*

Description

Computes transitivity of a network

Usage

```
transitivity(A, weighted = FALSE)
```

Arguments

`A`	An adjacency matrix of network data
`weighted`	Is the network weighted? Defaults to `FALSE`. Set to `TRUE` for a weighted measure of transitivity

Value

Returns a value of transitivity

Author(s)

Alexander Christensen <alexpaulchristensen@gmail.com>

References

Rubinov, M., & Sporns, O. (2010). Complex network measures of brain connectivity: Uses and interpretations. *NeuroImage*, *52*, 1059-1069.

Examples

```
# Pearson's correlation only for CRAN checks
A <- TMFG(neoOpen, normal = FALSE)$A

trans <- transitivity(A, weighted=TRUE)
```

`un.direct` *Convert Directed Network to Undirected Network*

Description

Converts a directed network to an undirected network

Usage

```
un.direct(A, diagonal = 0)
```

Arguments

`A`	Matrix or data frame. Adjacency matrix (network matrix)
`diagonal`	Numeric. Number to be placed on the diagonal. Defaults to `0`

Value

Returns a symmetric adjacency matrix

Author(s)

Alexander Christensen <alexpaulchristensen@gmail.com>

Examples

```
# Pearson's correlation only for CRAN checks
A <- TMFG(neoOpen, normal = FALSE)$A

# create a directed network
dir <- A * sample(c(0,1), size = length(A), replace = TRUE)

# undirect the directed network
undir <- un.direct(dir)
```

Index