Package 'EGAnet'

November 9, 2024

Title Exploratory Graph Analysis – a Framework for Estimating the Number of Dimensions in Multivariate Data using Network Psychometrics

Version 2.1.0

Date 2024-11-09

Maintainer Hudson Golino <hfg9s@virginia.edu>

Description Implements the Exploratory Graph Analysis (EGA) framework for dimensionality and psychometric assessment. EGA estimates the number of dimensions in psychological data using network estimation methods and community detection algorithms. A bootstrap method is provided to assess the stability of dimensions and items. Fit is evaluated using the Entropy Fit family of indices. Unique Variable Analysis evaluates the extent to which items are locally dependent (or redundant). Network loadings provide similar information to factor loadings and can be used to compute network scores. A bootstrap and permutation approach are available to assess configural and metric invariance. Hierarchical structures can be detected using Hierarchical EGA. Time series and intensive longitudinal data can be analyzed using Dynamic EGA, supporting individual, group, and population level assessments.

Depends R (>= 3.5.0)

License GPL (>= 3.0)

Encoding UTF-8

LazyData true

Imports dendextend, fungible, future, future.apply, glasso, GGally, ggplot2, ggpubr, GPArotation, igraph (>= 1.3.0), lavaan, Matrix, methods, network, progressr, qgraph, semPlot, sna, stats

Suggests fitdistrplus, gridExtra, knitr, markdown, pbapply, progress, psych, pwr, RColorBrewer

URL https://r-ega.net

BugReports https://github.com/hfgolino/EGAnet/issues

RoxygenNote 7.3.2

NeedsCompilation yes

Author Hudson Golino [aut, cre] (<https://orcid.org/0000-0002-1601-1447>),
Alexander Christensen [aut] (<https://orcid.org/0000-0002-9798-7037>),
Robert Moulder [ctb] (<https://orcid.org/0000-0001-7504-9560>),
Luis E. Garrido [ctb] (<https://orcid.org/0000-0001-8932-6063>),
Laura Jamison [ctb] (<https://orcid.org/0000-0002-4656-8684>),
Dingjing Shi [ctb] (<https://orcid.org/0000-0002-5652-3818>)

Repository CRAN

Date/Publication 2024-11-09 19:00:02 UTC

Contents

`EGAnet-package` *EGAnet-package*

Description

Implements the Exploratory Graph Analysis (EGA) framework for dimensionality and psychometric assessment. EGA estimates the number of dimensions in psychological data using network estimation methods and community detection algorithms. A bootstrap method is provided to assess the stability of dimensions and items. Fit is evaluated using the Entropy Fit family of indices. Unique Variable Analysis evaluates the extent to which items are locally dependent (or redundant). Network loadings provide similar information to factor loadings and can be used to compute network scores. A bootstrap and permutation approach are available to assess configural and metric invariance. Hierarchical structures can be detected using Hierarchical EGA. Time series and intensive longitudinal data can be analyzed using Dynamic EGA, supporting individual, group, and population level assessments.

Author(s)

Hudson Golino <hfg9s@virginia.edu> and Alexander P. Christensen <alexpaulchristensen@gmail.com>

References

Christensen, A. P. (2023). Unidimensional community detection: A Monte Carlo simulation, grid search, and comparison. *PsyArXiv*.
Related functions: `community.unidimensional`

Christensen, A. P., Garrido, L. E., & Golino, H. (2023). Unique variable analysis: A network psychometrics method to detect local dependence. *Multivariate Behavioral Research*.
Related functions: `UVA`

Christensen, A. P., Garrido, L. E., Guerra-Pena, K., & Golino, H. (2023). Comparing community detection algorithms in psychometric networks: A Monte Carlo simulation. *Behavior Research Methods*.
Related functions: `EGA`

Christensen, A. P., & Golino, H. (2021a). Estimating the stability of the number of factors via Bootstrap Exploratory Graph Analysis: A tutorial. *Psych*, *3*(3), 479-500.
Related functions: `bootEGA`, `dimensionStability`, # and `itemStability`

Christensen, A. P., & Golino, H. (2021b). Factor or network model? Predictions from neural networks. *Journal of Behavioral Data Science*, *1*(1), 85-126.
Related functions: `LCT`

Christensen, A. P., & Golino, H. (2021c). On the equivalency of factor and network loadings. *Behavior Research Methods*, *53*, 1563-1580.
Related functions: `LCT` and `net.loads`

Christensen, A. P., Golino, H., & Silvia, P. J. (2020). A psychometric network perspective on the validity and validation of personality trait questionnaires. *European Journal of Personality*, *34*, 1095-1108.
Related functions: `bootEGA`, `dimensionStability`, # `EGA`, `itemStability`, and `UVA`

Christensen, A. P., Gross, G. M., Golino, H., Silvia, P. J., & Kwapil, T. R. (2019). Exploratory graph analysis of the Multidimensional Schizotypy Scale. *Schizophrenia Research*, *206*, 43-51. # Related functions: `CFA` and `EGA`

Golino, H., Christensen, A. P., Moulder, R., Kim, S., & Boker, S. M. (2021). Modeling latent topics in social media using Dynamic Exploratory Graph Analysis: The case of the right-wing and left-wing trolls in the 2016 US elections. *Psychometrika*.
Related functions: `dynEGA` and `simDFM`

Golino, H., & Demetriou, A. (2017). Estimating the dimensionality of intelligence like data using Exploratory Graph Analysis. *Intelligence*, *62*, 54-70.
Related functions: `EGA`

Golino, H., & Epskamp, S. (2017). Exploratory graph analysis: A new approach for estimating the number of dimensions in psychological research. *PLoS ONE*, *12*, e0174035.
Related functions: `CFA`, `EGA`, and `bootEGA`

Golino, H., Moulder, R., Shi, D., Christensen, A. P., Garrido, L. E., Nieto, M. D., Nesselroade, J., Sadana, R., Thiyagarajan, J. A., & Boker, S. M. (2020). Entropy fit indices: New fit measures for assessing the structure and dimensionality of multiple latent variables. *Multivariate Behavioral*

Research.
Related functions: `entropyFit`, `tefi`, and `vn.entropy`

Golino, H., Nesselroade, J. R., & Christensen, A. P. (2022). Towards a psychology of individuals: The ergodicity information index and a bottom-up approach for finding generalizations. *PsyArXiv*.
Related functions: `boot.ergoInfo`, `ergoInfo`, `jsd`, and `infoCluster`

Golino, H., Shi, D., Christensen, A. P., Garrido, L. E., Nieto, M. D., Sadana, R., Thiyagarajan, J. A., & Martinez-Molina, A. (2020). Investigating the performance of exploratory graph analysis and traditional techniques to identify the number of latent factors: A simulation and tutorial. *Psychological Methods*, *25*, 292-320.
Related functions: `EGA`

Golino, H., Thiyagarajan, J. A., Sadana, M., Teles, M., Christensen, A. P., & Boker, S. M. (2020). Investigating the broad domains of intrinsic capacity, functional ability, and environment: An exploratory graph analysis approach for improving analytical methodologies for measuring healthy aging. *PsyArXiv*.
Related functions: `EGA.fit` and `tefi`

Jamison, L., Christensen, A. P., & Golino, H. (2021). Optimizing Walktrap's community detection in networks using the Total Entropy Fit Index. *PsyArXiv*.
Related functions: `EGA.fit` and `tefi`

Jamison, L., Golino, H., & Christensen, A. P. (2023). Metric invariance in exploratory graph analysis via permutation testing. *PsyArXiv*.
Related functions: `invariance`

Shi, D., Christensen, A. P., Day, E., Golino, H., & Garrido, L. E. (2023). A Bayesian approach for dimensionality assessment in psychological networks. *PsyArXiv*
Related functions: `EGA`

See Also

Useful links:

- https://r-ega.net
- Report bugs at https://github.com/hfgolino/EGAnet/issues

`auto.correlate` *Automatic correlations*

Description

This wrapper is similar to `cor_auto`. There are some minor adjustments that make this function simpler and to function within `EGAnet`. `NA` values are not treated as categories (this behavior differs from `cor_auto`)

Usage

```
auto.correlate(
  data,
  corr = c("cosine", "kendall", "pearson", "spearman"),
  ordinal.categories = 7,
  forcePD = TRUE,
  na.data = c("pairwise", "listwise"),
  empty.method = c("none", "zero", "all"),
  empty.value = c("none", "point_five", "one_over"),
  verbose = FALSE,
  ...
)
```

Arguments

`data`	Matrix or data frame. Should consist only of variables to be used in the analysis
`corr`	Character (length = 1). The standard correlation method to be used. Defaults to `"pearson"`. Using `"pearson"` will compute polychoric, tetrachoric, polyserial, and biserial correlations for categorical and categorical/continuous correlations by default. To obtain `"pearson"` correlations regardless, use `cor`. Other options of `"kendall"` and `"spearman"` are provided for completeness and use `cor`. `cosine` is also available
`ordinal.categories`	Numeric (length = 1). *Up to* the number of categories *before* a variable is considered continuous. Defaults to `7` categories before `8` is considered continuous
`forcePD`	Boolean (length = 1). Whether positive definite matrix should be enforced. Defaults to `TRUE`
`na.data`	Character (length = 1). How should missing data be handled? Defaults to `"pairwise"`. Available options: • `"pairwise"` — Computes correlation for all available cases between two variables • `"listwise"` — Computes correlation for all complete cases in the dataset
`empty.method`	Character (length = 1). Method for empty cell correction in `polychoric.matrix`. Defaults to `"none"` Available options: • `"none"` — Adds no value (`empty.value = "none"`) to the empirical joint frequency table between two variables • `"zero"` — Adds `empty.value` to the cells with zero in the joint frequency table between two variables • `"all"` — Adds `empty.value` to all in the joint frequency table between two variables
`empty.value`	Character (length = 1). Value to add to the joint frequency table cells in `polychoric.matrix`. Defaults to `"none"`. Accepts numeric values between 0 and 1 or specific methods: • `"none"` — Adds no value (`0`) to the empirical joint frequency table between two variables

- "point_five" — Adds 0.5 to the cells defined by empty.method
- "one_over" — Adds 1 / n where n equals the number of cells based on empty.method. For empty.method = "zero", n equals the number of zero cells

verbose	Boolean (length = 1). Whether messages should be printed. Defaults to FALSE
...	Not actually used but makes it easier for general functionality in the package

Author(s)

Alexander P. Christensen <alexpaulchristensen@gmail.com>

Examples

```
# Load data
wmt <- wmt2[,7:24]

# Obtain correlations
wmt_corr <- auto.correlate(wmt)
```

---

boot.ergoInfo *Bootstrap Test for the Ergodicity Information Index*

Description

Tests the Ergodicity Information Index obtained in the empirical sample with a distribution of EII obtained by a variant of bootstrap sampling (see **Details** for the procedure)

Usage

```
boot.ergoInfo(
  dynEGA.object,
  EII,
  use = c("edge.list", "unweighted", "weighted"),
  shuffles = 5000,
  iter = 100,
  ncores,
  verbose = TRUE
)
```

Arguments

dynEGA.object	A dynEGA or a dynEGA.ind.pop object. If a dynEGA object, then level = c("individual", "population") is required
EII	A ergoInfo object used to estimate the Empirical Ergodicity Information Index or the estimated value of EII estimated using the ergoInfo function. Inherits use from ergoInfo. If no ergoInfo object is provided, then it is estimated

`use` Character (length = 1). A string indicating what network element will be used to compute the algorithm complexity, the list of edges or the weights of the network. Defaults to `use = "unweighted"`. Current options are:

- `"edge.list"` — Calculates the algorithm complexity using the list of edges
- `"unweighted"` — Calculates the algorithm complexity using the binary weights of the encoded prime transformed network. 0 = edge absent and 1 = edge present
- `"weighted"` — Calculates the algorithm complexity using the weights of encoded prime-weight transformed network

`shuffles` Numeric. Number of shuffles used to compute the Kolmogorov complexity. Defaults to `5000`

`iter` Numeric (length = 1). Number of replica samples to generate from the bootstrap analysis. Defaults to `100` (`1000` for robustness)

`ncores` Numeric (length = 1). Number of cores to use in computing results. Defaults to `ceiling(parallel::detectCores() / 2)` or half of your computer's processing power. Set to 1 to not use parallel computing

If you're unsure how many cores your computer has, then type: `parallel::detectCores()`

`verbose` Boolean (length = 1). Should progress be displayed? Defaults to `TRUE`. Set to `FALSE` to not display progress

Details

In traditional bootstrap sampling, individual participants are resampled with replacement from the empirical sample. This process is time consuming when carried out across *v* number of variables, *n* number of participants, *t* number of time points, and *i* number of iterations. Instead, `boot.ergoInfo` uses the premise of an ergodic process to establish more efficient test that works directly on the sample's networks.

With an ergodic process, the expectation is that all individuals will have a systematic relationship with the population. Destroying this relationship should result in a significant loss of information. Following this conjecture, `boot.ergoInfo` shuffles a random subset of edges that exist in the **population** that is *equal* to the number of shared edges it has with an individual. An individual's unique edges remain the same, controlling for their unique information. The result is a replicate individual that contains the same total number of edges as the actual individual but its shared information with the population has been scrambled.

This process is repeated over each individual to create a replicate sample and is repeated for *X* iterations (e.g., 100). This approach creates a sampling distribution that represents the expected information between the population and individuals when a random process generates the shared information between them. If the shared information between the population and individuals in the empirical sample is sufficiently meaningful, then this process should result in significant information loss.

How to interpret the results: the result of `boot.ergoInfo` is a sampling distribution of EII values that would be expected if the process was random (null distribution). If the empirical EII value is *greater than* or not significantly different from the null distribution, then the empirical data can be expected to be generated from an nonergodic process and the population structure is not sufficient to describe all individuals. If the empirical EII value is significantly *lower than* the null distribution, then the empirical data can be described by the population structure – the population structure is sufficient to describe all individuals.

Value

Returns a list containing:

`empirical.ergoInfo`	Empirical Ergodicity Information Index
`boot.ergoInfo`	The values of the Ergodicity Information Index obtained in the bootstrap
`p.value`	The two-sided *p*-value of the bootstrap test for the Ergodicity Information Index. The null hypothesis is that the empirical Ergodicity Information index is equal to or greater than the expected value of the EII with small variation in the population structure
`effect`	Indicates wheter the empirical EII is greater or less then the bootstrap distribution of EII.
`interpretation`	How you can interpret the result of the test in plain English

Author(s)

Hudson Golino <hfg9s at virginia.edu> & Alexander P. Christensen <alexander.christensen at Vanderbilt.Edu>

References

Original Implementation
Golino, H., Nesselroade, J. R., & Christensen, A. P. (2022). Toward a psychology of individuals: The ergodicity information index and a bottom-up approach for finding generalizations. *PsyArXiv*.

See Also

`plot.EGAnet` for plot usage in `EGAnet`

Examples

```
# Obtain simulated data
sim.data <- sim.dynEGA

## Not run:
# Dynamic EGA individual and population structures
dyn1 <- dynEGA.ind.pop(
  data = sim.dynEGA[,-26], n.embed = 5, tau = 1,
  delta = 1, id = 25, use.derivatives = 1,
  model = "glasso", ncores = 2, corr = "pearson"
)

# Empirical Ergodicity Information Index
eii1 <- ergoInfo(dynEGA.object = dyn1, use = "unweighted")

# Bootstrap Test for Ergodicity Information Index
testing.ergoinfo <- boot.ergoInfo(
  dynEGA.object = dyn1, EII = eii1,
  ncores = 2, use = "unweighted"
```

```
)

# Plot result
plot(testing.ergoinfo)

# Example using `dynEGA`
dyn2 <- dynEGA(
  data = sim.dynEGA, n.embed = 5, tau = 1,
  delta = 1, use.derivatives = 1, ncores = 2,
  level = c("individual", "population")
)

# Empirical Ergodicity Information Index
eii2 <- ergoInfo(dynEGA.object = dyn2, use = "unweighted")

# Bootstrap Test for Ergodicity Information Index
testing.ergoinfo2 <- boot.ergoInfo(
  dynEGA.object = dyn2, EII = eii2,
  ncores = 2
)

# Plot result
plot(testing.ergoinfo2)
## End(Not run)
```

---

boot.wmt bootEGA *Results of* wmt2*Data*

Description

bootEGA results from `boot.wmt <- bootEGA(wmt2[,7:24], seed = 1234)`

Usage

```
data(boot.wmt)
```

Format

A list with 12 objects (see **Value** in bootEGA)

Examples

```
data("boot.wmt")
```

bootEGA *Bootstrap Exploratory Graph Analysis*

Description

bootEGA Estimates the number of dimensions of iter bootstraps using the empirical zero-order correlation matrix ("parametric") or "resampling" from the empirical dataset (non-parametric). bootEGA estimates a typical median network structure, which is formed by the median or mean pairwise (partial) correlations over the *iter* bootstraps (see **Details** for information about the typical median network structure).

Usage

```
bootEGA(
  data,
  n = NULL,
  corr = c("auto", "cor_auto", "cosine", "pearson", "spearman"),
  na.data = c("pairwise", "listwise"),
  model = c("BGGM", "glasso", "TMFG"),
  algorithm = c("leiden", "louvain", "walktrap"),
  uni.method = c("expand", "LE", "louvain"),
  iter = 500,
  type = c("parametric", "resampling"),
  ncores,
  EGA.type = c("EGA", "EGA.fit", "hierEGA", "riEGA"),
  plot.itemStability = TRUE,
  typicalStructure = FALSE,
  plot.typicalStructure = FALSE,
  seed = NULL,
  verbose = TRUE,
  ...
)
```

Arguments

data	Matrix or data frame. Should consist only of variables to be used in the analysis
n	Numeric (length = 1). Sample size if data provided is a correlation matrix
corr	Character (length = 1). Method to compute correlations. Defaults to "auto". Available options:

- "auto" — Automatically computes appropriate correlations for the data using Pearson's for continuous, polychoric for ordinal, tetrachoric for binary, and polyserial/biserial for ordinal/binary with continuous. To change the number of categories that are considered ordinal, use ordinal.categories (see polychoric.matrix for more details)
- "cor_auto" — Uses cor_auto to compute correlations. Arguments can be passed along to the function

- "cosine" — Uses cosine to compute cosine similarity
- "pearson" — Pearson's correlation is computed for all variables regardless of categories
- "spearman" — Spearman's rank-order correlation is computed for all variables regardless of categories

For other similarity measures, compute them first and input them into data with the sample size (n)

na.data Character (length = 1). How should missing data be handled? Defaults to "pairwise". Available options:

- "pairwise" — Computes correlation for all available cases between two variables
- "listwise" — Computes correlation for all complete cases in the dataset

model Character (length = 1). Defaults to "glasso". Available options:

- "BGGM" — Computes the Bayesian Gaussian Graphical Model. Set argument ordinal.categories to determine levels allowed for a variable to be considered ordinal. See ?BGGM::estimate for more details
- "glasso" — Computes the GLASSO with EBIC model selection. See EBICglasso.qgraph for more details
- "TMFG" — Computes the TMFG method. See TMFG for more details

algorithm Character or igraph cluster_* function (length = 1). Defaults to "walktrap". Three options are listed below but all are available (see community.detection for other options):

- "leiden" — See cluster_leiden for more details
- "louvain" — By default, "louvain" will implement the Louvain algorithm using the consensus clustering method (see community.consensus for more information). This function will implement consensus.method = "most_common" and consensus.iter = 1000 unless specified otherwise
- "walktrap" — See cluster_walktrap for more details

uni.method Character (length = 1). What unidimensionality method should be used? Defaults to "louvain". Available options:

- "expand" — Expands the correlation matrix with four variables correlated 0.50. If number of dimension returns 2 or less in check, then the data are unidimensional; otherwise, regular EGA with no matrix expansion is used. This method was used in the Golino et al.'s (2020) *Psychological Methods* simulation
- "LE" — Applies the Leading Eigenvector algorithm (cluster_leading_eigen) on the empirical correlation matrix. If the number of dimensions is 1, then the Leading Eigenvector solution is used; otherwise, regular EGA is used. This method was used in the Christensen et al.'s (2023) *Behavior Research Methods* simulation
- "louvain" — Applies the Louvain algorithm (cluster_louvain) on the empirical correlation matrix. If the number of dimensions is 1, then the Louvain solution is used; otherwise, regular EGA is used. This method was validated Christensen's (2022) *PsyArXiv* simulation. Consensus clustering can be used by specifying either "consensus.method" or "consensus.iter"

`iter` Numeric (length = 1). Number of replica samples to generate from the bootstrap analysis. Defaults to `500` (recommended)

`type` Character (length = 1). What type of bootstrap should be performed? Defaults to `"parametric"`. Available options:

- `"parametric"` — Generates `iter` new datasets from (multivariate normal random distributions) based on the original dataset using `mvrnorm`
- `"resampling"` — Generates `iter` new datasets from random subsamples of the original data

`ncores` Numeric (length = 1). Number of cores to use in computing results. Defaults to `ceiling(parallel::detectCores() / 2)` or half of your computer's processing power. Set to `1` to not use parallel computing

If you're unsure how many cores your computer has, then type: `parallel::detectCores()`

`EGA.type` Character (length = 1). Type of EGA model to use. Defaults to `"EGA"` Available options:

- `"EGA"` — Uses standard exploratory graph analysis
- `"EGA.fit"` — Uses `tefi` to determine best fit of `EGA`
- `"hierEGA"` — Uses hierarchical exploratory graph analysis
- `"riEGA"` — Uses random-intercept exploratory graph analysis

Arguments for `EGA.type` can be added (see links for details on specific function arguments)

`plot.itemStability` Boolean (length = 1). Should the plot be produced for `item.replication`? Defaults to `TRUE`

`typicalStructure` Boolean (length = 1). If `TRUE`, returns the median (`"glasso"` or `"BGGM"`) or mean (`"TMFG"`) network structure and estimates its dimensions (see **Details** for more information). Defaults to `FALSE`

`plot.typicalStructure` Boolean (length = 1). If `TRUE`, returns a plot of the typical network structure. Defaults to `FALSE`

`seed` Numeric (length = 1). Defaults to `NULL` or random results. Set for reproducible results. See Reproducibility and PRNG for more details on random number generation in `EGAnet`

`verbose` Boolean (length = 1). Should progress be displayed? Defaults to `TRUE`. Set to `FALSE` to not display progress

`...` Additional arguments that can be passed on to `auto.correlate`, `network.estimation`, `community.detection`, `community.consensus`, `EGA`, `EGA.fit`, `hierEGA`, and `riEGA`

Details

The typical network structure is derived from the median (or mean) value of each pairwise relationship. These values tend to reflect the "typical" value taken by an edge across the bootstrap networks. Afterward, the same community detection algorithm is applied to the typical network as the bootstrap networks.

Because the community detection algorithm is applied to the typical network structure, there is a possibility that the community algorithm determines a different number of dimensions than the median number derived from the bootstraps. The typical network structure (and number of dimensions) may *not* match the empirical EGA number of dimensions or the median number of dimensions from the bootstrap. This result is known and *not* a bug.

Value

Returns a list containing:

`iter`	Number of replica samples in bootstrap
`bootGraphs`	A list containing the networks of each replica sample
`boot.wc`	A matrix of membership assignments for each replica network with variables down the columns and replicas across the rows
`boot.ndim`	Number of dimensions identified in each replica sample
`summary.table`	A data frame containing number of replica samples, median, standard deviation, standard error, 95% confidence intervals, and quantiles (lower = 2.5% and upper = 97.5%)
`frequency`	A data frame containing the proportion of times the number of dimensions was identified (e.g., .85 of 1,000 = 850 times that specific number of dimensions was found)
`TEFI`	`tefi` value for each replica sample
`type`	Type of bootstrap used
`EGA`	Output of the empirical EGA results (output will vary based on `EGA.type`)
`EGA.type`	Type of `*EGA` function used
`typicalGraph`	A list containing: • `graph` — Network matrix of the median network structure • `typical.dim.variables` — An ordered matrix of item allocation • `wc` — Membership assignments of the median network
`plot.typical.ega`	Plot output if `plot.typicalStructure = TRUE`

Author(s)

Hudson Golino <hfg9s at virginia.edu> and Alexander P. Christensen <alexpaulchristensen@gmail.com>

References

Original implementation of bootEGA

Christensen, A. P., & Golino, H. (2021). Estimating the stability of the number of factors via Bootstrap Exploratory Graph Analysis: A tutorial. *Psych*, *3*(3), 479-500.

See Also

`itemStability` to estimate the stability of the variables in the empirical dimensions and `dimensionStability` to estimate the stability of the dimensions (structural consistency)

Examples

```
# Load data
wmt <- wmt2[,7:24]

## Not run:
# Standard EGA parametric example
boot.wmt <- bootEGA(
  data = wmt, iter = 500,
  type = "parametric", ncores = 2
)

# Standard resampling example
boot.wmt <- bootEGA(
  data = wmt, iter = 500,
  type = "resampling", ncores = 2
)

# Example using {igraph} `cluster_*` function
boot.wmt.spinglass <- bootEGA(
  data = wmt, iter = 500,
  algorithm = igraph::cluster_spinglass,
  # use any function from {igraph}
  type = "parametric", ncores = 2
)

# EGA fit example
boot.wmt.fit <- bootEGA(
  data = wmt, iter = 500,
  EGA.type = "EGA.fit",
  type = "parametric", ncores = 2
)

# Hierarchical EGA example
boot.wmt.hier <- bootEGA(
  data = wmt, iter = 500,
  EGA.type = "hierEGA",
  type = "parametric", ncores = 2
)

# Random-intercept EGA example
boot.wmt.ri <- bootEGA(
  data = wmt, iter = 500,
  EGA.type = "riEGA",
  type = "parametric", ncores = 2
)
## End(Not run)
```

Description

Verifies the fit of the structure suggested by EGA or by hierEGA using confirmatory factor analysis

Usage

```
CFA(ega.obj, data, estimator, plot.CFA = TRUE, layout = "spring", ...)
```

Arguments

ega.obj	An EGA object or an hierEGA
data	Matrix or data frame. Should consist only of variables to be used in the analysis
estimator	The estimator used in the confirmatory factor analysis. 'WLSMV' is the estimator of choice for ordinal variables. 'ML' or 'WLS' for interval variables. See lavOptions for more details
plot.CFA	Logical. Should the CFA structure with its standardized loadings be plot? Defaults to TRUE
layout	Layout of plot (see semPaths). Defaults to "spring"
...	Arguments passed to cfa

Value

Returns a list containing:

fit	Output from cfa
summary	Summary output from lavaan-class
fit.measures	Fit measures: chi-squared, degrees of freedom, p-value, CFI, RMSEA, GFI, and NFI. Additional fit measures can be applied using the fitMeasures function (see examples)

Author(s)

Hudson F. Golino <hfg9s at virginia.edu>

References

Demonstrative use
Christensen, A. P., Gross, G. M., Golino, H., Silvia, P. J., & Kwapil, T. R. (2019). Exploratory graph analysis of the Multidimensional Schizotypy Scale. *Schizophrenia Research*, *206*, 43-51.

Initial implementation
Golino, H., & Epskamp, S. (2017). Exploratory graph analysis: A new approach for estimating the number of dimensions in psychological research. *PLoS ONE*, *12*, e0174035.

Examples

```
# Load data
wmt <- wmt2[,7:24]

## Not run:
# Estimate EGA
ega.wmt <- EGA(
  data = wmt,
  plot.EGA = FALSE # No plot for CRAN checks
)

# Fit CFA model to EGA results
cfa.wmt <- CFA(
  ega.obj = ega.wmt, estimator = "WLSMV",
  plot.CFA = FALSE, # No plot for CRAN checks
  data = wmt
)

# Additional fit measures
lavaan::fitMeasures(cfa.wmt$fit, fit.measures = "all")
## End(Not run)
```

`color_palette_EGA` *EGA Color Palettes*

Description

Color palettes for plotting `ggnet2` `EGA` network plots

Usage

```
color_palette_EGA(
 name = c("polychrome", "blue.ridge1", "blue.ridge2", "rainbow", "rio", "itacare",
    "grayscale"),
  wc,
  sorted = FALSE
)
```

Arguments

`name`	Character. Name of color scheme (see `RColorBrewer`). Defaults to `"polychrome"`. `EGA` palettes: • `"polychrome"` — Default 40 color palette • `"grayscale"` — "grayscale", "greyscale", or "colorblind" will produce plots suitable for publication purposes • `"blue.ridge1"` — Palette inspired by the Blue Ridge Mountains • `"blue.ridge2"` — Second palette inspired by the Blue Ridge Mountains

- "rainbow" — Rainbow colors. Default for qgraph
- "rio" — Palette inspired by Rio de Janiero, Brazil
- "itacare" — Palette inspired by Itacare, Brazil

For custom colors, enter HEX codes for each dimension in a vector

wc — Numeric vector. A vector representing the community (dimension) membership of each node in the network. NA values mean that the node was disconnected from the network

sorted — Boolean. Should colors be sorted by wc? Defaults to FALSE

Value

Vector of colors for community memberships

Author(s)

Hudson Golino <hfg9s at virginia.edu>, Alexander P. Christensen <alexpaulchristensen at gmail.com>

See Also

plot.EGAnet for plot usage in EGAnet

Examples

```
# Default
color_palette_EGA(name = "polychrome", wc = ega.wmt$wc)

# Blue Ridge Moutains 1
color_palette_EGA(name = "blue.ridge1", wc = ega.wmt$wc)

# Custom
color_palette_EGA(name = c("#7FD1B9", "#24547e"), wc = ega.wmt$wc)
```

---

community.compare *Compares Community Detection Solutions Using Permutation*

Description

A permutation implementation to determine statistical significance of whether the community comparison measure is different from zero

Usage

```
community.compare(
  base,
  comparison,
  method = c("vi", "nmi", "split.join", "rand", "adjusted.rand"),
  iter = 1000,
  shuffle.base = TRUE,
  verbose = TRUE,
  seed = NULL
)
```

Arguments

`base`	Character or numeric vector. A vector of characters or numbers that are treated as the baseline communities
`comparison`	Character or numeric vector (length = `length(base)`). A vector of characters or numbers that are treated as the baseline communities
`method`	Character (length = 1). Comparison metrics from `compare`. Defaults to `"adjusted.rand"`. Available options: • `"vi"` — Variation of information (Meila, 2003) • `"nmi"` — Normalized mutual information (Danon et al., 2003) • `"split.join"` — Split-join distance (Dongen, 2000) • `"rand"` — Rand index (Rand, 1971) • `"adjusted.rand"` — adjusted Rand index (Hubert & Arabie, 1985; Steinley, 2004)
`iter`	Numeric (length = 1). Number of permutations to perform. Defaults to `1000` (recommended)
`shuffle.base`	Boolean (length = 1). Whether the `base` cluster solution should be shuffled. Defaults to `TRUE` to remain consistent with original implementation (Qannari et al., 2014); however, from a theoretical standpoint, it might make sense to only shuffle the `comparison` to determine whether it is specifically different from the recognized `base`
`verbose`	Boolean (length = 1). Should progress be displayed? Defaults to `TRUE`. Set to `FALSE` to not display progress
`seed`	Numeric (length = 1). Defaults to `NULL` or random results. Set for reproducible results. See Reproducibility and PRNG for more details on random number generation in `EGAnet`

Value

Returns data frame containing method used (`Method`), empirical or observed value (`Empirical`), and p-value based on the permutation test (`p.value`)

Author(s)

Hudson Golino <hfg9s at virginia.edu> and Alexander P. Christensen <alexpaulchristensen@gmail.com>

References

Implementation of Permutation Test
Qannari, E. M., Courcoux, P., & Faye, P. (2014). Significance test of the adjusted Rand index. Application to the free sorting task. *Food Quality and Preference*, *32*, 93–97.

Variation of Information
Meila, M. (2003, August). Comparing clusterings by the variation of information. In *Learning Theory and Kernel Machines: 16th Annual Conference on Learning Theory and 7th Kernel Workshop*, COLT/Kernel 2003, Washington, DC, USA, August 24-27, 2003. Proceedings (pp. 173-187). Berlin, DE: Springer Berlin Heidelberg.

Normalized Mutual Information
Danon, L., Diaz-Guilera, A., Duch, J., & Arenas, A. (2005). Comparing community structure identification. *Journal of Statistical Mechanics: Theory and Experiment*, *2005*(09), P09008.

Split-join Distance
Dongen, S. (2000). Performance criteria for graph clustering and Markov cluster experiments. *CWI (Centre for Mathematics and Computer Science)*.

Rand Index
Rand, W. M. (1971). Objective criteria for the evaluation of clustering methods. *Journal of the American Statistical Association*, *66*(336), 846-850.

Adjusted Rand Index
Hubert, L., & Arabie, P. (1985). Comparing partitions. *Journal of Classification*, *2*, 193-218.

Steinley, D. (2004). Properties of the Hubert-Arabie adjusted rand index. *Psychological Methods*, *9*(3), 386.

Examples

```
# Load data
wmt <- wmt2[,7:24]

# Estimate network
network <- EBICglasso.qgraph(data = wmt)

# Compute Edge Betweenness
edge_between <- community.detection(network, algorithm = "edge_betweenness")

# Compute Fast Greedy
fast_greedy <- community.detection(network, algorithm = "fast_greedy")

# Perform permutation test
community.compare(edge_between, fast_greedy)
```

`community.consensus` *Applies the Consensus Clustering Method (Louvain only)*

Description

Applies the consensus clustering method introduced by (Lancichinetti & Fortunato, 2012). The original implementation of this method applies a community detection algorithm repeatedly to the same network. With stochastic networks, the algorithm is likely to identify different community solutions with many repeated applications.

Usage

```
community.consensus(
  network,
  order = c("lower", "higher"),
  resolution = 1,
 consensus.method = c("highest_modularity", "iterative", "most_common", "lowest_tefi"),
  consensus.iter = 1000,
  correlation.matrix = NULL,
  allow.singleton = FALSE,
  membership.only = TRUE,
  ...
)
```

Arguments

network	Matrix or igraph network object
order	Character (length = 1). Defaults to "higher". Whether "lower" or "higher" order memberships from the Louvain algorithm should be obtained for the consensus. The "lower" order Louvain memberships are from the first initial pass of the Louvain algorithm whereas the "higher" order Louvain memberships are from the last pass of the Louvain algorithm
resolution	Numeric (length = 1). A parameter that adjusts modularity to allow the algorithm to prefer smaller (resolution > 1) or larger (0 < resolution < 1) communities. Defaults to 1 (standard modularity computation)
consensus.method	Character (length = 1). Defaults to "most_common". Available options for arriving at a consensus (*Note*: All methods except "iterative" are considered experimental until validated): • "highest_modularity" — **EXPERIMENTAL.** Selects the community solution with the highest modularity across the applications. Modularity is a reasonable metric for identifying the number of communities in a network but it comes with limitations (e.g., resolution limit) • "iterative" — The original approach proposed by Lancichinetti & Fortunato (2012). See "Details" for more information • "most_common" — Selects the community solution that appears the most frequently across the applications. The idea behind this method is that the solution that appears most often will be the most likely solution for the algorithm as well as most reproducible. Can be less stable as the number of nodes increase requiring a larger value for consensus.iter. This method is the **default**

- `"lowest_tefi"` — **EXPERIMENTAL.** Selects the community solution with the lowest Total Entropy Fit Index (`tefi`) across the applications. TEFI is a reasonable metric to identify the number of communities in a network based on Golino, Moulder et al. (2020)

`consensus.iter` Numeric (length = 1). Number of algorithm applications to the network. Defaults to `1000`

`correlation.matrix`
Symmetric matrix. Used for computation of `tefi`. Only needed when `consensus.method = "tefi"`

`allow.singleton`
Boolean (length = 1). Whether singleton or single node communities should be allowed. Defaults to `FALSE`. When `FALSE`, singleton communities will be set to missing (`NA`); otherwise, when `TRUE`, singleton communities will be allowed

`membership.only`
Boolean. Whether the memberships only should be output. Defaults to `TRUE`. Set to `FALSE` to obtain all output for the community detection algorithm

`...` Not actually used but makes it easier for general functionality in the package

Details

The goal of the consensus clustering method is to identify a stable solution across algorithm applications to derive a "consensus" clustering. The standard or "iterative" approach is to apply the community detection algorithm *N* times. Then, a co-occurrence matrix is created representing how often each pair of nodes co-occurred across the applications. Based on some cut-off value (e.g., 0.30), co-occurrences below this value are set to zero, forming a "new" sparse network. The procedure proceeds until all nodes co-occur with all other nodes in their community (or a proportion of 1.00).

Variations of this procedure are also available in this package but are **experimental**. Use these experimental procedures with caution. More work is necessary before these experimental procedures are validated

At this time, seed setting for consensus clustering is not supported

Value

Returns either a vector with the selected solution or a list when `membership.only = FALSE`:

`selected_solution`
Resulting solution from the consensus method

`memberships` Matrix of memberships across the consensus iterations

`proportion_table`
For methods that use frequency, a table that reports those frequencies alongside their corresponding memberships

Author(s)

Hudson Golino <hfg9s at virginia.edu> and Alexander P. Christensen <alexpaulchristensen@gmail.com>

References

Louvain algorithm
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*(10), P10008.

Consensus clustering
Lancichinetti, A., & Fortunato, S. (2012). Consensus clustering in complex networks. *Scientific Reports*, *2*(1), 1–7.

Entropy fit indices
Golino, H., Moulder, R. G., Shi, D., Christensen, A. P., Garrido, L. E., Nieto, M. D., Nesselroade, J., Sadana, R., Thiyagarajan, J. A., & Boker, S. M. (2020). Entropy fit indices: New fit measures for assessing the structure and dimensionality of multiple latent variables. *Multivariate Behavioral Research*.

Examples

```
# Load data
wmt <- wmt2[,7:24]

# Estimate correlation matrix
correlation.matrix <- auto.correlate(wmt)

# Estimate network
network <- EBICglasso.qgraph(data = wmt)

# Compute standard Louvain with highest modularity approach
community.consensus(
  network,
  consensus.method = "highest_modularity"
)

# Compute standard Louvain with iterative (original) approach
community.consensus(
  network,
  consensus.method = "iterative"
)

# Compute standard Louvain with most common approach
community.consensus(
  network,
  consensus.method = "most_common"
)

# Compute standard Louvain with lowest TEFI approach
community.consensus(
  network,
  consensus.method = "lowest_tefi",
  correlation.matrix = correlation.matrix
)
```

`community.detection` *Apply a Community Detection Algorithm*

Description

General function to apply community detection algorithms available in `igraph`. Follows the `EGAnet` approach of setting singleton and disconnected nodes to missing (`NA`)

Usage

```
community.detection(
  network,
 algorithm = c("edge_betweenness", "fast_greedy", "fluid", "infomap", "label_prop",
    "leading_eigen", "leiden", "louvain", "optimal", "spinglass", "walktrap"),
  allow.singleton = FALSE,
  membership.only = TRUE,
  ...
)
```

Arguments

`network` Matrix or `igraph` network object

`algorithm` Character or `igraph` `cluster_*` function (length = 1). Available options:

- `"edge_betweenness"` — See `cluster_edge_betweenness` for more details
- `"fast_greedy"` — See `cluster_fast_greedy` for more details
- `"fluid"` — See `cluster_fluid_communities` for more details
- `"infomap"` — See `cluster_infomap` for more details
- `"label_prop"` — See `cluster_label_prop` for more details
- `"leading_eigen"` — See `cluster_leading_eigen` for more details
- `"leiden"` — See `cluster_leiden` for more details. *Note*: The Leiden algorithm will default to the modularity objective function (`objective_function = "modularity"`). Set `objective_function = "CPM"` to use the Constant Potts Model instead (see examples)
- `"louvain"` — See `cluster_louvain` for more details
- `"optimal"` — See `cluster_optimal` for more details
- `"spinglass"` — See `cluster_spinglass` for more details
- `"walktrap"` — See `cluster_walktrap` for more details

`allow.singleton` Boolean (length = 1). Whether singleton or single node communities should be allowed. Defaults to `FALSE`. When `FALSE`, singleton communities will be set to missing (`NA`); otherwise, when `TRUE`, singleton communities will be allowed

`membership.only` Boolean (length = 1). Whether the memberships only should be output. Defaults to `TRUE`. Set to `FALSE` to obtain all output for the community detection algorithm

... Additional arguments to be passed on to igraph's community detection functions (see algorithm for link to arguments of each algorithm)

Value

Returns memberships from a community detection algorithm

Author(s)

Hudson Golino <hfg9s at virginia.edu> and Alexander P. Christensen <alexpaulchristensen@gmail.com>

References

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

Examples

```
# Load data
wmt <- wmt2[,7:24]

# Estimate network
network <- EBICglasso.qgraph(data = wmt)

# Compute Edge Betweenness
community.detection(network, algorithm = "edge_betweenness")

# Compute Fast Greedy
community.detection(network, algorithm = "fast_greedy")

# Compute Fluid
community.detection(
  network, algorithm = "fluid",
  no.of.communities = 2 # needs to be set
)

# Compute Infomap
community.detection(network, algorithm = "infomap")

# Compute Label Propagation
community.detection(network, algorithm = "label_prop")

# Compute Leading Eigenvector
community.detection(network, algorithm = "leading_eigen")

# Compute Leiden (with modularity)
community.detection(
  network, algorithm = "leiden",
  objective_function = "modularity"
)

# Compute Leiden (with CPM)
```

```
community.detection(
  network, algorithm = "leiden",
  objective_function = "CPM",
  resolution_parameter = 0.05 # "edge density"
)

# Compute Louvain
community.detection(network, algorithm = "louvain")

# Compute Optimal (identifies maximum modularity solution)
community.detection(network, algorithm = "optimal")

# Compute Spinglass
community.detection(network, algorithm = "spinglass")

# Compute Walktrap
community.detection(network, algorithm = "walktrap")

# Example with {igraph} network
community.detection(
  convert2igraph(network), algorithm = "walktrap"
)
```

`community.homogenize` *Homogenize Community Memberships*

Description

Memberships from community detection algorithms do not always align numerically. This function seeks to homogenize community memberships between a target membership (the membership to homogenize toward) and one or more other memberships. This function is the core of the `dimensionStability` and `itemStability` functions

Usage

```
community.homogenize(target.membership, convert.membership)
```

Arguments

`target.membership`
: Vector, matrix, or data frame. The target memberships that all other memberships input into `convert.membership` should be homogenize **toward**

`convert.membership`
: Vector, matrix, or data frame. Either a vector of memberships the same length as `target.membership` or a matrix or data frame of many membership solutions with either across rows or down columns the same length as `target.membership` (this function will automatically determine this orientation for you with precedence given solutions *across rows*)

Value

Returns a vector or matrix the length or size of `convert.membership` with memberships homogenized toward `target.membership`

Author(s)

Hudson Golino <hfg9s at virginia.edu> and Alexander P. Christensen <alexpaulchristensen@gmail.com>

References

Original implementation of bootEGA
Christensen, A. P., & Golino, H. (2021). Estimating the stability of the number of factors via Bootstrap Exploratory Graph Analysis: A tutorial. *Psych*, *3*(3), 479-500.

Examples

```
# Get network
network <- network.estimation(wmt2[,7:24])

# Apply Walktrap
network_walktrap <- community.detection(
  network, algorithm = "walktrap"
)

# Apply Louvain
network_louvain <- community.detection(
  network, algorithm = "louvain"
)

# Homogenize toward Walktrap
community.homogenize(network_walktrap, network_louvain)
```

---

community.unidimensional

Approaches to Detect Unidimensional Communities

Description

A function to apply several approaches to detect a unidimensional community in networks. There have many different approaches recently such as expanding the correlation matrix to have orthogonal correlations (`"expand"`), applying the Leading Eigenvalue community detection algorithm `cluster_leading_eigen` to the correlation matrix (`"LE"`), and applying the Louvain community detection algorithm `cluster_louvain` to the correlation matrix (`"louvain"`). Not necessarily intended for individual use – it's better to use `EGA`

Usage

```
community.unidimensional(
  data,
  n = NULL,
  corr = c("auto", "cor_auto", "pearson", "spearman"),
  na.data = c("pairwise", "listwise"),
  model = c("BGGM", "glasso", "TMFG"),
  uni.method = c("expand", "LE", "louvain"),
  verbose = FALSE,
  ...
)
```

Arguments

`data`	Matrix or data frame. Should consist only of variables that are desired to be in analysis
`n`	Numeric (length = 1). Sample size if `data` provided is a correlation matrix
`corr`	Character (length = 1). Method to compute correlations. Defaults to `"auto"`. Available options: • `"auto"` — Automatically computes appropriate correlations for the data using Pearson's for continuous, polychoric for ordinal, tetrachoric for binary, and polyserial/biserial for ordinal/binary with continuous. To change the number of categories that are considered ordinal, use `ordinal.categories` (see `polychoric.matrix` for more details) • `"cor_auto"` — Uses `cor_auto` to compute correlations. Arguments can be passed along to the function • `"pearson"` — Pearson's correlation is computed for all variables regardless of categories • `"spearman"` — Spearman's rank-order correlation is computed for all variables regardless of categories For other similarity measures, compute them first and input them into `data` with the sample size (`n`)
`na.data`	Character (length = 1). How should missing data be handled? Defaults to `"pairwise"`. Available options: • `"pairwise"` — Computes correlation for all available cases between two variables • `"listwise"` — Computes correlation for all complete cases in the dataset
`model`	Character (length = 1). Defaults to `"glasso"`. Available options: • `"BGGM"` — Computes the Bayesian Gaussian Graphical Model. Set argument `ordinal.categories` to determine levels allowed for a variable to be considered ordinal. See `?BGGM::estimate` for more details • `"glasso"` — Computes the GLASSO with EBIC model selection. See `EBICglasso.qgraph` for more details • `"TMFG"` — Computes the TMFG method. See `TMFG` for more details

`uni.method` Character (length = 1). What unidimensionality method should be used? Defaults to `"louvain"`. Available options:

- `"expand"` — Expands the correlation matrix with four variables correlated 0.50. If number of dimension returns 2 or less in check, then the data are unidimensional; otherwise, regular EGA with no matrix expansion is used. This method was used in the Golino et al.'s (2020) *Psychological Methods* simulation
- `"LE"` — Applies the Leading Eigenvector algorithm (`cluster_leading_eigen`) on the empirical correlation matrix. If the number of dimensions is 1, then the Leading Eigenvector solution is used; otherwise, regular EGA is used. This method was used in the Christensen et al.'s (2023) *Behavior Research Methods* simulation
- `"louvain"` — Applies the Louvain algorithm (`cluster_louvain`) on the empirical correlation matrix. If the number of dimensions is 1, then the Louvain solution is used; otherwise, regular EGA is used. This method was validated Christensen's (2022) *PsyArXiv* simulation. Consensus clustering can be used by specifying either `"consensus.method"` or `"consensus.iter"`

`verbose` Boolean. Whether messages and (insignificant) warnings should be output. Defaults to `FALSE` (silent calls). Set to `TRUE` to see all messages and warnings for every function call

`...` Additional arguments to be passed on to `auto.correlate`, `network.estimation`, `community.consensus`, and `community.detection`

Value

Returns the memberships of the community detection algorithm. The memberships will output *regardless* of whether the network is unidimensional

Author(s)

Hudson Golino <hfg9s at virginia.edu> and Alexander P. Christensen <alexpaulchristensen@gmail.com>

References

Expand approach
Golino, H., Shi, D., Christensen, A. P., Garrido, L. E., Nieto, M. D., Sadana, R., Thiyagarajan, J. A., & Martinez-Molina, A. (2020). Investigating the performance of exploratory graph analysis and traditional techniques to identify the number of latent factors: A simulation and tutorial. *Psychological Methods*, *25*, 292-320.

Leading Eigenvector approach
Christensen, A. P., Garrido, L. E., Guerra-Pena, K., & Golino, H. (2023). Comparing community detection algorithms in psychometric networks: A Monte Carlo simulation. *Behavior Research Methods*.

Louvain approach
Christensen, A. P. (2023). Unidimensional community detection: A Monte Carlo simulation, grid search, and comparison. *PsyArXiv*.

Examples

```
# Load data
wmt <- wmt2[,7:24]

# Louvain with Consensus Clustering (default)
community.unidimensional(wmt)

# Leading Eigenvector
community.unidimensional(wmt, uni.method = "LE")

# Expand
community.unidimensional(wmt, uni.method = "expand")
```

---

compare.EGA.plots *Visually Compare Two or More* EGAnet *plots*

Description

Organizes EGA plots for comparison. Ensures that nodes are placed in the same layout to maximize comparison

Usage

```
compare.EGA.plots(
  ...,
  input.list = NULL,
  base = 1,
  labels = NULL,
  rows = NULL,
  columns = NULL,
  plot.all = TRUE
)
```

Arguments

`...`	Handles multiple arguments: • *EGA objects — can be dropped in without any argument designation. The function will search across input to find necessary EGAnet objects • ggnet2 arguments — can be passed along to ggnet2 • gplot.layout — can be specified using mode = or layout = using the name of the layout (e.g., mode = "circle" will produce the circle layout from gplot.layout). By default, the layout is the same as qgraph
input.list	List. Bypasses ... argument in favor of using a list as an input
base	Numeric (length = 1). Plot to be used as the base for the configuration of the networks. Uses the number of the order in which the plots are input. Defaults to 1 or the first plot

labels	Character (same length as input). Labels for each EGAnet object
rows	Numeric (length = 1). Number of rows to spread plots across
columns	Numeric (length = 1). Number of columns to spread plots down
plot.all	Boolean (length = 1). Whether plot should be produced or just output. Defaults to TRUE. Set to FALSE to avoid plotting (but still obtain plot objects)

Value

Visual comparison of EGAnet objects

Author(s)

Alexander Christensen <alexpaulchristensen@gmail.com>

See Also

plot.EGAnet for plot usage in EGAnet

Examples

```
# Obtain WMT-2 data
wmt <- wmt2[,7:24]

# Draw random samples of 300 cases
sample1 <- wmt[sample(1:nrow(wmt), 300),]
sample2 <- wmt[sample(1:nrow(wmt), 300),]

# Estimate EGAs
ega1 <- EGA(sample1)
ega2 <- EGA(sample2)


# Compare EGAs via plot
compare.EGA.plots(
  ega1, ega2,
  base = 1, # use "ega1" as base for comparison
  labels = c("Sample 1", "Sample 2"),
  rows = 1, columns = 2
)

# Change layout to circle plots
compare.EGA.plots(
  ega1, ega2,
  labels = c("Sample 1", "Sample 2"),
  mode = "circle"
)
```

convert2igraph *Convert networks to* igraph

Description

Converts networks to igraph format

Usage

```
convert2igraph(A, diagonal = 0)
```

Arguments

A	Matrix or data frame. *N* x *N* matrix where *N* is the number of nodes
diagonal	Numeric. Value to be placed on the diagonal of A. Defaults to 0

Value

Returns a network in the igraph format

Author(s)

Hudson Golino <hfg9s at virginia.edu> & Alexander P. Christensen <alexander.christensen at Vanderbilt.Edu>

Examples

```
convert2igraph(ega.wmt$network)
```

convert2tidygraph *Convert networks to* tidygraph

Description

Converts networks to tidygraph format

Usage

```
convert2tidygraph(EGA.object)
```

Arguments

EGA.object	A single EGAnet object containing the outputs $network and $wc

Value

Returns a network in the `tidygraph` format

Author(s)

Dominique Makowski, Hudson Golino <hfg9s at virginia.edu>, & Alexander P. Christensen <alexander.christensen at Vanderbilt.Edu>

Examples

```
convert2tidygraph(ega.wmt)
```

---

cosine *Cosine similarity*

Description

Computes cosine similarity

Usage

```
cosine(x, y = NULL, ...)
```

Arguments

`x`	Numeric vector, matrix, or data frame. If `nrow(x) > 1`, then `x` will be treated as a matrix to compute an *n* by *n* similarity matrix (`y` will not be used!)
`y`	Numeric vector, matrix, or data frame. Only used if `x` is a single variable. Used to compute similarity between one variable and *n* other variables. Defaults to `NULL`
`...`	Not actually used but makes it easier for general functionality in the package

Details

On missing values: `0` will be used to replace missing values. When using (matrix) multiplication, the `0` value cancels out the product rendering the missing value as "not counting" in the sums

Author(s)

Alexander P. Christensen <alexpaulchristensen@gmail.com>

Examples

```
# Load data
wmt <- wmt2[,7:24]

# Obtain cosines
wmt_cosine <- cosine(wmt)
```

`depression` *Depression Data*

Description

A response matrix (n = 574) of the Beck Depression Inventory, Beck Anxiety Inventory, and the Athens Insomnia Scale.

Usage

```
data(depression)
```

Format

A 574x78 response matrix

Examples

```
data("depression")
```

`dimensionStability` *Dimension Stability Statistics from* `bootEGA`

Description

Based on the `bootEGA` results, this function computes the stability of dimensions. Stability is computed by assessing the proportion of times the original dimension is exactly replicated in across bootstrap samples

Usage

```
dimensionStability(bootega.obj, IS.plot = TRUE, structure = NULL, ...)
```

Arguments

`bootega.obj`	A `bootEGA` object
`IS.plot`	Boolean (length = 1). Should the plot be produced for `item.replication`? Defaults to `TRUE`
`structure`	Numeric (length = number of variables). A theoretical or pre-defined structure. Defaults to `NULL` or the empirical `EGA` result in the `bootega.obj`
`...`	Additional arguments. Used for deprecated arguments from previous versions of `itemStability`

Value

Returns a list containing:

`dimension.stability`
: A list containing:

- `structural.consistency` — The proportion of times that each empirical `EGA` dimension *exactly* replicates across the `bootEGA` samples
- `average.item.stability` — The average item stability in each empirical `EGA` dimension

`item.stability`
: Results from `itemStability`

Author(s)

Hudson Golino <hfg9s at virginia.edu> and Alexander P. Christensen <alexpaulchristensen@gmail.com>

References

Original implementation of bootEGA
Christensen, A. P., & Golino, H. (2021). Estimating the stability of the number of factors via Bootstrap Exploratory Graph Analysis: A tutorial. *Psych*, *3*(3), 479-500.

Conceptual introduction
Christensen, A. P., Golino, H., & Silvia, P. J. (2020). A psychometric network perspective on the validity and validation of personality trait questionnaires. *European Journal of Personality*, *34*(6), 1095-1108.

Examples

```
# Load data
wmt <- wmt2[,7:24]

## Not run:
# Estimate bootstrap EGA
boot.wmt <- bootEGA(
  data = wmt, iter = 500,
  type = "parametric", ncores = 2
)
## End(Not run)
```

```
# Estimate stability statistics
dimensionStability(boot.wmt)
```

---

dnn.weights *Loadings Comparison Test Deep Learning Neural Network Weights*

Description

A list of weights from four different neural network models: random vs. non-random model (r_nr_weights), low correlation factor vs. network model (lf_n_weights), high correlation with variables less than or equal to factors vs. network model (hlf_n_weights), and high correlation with variables greater than factors vs. network model (hgf_n_weights)

Usage

```
data(dnn.weights)
```

Format

A list of with a length of 4

Examples

```
data("dnn.weights")
```

---

dynEGA *Dynamic Exploratory Graph Analysis*

Description

Estimates dynamic communities in multivariate time series (e.g., panel data, longitudinal data, intensive longitudinal data) at multiple time scales and at different levels of analysis: individuals (intraindividual structure), groups, and population (interindividual structure)

Usage

```
dynEGA(
  data,
  id = NULL,
  group = NULL,
  n.embed = 5,
  tau = 1,
  delta = 1,
  use.derivatives = 1,
```

```
  level = c("individual", "group", "population"),
  corr = c("auto", "cor_auto", "pearson", "spearman"),
  na.data = c("pairwise", "listwise"),
  model = c("BGGM", "glasso", "TMFG"),
  algorithm = c("leiden", "louvain", "walktrap"),
  uni.method = c("expand", "LE", "louvain"),
  ncores,
  verbose = TRUE,
  ...
)
```

Arguments

`data`	Matrix or data frame. Participants and variable should be in long format such that row *t* represents observations for all variables at time point *t* for a participant. The next row, *t + 1*, represents the next measurement occasion for that same participant. The next participant's data should immediately follow, in the same pattern, after the previous participant `data` should have an ID variable labeled `"ID"`; otherwise, it is assumed that the data represent the population For groups, `data` should have a Group variable labeled `"Group"`; otherwise, it is assumed that there are no groups in `data` Arguments `id` and `group` can be specified to tell the function which column in `data` it should use as the ID and Group variable, respectively A measurement occasion variable is not necessary and should be *removed* from the data before proceeding with the analysis
`id`	Numeric or character (length = 1). Number or name of the column identifying each individual. Defaults to `NULL`
`group`	Numeric or character (length = 1). Number of the column identifying group membership. Defaults to `NULL`
`n.embed`	Numeric (length = 1). Defaults to `5`. Number of embedded dimensions (the number of observations to be used in the `Embed` function). For example, an `"n.embed = 5"` will use five consecutive observations to estimate a single derivative
`tau`	Numeric (length = 1). Defaults to `1`. Number of observations to offset successive embeddings in the `Embed` function. Generally recommended to leave "as is"
`delta`	Numeric (length = 1). Defaults to `1`. The time between successive observations in the time series (i.e, lag). Generally recommended to leave "as is"
`use.derivatives`	Numeric (length = 1). Defaults to `1`. The order of the derivative to be used in the analysis. Available options: • `0` — No derivatives; consistent with moving average • `1` — First-order derivatives; interpreted as "velocity" or rate of change over time • `2` — Second-order derivatives; interpreted as "acceleration" or rate of the rate of change over time

Generally recommended to leave "as is"

`level` Character vector (up to length of 3). A character vector indicating which level(s) to estimate:

- `"individual"` — Estimates `EGA` for each individual in `data` (intraindividual structure; requires an `"ID"` column, see `data`)
- `"group"` — Estimates `EGA` for each group in `data` (group structure; requires a `"Group"` column, see `data`)
- `"population"` — Estimates `EGA` across all `data` (interindividual structure)

`corr` Character (length = 1). Method to compute correlations. Defaults to `"auto"`. Available options:

- `"auto"` — Automatically computes appropriate correlations for the data using Pearson's for continuous, polychoric for ordinal, tetrachoric for binary, and polyserial/biserial for ordinal/binary with continuous. To change the number of categories that are considered ordinal, use `ordinal.categories` (see `polychoric.matrix` for more details)
- `"cor_auto"` — Uses `cor_auto` to compute correlations. Arguments can be passed along to the function
- `"pearson"` — Pearson's correlation is computed for all variables regardless of categories
- `"spearman"` — Spearman's rank-order correlation is computed for all variables regardless of categories

For other similarity measures, compute them first and input them into `data` with the sample size (`n`)

`na.data` Character (length = 1). How should missing data be handled? Defaults to `"pairwise"`. Available options:

- `"pairwise"` — Computes correlation for all available cases between two variables
- `"listwise"` — Computes correlation for all complete cases in the dataset

`model` Character (length = 1). Defaults to `"glasso"`. Available options:

- `"BGGM"` — Computes the Bayesian Gaussian Graphical Model. Set argument `ordinal.categories` to determine levels allowed for a variable to be considered ordinal. See `?BGGM::estimate` for more details
- `"glasso"` — Computes the GLASSO with EBIC model selection. See `EBICglasso.qgraph` for more details
- `"TMFG"` — Computes the TMFG method. See `TMFG` for more details

`algorithm` Character or `igraph` `cluster_*` function (length = 1). Defaults to `"walktrap"`. Three options are listed below but all are available (see `community.detection` for other options):

- `"leiden"` — See `cluster_leiden` for more details
- `"louvain"` — By default, `"louvain"` will implement the Louvain algorithm using the consensus clustering method (see `community.consensus` for more information). This function will implement `consensus.method` = `"most_common"` and `consensus.iter` = `1000` unless specified otherwise

- "walktrap" — See cluster_walktrap for more details

uni.method — Character (length = 1). What unidimensionality method should be used? Defaults to "louvain". Available options:

- "expand" — Expands the correlation matrix with four variables correlated 0.50. If number of dimension returns 2 or less in check, then the data are unidimensional; otherwise, regular EGA with no matrix expansion is used. This method was used in the Golino et al.'s (2020) *Psychological Methods* simulation
- "LE" — Applies the Leading Eigenvector algorithm (cluster_leading_eigen) on the empirical correlation matrix. If the number of dimensions is 1, then the Leading Eigenvector solution is used; otherwise, regular EGA is used. This method was used in the Christensen et al.'s (2023) *Behavior Research Methods* simulation
- "louvain" — Applies the Louvain algorithm (cluster_louvain) on the empirical correlation matrix. If the number of dimensions is 1, then the Louvain solution is used; otherwise, regular EGA is used. This method was validated Christensen's (2022) *PsyArXiv* simulation. Consensus clustering can be used by specifying either "consensus.method" or "consensus.iter"

ncores — Numeric (length = 1). Number of cores to use in computing results. Defaults to ceiling(parallel::detectCores() / 2) or half of your computer's processing power. Set to 1 to not use parallel computing

If you're unsure how many cores your computer has, then type: parallel::detectCores()

verbose — Boolean (length = 1). Should progress be displayed? Defaults to TRUE. Set to FALSE to not display progress

... — Additional arguments to be passed on to auto.correlate, network.estimation, community.detection, community.consensus, and EGA

Details

Derivatives for each variable's time series for each participant are estimated using generalized local linear approximation (see glla). EGA is then applied to these derivatives to model how variables are changing together over time. Variables that change together over time are detected as communities

Value

A list containing:

Derivatives — A list containing:

- Estimates — A list the length of the unique IDs containing data frames of zero- to second-order derivatives for each ID in data
- EstimatesDF — A data frame of derivatives across all IDs containing columns of the zero- to second-order derivatives as well as id and group variables (group is automatically set to 1 for all if no group is provided)

dynEGA — A list containing:

- population — If level includes "populaton", then the EGA results for the entire sample

- `group` — If `level` includes `"group"`, then a list containing the `EGA` results for each `group`
- `individual` — If `level` includes `"individual"`, then a list containing the `EGA` results for each `id`

Author(s)

Hudson Golino <hfg9s at virginia.edu> and Alexander P. Christensen <alexpaulchristensen@gmail.com>

References

Generalized local linear approximation
Boker, S. M., Deboeck, P. R., Edler, C., & Keel, P. K. (2010) Generalized local linear approximation of derivatives from time series. In S.-M. Chow, E. Ferrer, & F. Hsieh (Eds.), *The Notre Dame series on quantitative methodology. Statistical methods for modeling human dynamics: An interdisciplinary dialogue*, (p. 161-178). *Routledge/Taylor & Francis Group*.

Deboeck, P. R., Montpetit, M. A., Bergeman, C. S., & Boker, S. M. (2009) Using derivative estimates to describe intraindividual variability at multiple time scales. *Psychological Methods*, *14(4)*, 367-386.

Original dynamic EGA implementation
Golino, H., Christensen, A. P., Moulder, R. G., Kim, S., & Boker, S. M. (2021). Modeling latent topics in social media using Dynamic Exploratory Graph Analysis: The case of the right-wing and left-wing trolls in the 2016 US elections. *Psychometrika*.

Time delay embedding procedure
Savitzky, A., & Golay, M. J. (1964). Smoothing and differentiation of data by simplified least squares procedures. *Analytical Chemistry*, *36(8)*, 1627-1639.

See Also

`plot.EGAnet` for plot usage in `EGAnet`

Examples

```
# Population structure
simulated_population <- dynEGA(
  data = sim.dynEGA, level = "population"
  # uses simulated data in package
  # useful to understand how data should be structured
)

# Group structure
simulated_group <- dynEGA(
  data = sim.dynEGA, level = "group"
  # uses simulated data in package
  # useful to understand how data should be structured
)

## Not run:
# Individual structure
simulated_individual <- dynEGA(
```

```
  data = sim.dynEGA, level = "individual",
  ncores = 2, # use more for quicker results
  verbose = TRUE # progress bar
)

# Population, group, and individual structure
simulated_all <- dynEGA(
  data = sim.dynEGA,
  level = c("individual", "group", "population"),
  ncores = 2, # use more for quicker results
  verbose = TRUE # progress bar
)

# Plot population
plot(simulated_all$dynEGA$population)

# Plot groups
plot(simulated_all$dynEGA$group)

# Plot individual
plot(simulated_all$dynEGA$individual, id = 1)

# Step through all plots
# Unless `id` is specified, 4 random IDs
# will be drawn from individuals
plot(simulated_all)
## End(Not run)
```

`dynEGA.ind.pop` *Intra- and Inter-individual* dynEGA

Description

A wrapper function to estimate both intraindividiual (`level = "individual"`) and interindividual (`level = "population"`) structures using dynEGA

Usage

```
dynEGA.ind.pop(
  data,
  id = NULL,
  n.embed = 5,
  tau = 1,
  delta = 1,
  use.derivatives = 1,
  corr = c("auto", "cor_auto", "pearson", "spearman"),
  na.data = c("pairwise", "listwise"),
  model = c("BGGM", "glasso", "TMFG"),
```

```
  algorithm = c("leiden", "louvain", "walktrap"),
  uni.method = c("expand", "LE", "louvain"),
  ncores,
  verbose = TRUE,
  ...
)
```

Arguments

`data`	Matrix or data frame. Participants and variable should be in long format such that row *t* represents observations for all variables at time point *t* for a participant. The next row, *t + 1*, represents the next measurement occasion for that same participant. The next participant's data should immediately follow, in the same pattern, after the previous participant `data` should have an ID variable labeled `"ID"`; otherwise, it is assumed that the data represent the population For groups, `data` should have a Group variable labeled `"Group"`; otherwise, it is assumed that there are no groups in `data` Arguments `id` and `group` can be specified to tell the function which column in `data` it should use as the ID and Group variable, respectively A measurement occasion variable is not necessary and should be *removed* from the data before proceeding with the analysis
`id`	Numeric or character (length = 1). Number or name of the column identifying each individual. Defaults to `NULL`
`n.embed`	Numeric (length = 1). Defaults to `5`. Number of embedded dimensions (the number of observations to be used in the `Embed` function). For example, an `"n.embed = 5"` will use five consecutive observations to estimate a single derivative
`tau`	Numeric (length = 1). Defaults to `1`. Number of observations to offset successive embeddings in the `Embed` function. Generally recommended to leave "as is"
`delta`	Numeric (length = 1). Defaults to `1`. The time between successive observations in the time series (i.e, lag). Generally recommended to leave "as is"
`use.derivatives`	Numeric (length = 1). Defaults to `1`. The order of the derivative to be used in the analysis. Available options: • `0` — No derivatives; consistent with moving average • `1` — First-order derivatives; interpreted as "velocity" or rate of change over time • `2` — Second-order derivatives; interpreted as "acceleration" or rate of the rate of change over time Generally recommended to leave "as is"
`corr`	Character (length = 1). Method to compute correlations. Defaults to `"auto"`. Available options: • `"auto"` — Automatically computes appropriate correlations for the data using Pearson's for continuous, polychoric for ordinal, tetrachoric for binary,

and polyserial/biserial for ordinal/binary with continuous. To change the number of categories that are considered ordinal, use `ordinal.categories` (see `polychoric.matrix` for more details)

- `"cor_auto"` — Uses `cor_auto` to compute correlations. Arguments can be passed along to the function
- `"pearson"` — Pearson's correlation is computed for all variables regardless of categories
- `"spearman"` — Spearman's rank-order correlation is computed for all variables regardless of categories

For other similarity measures, compute them first and input them into `data` with the sample size (`n`)

`na.data` Character (length = 1). How should missing data be handled? Defaults to `"pairwise"`. Available options:

- `"pairwise"` — Computes correlation for all available cases between two variables
- `"listwise"` — Computes correlation for all complete cases in the dataset

`model` Character (length = 1). Defaults to `"glasso"`. Available options:

- `"BGGM"` — Computes the Bayesian Gaussian Graphical Model. Set argument `ordinal.categories` to determine levels allowed for a variable to be considered ordinal. See `?BGGM::estimate` for more details
- `"glasso"` — Computes the GLASSO with EBIC model selection. See `EBICglasso.qgraph` for more details
- `"TMFG"` — Computes the TMFG method. See `TMFG` for more details

`algorithm` Character or `igraph` `cluster_*` function (length = 1). Defaults to `"walktrap"`. Three options are listed below but all are available (see `community.detection` for other options):

- `"leiden"` — See `cluster_leiden` for more details
- `"louvain"` — By default, `"louvain"` will implement the Louvain algorithm using the consensus clustering method (see `community.consensus` for more information). This function will implement `consensus.method = "most_common"` and `consensus.iter = 1000` unless specified otherwise
- `"walktrap"` — See `cluster_walktrap` for more details

`uni.method` Character (length = 1). What unidimensionality method should be used? Defaults to `"louvain"`. Available options:

- `"expand"` — Expands the correlation matrix with four variables correlated 0.50. If number of dimension returns 2 or less in check, then the data are unidimensional; otherwise, regular EGA with no matrix expansion is used. This method was used in the Golino et al.'s (2020) *Psychological Methods* simulation
- `"LE"` — Applies the Leading Eigenvector algorithm (`cluster_leading_eigen`) on the empirical correlation matrix. If the number of dimensions is 1, then the Leading Eigenvector solution is used; otherwise, regular EGA is used. This method was used in the Christensen et al.'s (2023) *Behavior Research Methods* simulation

- `"louvain"` — Applies the Louvain algorithm (`cluster_louvain`) on the empirical correlation matrix. If the number of dimensions is 1, then the Louvain solution is used; otherwise, regular EGA is used. This method was validated Christensen's (2022) *PsyArXiv* simulation. Consensus clustering can be used by specifying either `"consensus.method"` or `"consensus.iter"`

`ncores` Numeric (length = 1). Number of cores to use in computing results. Defaults to `ceiling(parallel::detectCores() / 2)` or half of your computer's processing power. Set to `1` to not use parallel computing

If you're unsure how many cores your computer has, then type: `parallel::detectCores()`

`verbose` Boolean (length = 1). Should progress be displayed? Defaults to `TRUE`. Set to `FALSE` to not display progress

`...` Additional arguments to be passed on to `auto.correlate`, `network.estimation`, `community.detection`, `community.consensus`, and `EGA`

Value

Same output as `EGAnet{dynEGA}` returning list objects for `level = "individual"` and `level = "population"`

Author(s)

Hudson Golino <hfg9s at virginia.edu>

See Also

`plot.EGAnet` for plot usage in `EGAnet`

Examples

```
# Obtain data
sim.dynEGA <- sim.dynEGA # bypasses CRAN checks

## Not run:
# Dynamic EGA individual and population structure
dyn.ega1 <- dynEGA.ind.pop(
  data = sim.dynEGA, n.embed = 5, tau = 1,
  delta = 1, id = 25, use.derivatives = 1,
  ncores = 2, corr = "pearson"
)
## End(Not run)
```

EBICglasso.qgraph EBICglasso *from* qgraph *1.4.4*

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 & Tibshirani, 2008). The tuning parameter is chosen using the Extended Bayesian Information criterion (EBIC) described by Foygel & Drton (2010).

Usage

```
EBICglasso.qgraph(
  data,
  n = NULL,
  corr = c("auto", "cor_auto", "cosine", "pearson", "spearman"),
  na.data = c("pairwise", "listwise"),
  gamma = 0.5,
  penalize.diagonal = FALSE,
  nlambda = 100,
  lambda.min.ratio = 0.1,
  returnAllResults = FALSE,
  penalizeMatrix,
  countDiagonal = FALSE,
  refit = FALSE,
  model.selection = c("EBIC", "JSD"),
  verbose = FALSE,
  ...
)
```

Arguments

data	Matrix or data frame. Should consist only of variables to be used in the analysis
n	Numeric (length = 1). Sample size if data provided is a correlation matrix
corr	Character (length = 1). Method to compute correlations. Defaults to "auto". Available options:

- "auto" — Automatically computes appropriate correlations for the data using Pearson's for continuous, polychoric for ordinal, tetrachoric for binary, and polyserial/biserial for ordinal/binary with continuous. To change the number of categories that are considered ordinal, use ordinal.categories (see polychoric.matrix for more details)
- "cor_auto" — Uses cor_auto to compute correlations. Arguments can be passed along to the function
- "cosine" — Uses cosine to compute cosine similarity
- "pearson" — Pearson's correlation is computed for all variables regardless of categories

- "spearman" — Spearman's rank-order correlation is computed for all variables regardless of categories

For other similarity measures, compute them first and input them into data with the sample size (n)

na.data Character (length = 1). How should missing data be handled? Defaults to "pairwise". Available options:

- "pairwise" — Computes correlation for all available cases between two variables
- "listwise" — Computes correlation for all complete cases in the dataset

gamma Numeric (length = 1) EBIC tuning parameter. Defaults to 0.50 and is generally a good choice. Setting to 0 will cause regular BIC to be used

penalize.diagonal
Boolean (length = 1). Should the diagonal be penalized? Defaults to FALSE

nlambda Numeric (length = 1). Number of lambda values to test. Defaults to 100

lambda.min.ratio
Numeric (length = 1). Ratio of lowest lambda value compared to maximal lambda. Defaults to 0.1. **NOTE** qgraph sets the default to 0.01

returnAllResults
Boolean (length = 1). Whether all results should be returned. Defaults to FALSE (network only). Set to TRUE to access glassopath output

penalizeMatrix Boolean matrix. Optional logical matrix to indicate which elements are penalized

countDiagonal Boolean (length = 1). Should diagonal be counted in EBIC computation? Defaults to FALSE. Set to TRUE to mimic qgraph < 1.3 behavior (not recommended!)

refit Boolean (length = 1). Should the optimal graph be refitted without LASSO regularization? Defaults to FALSE

model.selection
Character (length = 1). How lambda should be selected within GLASSO. Defaults to "EBIC". "JSD" is experimental and should not be used otherwise

verbose Boolean (length = 1). Whether messages and (insignificant) warnings should be output. Defaults to FALSE (silent calls). Set to TRUE to see all messages and warnings for every function call

... 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, lambda_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.

Value

A partial correlation matrix

Author(s)

Sacha Epskamp; for maintanence, Hudson Golino <hfg9s at virginia.edu> and Alexander P. Christensen <alexpaulchristensen at gmail.com>

References

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

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

glasso package
Friedman, J., Hastie, T., & Tibshirani, R. (2011). glasso: Graphical lasso-estimation of Gaussian graphical models. R package version 1.7.

Tutorial on EBICglasso
Epskamp, S., & Fried, E. I. (2018). A tutorial on regularized partial correlation networks. *Psychological Methods*, *23*(4), 617–634.

Examples

```
# Obtain data
wmt <- wmt2[,7:24]

# Compute graph with tuning = 0 (BIC)
BICgraph <- EBICglasso.qgraph(data = wmt, gamma = 0)

# Compute graph with tuning = 0.5 (EBIC)
EBICgraph <- EBICglasso.qgraph(data = wmt, gamma = 0.5)
```

EGA *Exploratory Graph Analysis*

Description

Estimates the number of communities (dimensions) of a dataset or correlation matrix using a network estimation method (Golino & Epskamp, 2017; Golino et al., 2020). After, a community detection algorithm is applied (Christensen et al., 2023) for multidimensional data. A unidimensional check is also applied based on findings from Golino et al. (2020) and Christensen (2023)

Usage

```
EGA(
  data,
  n = NULL,
  corr = c("auto", "cor_auto", "cosine", "pearson", "spearman"),
```

```
  na.data = c("pairwise", "listwise"),
  model = c("BGGM", "glasso", "TMFG"),
  algorithm = c("leiden", "louvain", "walktrap"),
  uni.method = c("expand", "LE", "louvain"),
  plot.EGA = TRUE,
  verbose = FALSE,
  ...
)
```

Arguments

`data`	Matrix or data frame. Should consist only of variables to be used in the analysis. Can be raw data or a correlation matrix
`n`	Numeric (length = 1). Sample size if `data` provided is a correlation matrix
`corr`	Character (length = 1). Method to compute correlations. Defaults to `"auto"`. Available options: • `"auto"` — Automatically computes appropriate correlations for the data using Pearson's for continuous, polychoric for ordinal, tetrachoric for binary, and polyserial/biserial for ordinal/binary with continuous. To change the number of categories that are considered ordinal, use `ordinal.categories` (see `polychoric.matrix` for more details) • `"cor_auto"` — Uses `cor_auto` to compute correlations. Arguments can be passed along to the function • `"cosine"` — Uses `cosine` to compute cosine similarity • `"pearson"` — Pearson's correlation is computed for all variables regardless of categories • `"spearman"` — Spearman's rank-order correlation is computed for all variables regardless of categories For other similarity measures, compute them first and input them into `data` with the sample size (`n`)
`na.data`	Character (length = 1). How should missing data be handled? Defaults to `"pairwise"`. Available options: • `"pairwise"` — Computes correlation for all available cases between two variables • `"listwise"` — Computes correlation for all complete cases in the dataset
`model`	Character (length = 1). Defaults to `"glasso"`. Available options: • `"BGGM"` — Computes the Bayesian Gaussian Graphical Model. Set argument `ordinal.categories` to determine levels allowed for a variable to be considered ordinal. See `?BGGM::estimate` for more details • `"glasso"` — Computes the GLASSO with EBIC model selection. See `EBICglasso.qgraph` for more details • `"TMFG"` — Computes the TMFG method. See `TMFG` for more details
`algorithm`	Character or `igraph` `cluster_*` function (length = 1). Defaults to `"walktrap"`. Three options are listed below but all are available (see `community.detection` for other options):

- "leiden" — See cluster_leiden for more details
- "louvain" — By default, "louvain" will implement the Louvain algorithm using the consensus clustering method (see community.consensus for more information). This function will implement consensus.method = "most_common" and consensus.iter = 1000 unless specified otherwise
- "walktrap" — See cluster_walktrap for more details

uni.method — Character (length = 1). What unidimensionality method should be used? Defaults to "louvain". Available options:

- "expand" — Expands the correlation matrix with four variables correlated 0.50. If number of dimension returns 2 or less in check, then the data are unidimensional; otherwise, regular EGA with no matrix expansion is used. This method was used in the Golino et al.'s (2020) *Psychological Methods* simulation
- "LE" — Applies the Leading Eigenvector algorithm (cluster_leading_eigen) on the empirical correlation matrix. If the number of dimensions is 1, then the Leading Eigenvector solution is used; otherwise, regular EGA is used. This method was used in the Christensen et al.'s (2023) *Behavior Research Methods* simulation
- "louvain" — Applies the Louvain algorithm (cluster_louvain) on the empirical correlation matrix. If the number of dimensions is 1, then the Louvain solution is used; otherwise, regular EGA is used. This method was validated Christensen's (2022) *PsyArXiv* simulation. Consensus clustering can be used by specifying either "consensus.method" or "consensus.iter"

plot.EGA — Boolean (length = 1). Defaults to TRUE. Whether the plot should be returned with the results. Set to FALSE for no plot

verbose — Boolean (length = 1). Whether messages and (insignificant) warnings should be output. Defaults to FALSE (silent calls). Set to TRUE to see all messages and warnings for every function call

... — Additional arguments to be passed on to auto.correlate, network.estimation, community.detection, community.consensus, and community.unidimensional

Value

Returns a list containing:

network — A matrix containing a network estimated using link[EGAnet]{network.estimation}

wc — A vector representing the community (dimension) membership of each node in the network. NA values mean that the node was disconnected from the network

n.dim — A scalar of how many total dimensions were identified in the network

correlation — The zero-order correlation matrix

n — Number of cases in data

dim.variables — An ordered matrix of item allocation

TEFI — link[EGAnet]{tefi} for the estimated structure

plot.EGA — Plot output if plot.EGA = TRUE

Author(s)

Hudson Golino <hfg9s at virginia.edu>, Alexander P. Christensen <alexpaulchristensen at gmail.com>, Maria Dolores Nieto <acinodam at gmail.com> and Luis E. Garrido <garrido.luiseduardo at gmail.com>

References

Original simulation and implementation of EGA
Golino, H. F., & Epskamp, S. (2017). Exploratory graph analysis: A new approach for estimating the number of dimensions in psychological research. *PLoS ONE*, *12*, e0174035.

Current implementation of EGA, introduced unidimensional checks, continuous and dichotomous data
Golino, H., Shi, D., Christensen, A. P., Garrido, L. E., Nieto, M. D., Sadana, R., & Thiyagarajan, J. A. (2020). Investigating the performance of Exploratory Graph Analysis and traditional techniques to identify the number of latent factors: A simulation and tutorial. *Psychological Methods*, *25*, 292-320.

Compared all *igraph* community detection algorithms, introduced Louvain algorithm, simulation with continuous and polytomous data
Also implements the Leading Eigenvalue unidimensional method
Christensen, A. P., Garrido, L. E., Pena, K. G., & Golino, H. (2023). Comparing community detection algorithms in psychological data: A Monte Carlo simulation. *Behavior Research Methods*.

Comprehensive unidimensionality simulation
Christensen, A. P. (2023). Unidimensional community detection: A Monte Carlo simulation, grid search, and comparison. *PsyArXiv*.

Compared all `igraph` community detection algorithms, simulation with continuous and polytomous data
Christensen, A. P., Garrido, L. E., Guerra-Pena, K., & Golino, H. (2023). Comparing community detection algorithms in psychometric networks: A Monte Carlo simulation. *Behavior Research Methods*.

See Also

`plot.EGAnet` for plot usage in `EGAnet`

Examples

```
# Obtain data
wmt <- wmt2[,7:24]

# Estimate EGA
ega.wmt <- EGA(
  data = wmt,
  plot.EGA = FALSE # No plot for CRAN checks
)

# Print results
print(ega.wmt)

# Estimate EGAtmfg
```

```
ega.wmt.tmfg <- EGA(
  data = wmt, model = "TMFG",
  plot.EGA = FALSE # No plot for CRAN checks
)

# Estimate EGA with Louvain algorithm
ega.wmt.louvain <- EGA(
  data = wmt, algorithm = "louvain",
  plot.EGA = FALSE # No plot for CRAN checks
)

# Estimate EGA with an {igraph} function (Fast-greedy)
ega.wmt.greedy <- EGA(
  data = wmt,
  algorithm = igraph::cluster_fast_greedy,
  plot.EGA = FALSE # No plot for CRAN checks
)
```

`EGA.estimate` *Estimates* `EGA` *for Multidimensional Structures*

Description

A basic function to estimate `EGA` for multidimensional structures. This function does *not* include the unidimensional check and it does not plot the results. This function can be used as a streamlined approach for quick `EGA` estimation when unidimensionality or visualization is not a priority

Usage

```
EGA.estimate(
  data,
  n = NULL,
  corr = c("auto", "cor_auto", "cosine", "pearson", "spearman"),
  na.data = c("pairwise", "listwise"),
  model = c("BGGM", "glasso", "TMFG"),
  algorithm = c("leiden", "louvain", "walktrap"),
  verbose = FALSE,
  ...
)
```

Arguments

`data`	Matrix or data frame. Should consist only of variables to be used in the analysis
`n`	Numeric (length = 1). Sample size if `data` provided is a correlation matrix
`corr`	Character (length = 1). Method to compute correlations. Defaults to `"auto"`. Available options:

- "auto" — Automatically computes appropriate correlations for the data using Pearson's for continuous, polychoric for ordinal, tetrachoric for binary, and polyserial/biserial for ordinal/binary with continuous. To change the number of categories that are considered ordinal, use ordinal.categories (see polychoric.matrix for more details)
- "cor_auto" — Uses cor_auto to compute correlations. Arguments can be passed along to the function
- "cosine" — Uses cosine to compute cosine similarity
- "pearson" — Pearson's correlation is computed for all variables regardless of categories
- "spearman" — Spearman's rank-order correlation is computed for all variables regardless of categories

For other similarity measures, compute them first and input them into data with the sample size (n)

na.data Character (length = 1). How should missing data be handled? Defaults to "pairwise". Available options:

- "pairwise" — Computes correlation for all available cases between two variables
- "listwise" — Computes correlation for all complete cases in the dataset

model Character (length = 1). Defaults to "glasso". Available options:

- "BGGM" — Computes the Bayesian Gaussian Graphical Model. Set argument ordinal.categories to determine levels allowed for a variable to be considered ordinal. See ?BGGM::estimate for more details
- "glasso" — Computes the GLASSO with EBIC model selection. See EBICglasso.qgraph for more details
- "TMFG" — Computes the TMFG method. See TMFG for more details

algorithm Character or igraph cluster_* function (length = 1). Defaults to "walktrap". Three options are listed below but all are available (see community.detection for other options):

- "leiden" — See cluster_leiden for more details
- "louvain" — By default, "louvain" will implement the Louvain algorithm using the consensus clustering method (see community.consensus for more information). This function will implement consensus.method = "most_common" and consensus.iter = 1000 unless specified otherwise
- "walktrap" — See cluster_walktrap for more details

verbose Boolean (length = 1). Whether messages and (insignificant) warnings should be output. Defaults to FALSE (silent calls). Set to TRUE to see all messages and warnings for every function call

... Additional arguments to be passed on to auto.correlate, network.estimation, community.detection, and community.consensus

Value

Returns a list containing:

network	A matrix containing a network estimated using link[EGAnet]{network.estimation}
wc	A vector representing the community (dimension) membership of each node in the network. NA values mean that the node was disconnected from the network
n.dim	A scalar of how many total dimensions were identified in the network
cor.data	The zero-order correlation matrix
n	Number of cases in data

Author(s)

Alexander P. Christensen <alexpaulchristensen at gmail.com> and Hudson Golino <hfg9s at virginia.edu>

References

Original simulation and implementation of EGA
Golino, H. F., & Epskamp, S. (2017). Exploratory graph analysis: A new approach for estimating the number of dimensions in psychological research. *PLoS ONE*, *12*, e0174035.

Introduced unidimensional checks, simulation with continuous and dichotomous data
Golino, H., Shi, D., Christensen, A. P., Garrido, L. E., Nieto, M. D., Sadana, R., & Thiyagarajan, J. A. (2020). Investigating the performance of Exploratory Graph Analysis and traditional techniques to identify the number of latent factors: A simulation and tutorial. *Psychological Methods*, *25*, 292-320.

Compared all igraph community detection algorithms, simulation with continuous and polytomous data
Christensen, A. P., Garrido, L. E., Guerra-Pena, K., & Golino, H. (2023). Comparing community detection algorithms in psychometric networks: A Monte Carlo simulation. *Behavior Research Methods*.

See Also

plot.EGAnet for plot usage in EGAnet

Examples

```
# Obtain data
wmt <- wmt2[,7:24]

# Estimate EGA
ega.wmt <- EGA.estimate(data = wmt)

# Estimate EGA with TMFG
ega.wmt.tmfg <- EGA.estimate(data = wmt, model = "TMFG")

# Estimate EGA with an {igraph} function (Fast-greedy)
ega.wmt.greedy <- EGA.estimate(
  data = wmt,
  algorithm = igraph::cluster_fast_greedy
)
```

EGA.fit — EGA *Optimal Model Fit using the Total Entropy Fit Index (*tefi*)*

Description

Estimates the best fitting model using EGA. The number of steps in the cluster_walktrap detection algorithm is varied and unique community solutions are compared using tefi.

Usage

```
EGA.fit(
  data,
  n = NULL,
  corr = c("auto", "cor_auto", "cosine", "pearson", "spearman"),
  na.data = c("pairwise", "listwise"),
  model = c("BGGM", "glasso", "TMFG"),
  algorithm = c("leiden", "louvain", "walktrap"),
  plot.EGA = TRUE,
  verbose = FALSE,
  ...
)
```

Arguments

data	Matrix or data frame. Should consist only of variables to be used in the analysis
n	Numeric (length = 1). Sample size if data is a correlation matrix
corr	Character (length = 1). Method to compute correlations. Defaults to "auto". Available options: • "auto" — Automatically computes appropriate correlations for the data using Pearson's for continuous, polychoric for ordinal, tetrachoric for binary, and polyserial/biserial for ordinal/binary with continuous. To change the number of categories that are considered ordinal, use ordinal.categories (see polychoric.matrix for more details) • "cor_auto" — Uses cor_auto to compute correlations. Arguments can be passed along to the function • "cosine" — Uses cosine to compute cosine similarity • "pearson" — Pearson's correlation is computed for all variables regardless of categories • "spearman" — Spearman's rank-order correlation is computed for all variables regardless of categories For other similarity measures, compute them first and input them into data with the sample size (n)
na.data	Character (length = 1). How should missing data be handled? Defaults to "pairwise". Available options:

- "pairwise" — Computes correlation for all available cases between two variables
- "listwise" — Computes correlation for all complete cases in the dataset

model — Character (length = 1). Defaults to "glasso". Available options:

- "BGGM" — Computes the Bayesian Gaussian Graphical Model. Set argument ordinal.categories to determine levels allowed for a variable to be considered ordinal. See ?BGGM::estimate for more details
- "glasso" — Computes the GLASSO with EBIC model selection. See EBICglasso.qgraph for more details
- "TMFG" — Computes the TMFG method. See TMFG for more details

algorithm — Character or igraph cluster_* function. Three options are listed below but all are available (see community.detection for other options):

- "leiden" — See cluster_leiden for more details. *Note*: The Leiden algorithm will default to the Constant Potts Model objective function (objective_function = "CPM"). Set objective_function = "modularity" to use modularity instead (see examples). By default, searches along resolutions from 0 to max(abs(network)) or the maximum absolute edge weight in the network in 0.01 increments (resolution_parameter = seq.int(0, max(abs(network)), 0.01)). For modularity, searches along resolutions from 0 to 2 in 0.05 increments (resolution_parameter = seq.int(0, 2, 0.05)) by default. Use the argument resolution_parameter to change the search parameters (see examples)
- "louvain" — See community.consensus for more details. By default, searches along resolutions from 0 to 2 in 0.05 increments (resolution_parameter = seq.int(0, 2, 0.05)). Use the argument resolution_parameter to change the search parameters (see examples)
- "walktrap" — This algorithm is the default. See cluster_walktrap for more details. By default, searches along 3 to 8 steps (steps = 3:8). Use the argument steps to change the search parameters (see examples)

plot.EGA — Boolean. If TRUE, returns a plot of the network and its estimated dimensions. Defaults to TRUE

verbose — Boolean. Whether messages and (insignificant) warnings should be output. Defaults to FALSE (silent calls). Set to TRUE to see all messages and warnings for every function call

... — Additional arguments to be passed on to auto.correlate, network.estimation, community.detection, community.consensus, and EGA.estimate

Value

Returns a list containing:

EGA — EGA results of the best fitting solution

EntropyFit — tefi fit values for each solution

Lowest.EntropyFit — The best fitting solution based on tefi

parameter.space — Parameter values used in search space

Author(s)

Hudson Golino <hfg9s at virginia.edu> and Alexander P. Christensen <alexpaulchristensen@gmail.com>

References

Entropy fit measures
Golino, H., Moulder, R. G., Shi, D., Christensen, A. P., Garrido, L. E., Neito, M. D., Nesselroade, J., Sadana, R., Thiyagarajan, J. A., & Boker, S. M. (in press). Entropy fit indices: New fit measures for assessing the structure and dimensionality of multiple latent variables. *Multivariate Behavioral Research*.

Simulation for EGA.fit
Jamison, L., Christensen, A. P., & Golino, H. (under review). Optimizing Walktrap's community detection in networks using the Total Entropy Fit Index. *PsyArXiv*.

Leiden algorithm
Traag, V. A., Waltman, L., & Van Eck, N. J. (2019). From Louvain to Leiden: guaranteeing well-connected communities. *Scientific Reports*, *9*(1), 1-12.

Louvain algorithm
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*(10), P10008.

Walktrap algorithm
Pons, P., & Latapy, M. (2006). Computing communities in large networks using random walks. *Journal of Graph Algorithms and Applications*, *10*, 191-218.

See Also

`plot.EGAnet` for plot usage in `EGAnet`

Examples

```
# Load data
wmt <- wmt2[,7:24]

# Estimate optimal EGA with Walktrap
fit.walktrap <- EGA.fit(
  data = wmt, algorithm = "walktrap",
  steps = 3:8, # default
  plot.EGA = FALSE # no plot for CRAN checks
)

# Estimate optimal EGA with Leiden and CPM
fit.leiden <- EGA.fit(
  data = wmt, algorithm = "leiden",
  objective_function = "CPM", # default
  # resolution_parameter = seq.int(0, max(abs(network)), 0.01),
  # For CPM, the default max resolution parameter
  # is set to the largest absolute edge in the network
  plot.EGA = FALSE # no plot for CRAN checks
)
```

```
# Estimate optimal EGA with Leiden and modularity
fit.leiden <- EGA.fit(
  data = wmt, algorithm = "leiden",
  objective_function = "modularity",
  resolution_parameter = seq.int(0, 2, 0.05),
  # default for modularity
  plot.EGA = FALSE # no plot for CRAN checks
)

## Not run:
# Estimate optimal EGA with Louvain
fit.louvain <- EGA.fit(
  data = wmt, algorithm = "louvain",
  resolution_parameter = seq.int(0, 2, 0.05), # default
  plot.EGA = FALSE # no plot for CRAN checks
)
## End(Not run)
```

`ega.wmt` EGA *Network of* wmt2*Data*

Description

EGA results from `ega.wmt <- EGA(wmt2[,7:24])` for the Wiener Matrizen-Test (WMT-2)

Usage

```
data(ega.wmt)
```

Format

A list with 8 objects (see **Value** in EGA)

Examples

```
data("ega.wmt")
```

EGAnet-plot *S3 Plot Methods for* EGAnet

Description

General usage for plots created by EGAnet's S3 methods. Plots across the EGAnet package leverage GGally's ggnet2 and ggplot2's ggplot.

Most plots allow the full usage of the gg* series functionality and therefore plotting arguments should be referenced through those packages rather than here in EGAnet.

The sections below list the functions and their usage for the S3 plot methods. The plot methods are intended to be generic and without many arguments so that nearly all arguments are passed to ggnet2 and ggplot.

There are some constraints placed on certain plots to keep the EGAnet style throughout the (network) plots in the package, so be aware that if some settings are not changing your plot output, then these settings might be fixed to maintain the EGAnet style

General Usage

```
plot(x, ...)

plot.dynEGA(x, base = 1, id = NULL, ...)

plot.dynEGA.Group(x, base = 1, ...)

plot.dynEGA.Individual(x, base = 1, id = NULL, ...)

plot.hierEGA(
  x, plot.type = c("multilevel", "separate"),
  color.match = FALSE, ...
)

plot.invariance(x, p_type = c("p", "p_BH"), p_value = 0.05, ...)
```

General Arguments

- x — EGAnet object with available S3 plot method (see full list below)
- color.palette — Character (vector). Either a character (length = 1) from the pre-defined palettes in color_palette_EGA or character (length = total number of communities) using HEX codes (see **Color Palettes** and **Examples** sections)
- layout — Character (length = 1). Layouts can be set using gplot.layout and the ending layout name; for example, gplot.layout.circle can be set in these functions using layout = "circle" or mode = "circle" (see **Examples**)
- base — Numeric (length = 1). Plot to be used as the base for the configuration of the networks. Uses the number of the order in which the plots are input. Defaults to 1 or the first plot

- id — Numeric index(es) or character name(s). IDs to use when plotting dynEGA level = "individual". Defaults to NULL or 4 IDs drawn at random
- plot.type — Character (length = 1). Whether hierEGA networks should plotted in a stacked, "multilevel" fashion or as "separate" plots. Defaults to "multilevel"
- color.match — Boolean (length = 1). Whether lower order community colors in the hierEGA plot should be "matched" and used as the border color for the higher order communities. Defaults to FALSE
- p_type — Character (length = 1). Type of *p*-value when plotting invariance. Defaults to "p" or uncorrected *p*-value. Set to "p_BH" for the Benjamini-Hochberg corrected *p*-value
- p_value — Numeric (length = 1). The *p*-value to use alongside p_type when plotting invariance. Defaults to 0.05
- ... — Additional arguments to pass on to ggnet2 and gplot.layout (see **Examples**)

*EGA Plots

bootEGA, dynEGA, EGA, EGA.estimate, EGA.fit, hierEGA, invariance, riEGA

All Available S3 Plot Methods

boot.ergoInfo, bootEGA, dynEGA, dynEGA.Group, dynEGA.Individual, dynEGA.Population, EGA, EGA.estimate, EGA.fit, hierEGA, infoCluster, invariance, itemStability, riEGA

Color Palettes

color_palette_EGA will implement some color palettes in EGAnet. The main EGAnet style palette is "polychrome". This palette currently has 40 colors but there will likely be a need to expand it further (e.g., hierEGA demands a lot of colors).

The color.palette argument will also accept HEX code colors that are the same length as the number of communities in the plot.

In any network plots, the color.palette argument can be used to select color palettes from color_palette_EGA as well as those in the color scheme of RColorBrewer

For more worked examples than below, see Plots in {EGAnet}

Examples

```
# Using different arguments in {GGally}'s `ggnet2`
plot(ega.wmt, node.size = 6, edge.size = 4)

# Using a different layout in {sna}'s `gplot.layout`
plot(ega.wmt, layout = "circle") # 'layout' argument
plot(ega.wmt, mode = "circle") # 'mode' argument

# Using different color palettes with `color_palette_EGA`

## Pre-defined palette
plot(ega.wmt, color.palette = "blue.ridge2")

## University of Virginia colors
```

```
plot(ega.wmt, color.palette = c("#232D4B", "#F84C1E"))

## Vanderbilt University colors
## (with additional {GGally} `ggnet2` argument)
plot(
  ega.wmt, color.palette = c("#FFFFFF", "#866D4B"),
  label.color = "#000000"
)
```

EGM *Exploratory Graph Model*

Description

Function to fit the Exploratory Graph Model

Usage

```
EGM(
  data,
  EGM.model = c("standard", "EGA"),
  communities = NULL,
  structure = NULL,
  search = FALSE,
  p.in = NULL,
  p.out = NULL,
 opt = c("AIC", "BIC", "CFI", "chisq", "logLik", "RMSEA", "SRMR", "TEFI", "TEFI.adj",
    "TLI"),
  constrained = TRUE,
  verbose = TRUE,
  ...
)
```

Arguments

`data`	Matrix or data frame. Should consist only of variables to be used in the analysis. Can be raw data or a correlation matrix
`EGM.model`	Character vector (length = 1). Sets the procedure to conduct `EGM`. Available options: • `"EGA"` (default) — Applies `EGA` to obtain the (sparse) regularized network structure, communities, and memberships • `"standard"` — Applies the standard EGM model which estimates communities based on the non-regularized empirical partial correlation matrix and sparsity is set using `p.in` and `p.out`

`communities` Numeric vector (length = 1). Number of communities to use for the `"standard"` type of EGM. Defaults to `NULL`. Providing no input will use the communities and memberships output from the Walktrap algorithm (`cluster_walktrap`) based on the empirical non-regularized partial correlation matrix

`structure` Numeric or character vector (length = `ncol(data)`). Can be theoretical factors or the structure detected by `EGA`. Defaults to `NULL`

`search` Boolean (length = 1). Whether a search over parameters should be conducted. Defaults to `FALSE`. Set to `TRUE` to select a model over a variety of parameters that optimizes the `opt` objective

`p.in` Numeric vector (length = 1). Probability that a node is randomly linked to other nodes in the same community. Within community edges are set to zero based on `quantile(x, prob = 1 - p.in)` ensuring the lowest edge values are set to zero (i.e., most probable to *not* be randomly connected). Only used for `EGM.type = "standard"`. Defaults to `NULL` but must be set

`p.out` Numeric vector (length = 1). Probability that a node is randomly linked to other nodes *not* in the same community. Between community edges are set to zero based on `quantile(x, prob = 1 - p.out)` ensuring the lowest edge values are set to zero (i.e., most probable to *not* be randomly connected). Only used for `EGM.type = "standard"`. Defaults to `NULL` but must be set

`opt` Character vector (length = 1). Fit index used to select from when searching over models (only applies to `EGM.type = "search"`). Available options include:

- `"AIC"`
- `"BIC"`
- `"CFI"`
- `"chisq"`
- `"logLik"`
- `"RMSEA"`
- `"SRMR"`
- `"TEFI"`
- `"TEFI.adj"`
- `"TLI"`

Defaults to `"SRMR"`

`constrained` Boolean (length = 1). Whether memberships of the communities should be added as a constraint when optimizing the network loadings. Defaults to `TRUE` which ensures assigned loadings are guaranteed to never be smaller than any cross-loadings. Set to `FALSE` to freely estimate each loading similar to exploratory factor analysis

`verbose` Boolean (length = 1). Should progress be displayed? Defaults to `TRUE`. Set to `FALSE` to not display progress

`...` Additional arguments to be passed on to `auto.correlate`, `network.estimation`, `community.detection`, `community.consensus`, `community.unidimensional`, `EGA`, and `net.loads`

Author(s)

Hudson F. Golino <hfg9s at virginia.edu> and Alexander P. Christensen <alexpaulchristensen@gmail.com>

Examples

```
# Get depression data
data <- depression[,24:44]

# Estimate EGM (using EGA)
egm_ega <- EGM(data)

# Estimate EGM (using EGA) specifying communities
egm_ega_communities <- EGM(data, communities = 3)

# Estimate EGM (using EGA) specifying structure
egm_ega_structure <- EGM(
  data, structure = c(
    1, 1, 1, 2, 1, 1, 1,
    1, 1, 1, 3, 2, 2, 2,
    2, 3, 3, 3, 3, 3, 2
  )
)

# Estimate EGM (using standard)
egm_standard <- EGM(
  data, EGM.model = "standard",
  communities = 3, # specify number of communities
  p.in = 0.95, # probability of edges *in* each community
  p.out = 0.80 # probability of edges *between* each community
)

## Not run:
# Estimate EGM (using EGA search)
egm_ega_search <- EGM(
  data, EGM.model = "EGA", search = TRUE
)

# Estimate EGM (using EGA search and AIC criterion)
egm_ega_search_AIC <- EGM(
  data, EGM.model = "EGA", search = TRUE, opt = "AIC"
)

# Estimate EGM (using search)
egm_search <- EGM(
  data, EGM.model = "standard", search = TRUE,
  communities = 3, # need communities or structure
  p.in = 0.95 # only need 'p.in'
)
## End(Not run)
```

EGM.compare *Compare* EGM *with EFA*

Description

Estimates an EGM based on EGA and uses the number of communities as the number of dimensions in exploratory factor analysis (EFA) using fa

Usage

```
EGM.compare(data, constrained = FALSE, rotation = "geominQ", ...)
```

Arguments

data	Matrix or data frame. Should consist only of variables to be used in the analysis. Can be raw data or a correlation matrix
constrained	Boolean (length = 1). Whether memberships of the communities should be added as a constraint when optimizing the network loadings. Defaults to FALSE to freely estimate each loading similar to exploratory factor analysis. *Note: This default differs from EGM. Constraining loadings puts EGM at a deficit relative to EFA and therefore biases the comparability between the methods. It's best to leave the default of unconstrained when using this function.*
rotation	Character. A rotation to use to obtain a simpler structure for EFA. For a list of rotations, see rotations for options. Defaults to "geominQ"
...	Additional arguments to be passed on to auto.correlate, network.estimation, community.detection, community.consensus, community.unidimensional, EGA, EGM, net.loads, and fa

Author(s)

Hudson F. Golino <hfg9s at virginia.edu> and Alexander P. Christensen <alexpaulchristensen@gmail.com>

Examples

```
# Get depression data
data <- depression[,24:44]

# Compare EGM (using EGA) with EFA
## Not run:
results <- EGM.compare(data)

# Print summary
summary(results)
## End(Not run)
```

Embed *Time-delay Embedding*

Description

Reorganizes a single observed time series into an embedded matrix. The embedded matrix is constructed with replicates of an individual time series that are offset from each other in time. The function requires two parameters, one that specifies the number of observations to be used (i.e., the number of embedded dimensions) and the other that specifies the number of observations to offset successive embeddings

Usage

```
Embed(x, E, tau)
```

Arguments

x	Numeric vector. An observed time series to be reorganized into a time-delayed embedded matrix.
E	Numeric (length = 1). Number of embedded dimensions or the number of observations to be used. E = 5, for example, will generate a matrix with five columns corresponding to five consecutive observations across each row of the embedded matrix
tau	Numeric (length = 1). Number of observations to offset successive embeddings. A tau of one uses adjacent observations. Default is tau = 1

Value

Returns a numeric matrix

Author(s)

Pascal Deboeck <pascal.deboeck at psych.utah.edu> and Alexander P. Christensen <alexpaulchristensen@gmail.com>

References

Deboeck, P. R., Montpetit, M. A., Bergeman, C. S., & Boker, S. M. (2009) Using derivative estimates to describe intraindividual variability at multiple time scales. *Psychological Methods*, *14*, 367-386.

Examples

```
# A time series with 8 time points
time_series <- 49:56

# Time series embedding
Embed(time_series, E = 5, tau = 1)
```

`entropyFit` *Entropy Fit Index*

Description

Computes the fit of a dimensionality structure using empirical entropy. Lower values suggest better fit of a structure to the data.

Usage

```
entropyFit(data, structure)
```

Arguments

`data`	Matrix or data frame. Contains variables to be used in the analysis
`structure`	Numeric or character vector (length = `ncol(data)`). A vector representing the structure (numbers or labels for each item). Can be theoretical factors or the structure detected by `EGA`

Value

Returns a list containing:

`Total.Correlation`
: The total correlation of the dataset

`Total.Correlation.MM`
: Miller-Madow correction for the total correlation of the dataset

`Entropy.Fit`
: The Entropy Fit Index

`Entropy.Fit.MM`
: Miller-Madow correction for the Entropy Fit Index

`Average.Entropy`
: The average entropy of the dataset

Author(s)

Hudson F. Golino <hfg9s at virginia.edu>, Alexander P. Christensen <alexpaulchristensen@gmail.com> and Robert Moulder <rgm4fd@virginia.edu>

References

Initial formalization and simulation
Golino, H., Moulder, R. G., Shi, D., Christensen, A. P., Garrido, L. E., Nieto, M. D., Nesselroade, J., Sadana, R., Thiyagarajan, J. A., & Boker, S. M. (2020). Entropy fit indices: New fit measures for assessing the structure and dimensionality of multiple latent variables. *Multivariate Behavioral Research*.

Examples

```
# Load data
wmt <- wmt2[,7:24]

## Not run:
# Estimate EGA model
ega.wmt <- EGA(data = wmt)
## End(Not run)

# Compute entropy indices
entropyFit(data = wmt, structure = ega.wmt$wc)
```

`ergoInfo` *Ergodicity Information Index*

Description

Computes the Ergodicity Information Index

Usage

```
ergoInfo(
  dynEGA.object,
  use = c("edge.list", "unweighted", "weighted"),
  shuffles = 5000
)
```

Arguments

`dynEGA.object`	A `dynEGA.ind.pop` object
`use`	Character (length = 1). A string indicating what network element will be used to compute the algorithm complexity, the list of edges or the weights of the network. Defaults to `use = "unweighted"`. Current options are: • `"edge.list"` — Calculates the algorithm complexity using the list of edges • `"unweighted"` — Calculates the algorithm complexity using the binary weights of the encoded prime transformed network. 0 = edge absent and 1 = edge present • `"weighted"` — Calculates the algorithm complexity using the weights of encoded prime-weight transformed network
`shuffles`	Numeric. Number of shuffles used to compute the Kolmogorov complexity. Defaults to `5000`

Value

Returns a list containing:

`PrimeWeight`	The prime-weight encoding of the individual networks
`PrimeWeight.pop`	The prime-weight encoding of the population network
`Kcomp`	The Kolmogorov complexity of the prime-weight encoded individual networks
`Kcomp.pop`	The Kolmogorov complexity of the prime-weight encoded population network
`complexity`	The complexity metric proposed by Santora and Nicosia (2020)
`EII`	The Ergodicity Information Index

Author(s)

Hudson Golino <hfg9s at virginia.edu> and Alexander Christensen <alexpaulchristensen@gmail.com>

References

Original Implementation
Golino, H., Nesselroade, J. R., & Christensen, A. P. (2022). Toward a psychology of individuals: The ergodicity information index and a bottom-up approach for finding generalizations. *PsyArXiv*.

Examples

```
# Obtain data
sim.dynEGA <- sim.dynEGA # bypasses CRAN checks

## Not run:
# Dynamic EGA individual and population structure
dyn.ega1 <- dynEGA.ind.pop(
  data = sim.dynEGA[,-26], n.embed = 5, tau = 1,
  delta = 1, id = 25, use.derivatives = 1,
  ncores = 2, corr = "pearson"
)

# Compute empirical ergodicity information index
eii <- ergoInfo(dyn.ega1)
## End(Not run)
```

---

frobenius *Frobenius Norm (Similarity)*

Description

Computes the Frobenius Norm (Ulitzsch et al., 2023)

Usage

```
frobenius(network1, network2)
```

Arguments

network1	Matrix or data frame. Network to be compared
network2	Matrix or data frame. Second network to be compared

Value

Returns Frobenius Norm

Author(s)

Hudson Golino <hfg9s at virginia.edu> & Alexander P. Christensen <alexander.christensen at Vanderbilt.Edu>

References

Simulation Study
Ulitzsch, E., Khanna, S., Rhemtulla, M., & Domingue, B. W. (2023). A graph theory based similarity metric enables comparison of subpopulation psychometric networks *Psychological Methods*.

Examples

```
# Obtain wmt2 data
wmt <- wmt2[,7:24]

# Set seed (for reproducibility)
set.seed(1234)

# Split data
split1 <- sample(
  1:nrow(wmt), floor(nrow(wmt) / 2)
)
split2 <- setdiff(1:nrow(wmt), split1)

# Obtain split data
data1 <- wmt[split1,]
data2 <- wmt[split2,]

# Perform EBICglasso
glas1 <- EBICglasso.qgraph(data1)
glas2 <- EBICglasso.qgraph(data2)

# Frobenius norm
frobenius(glas1, glas2)
# 0.7070395
```

genTEFI *Generalized Total Entropy Fit Index using Von Neumman's entropy (Quantum Information Theory) for correlation matrices*

Description

Computes the fit (Generalized TEFI) of a hierarchical or correlated bifactor dimensionality structure (or hierEGA objects) using Von Neumman's entropy when the input is a correlation matrix. Lower values suggest better fit of a structure to the data

Usage

```
genTEFI(data, structure = NULL, verbose = TRUE)
```

Arguments

data	Matrix, data frame, or hierEGA object. Can be raw data or correlation matrix
structure	For high-order and correlated bifactor structures, structure should be a list containing: • lower_order — A vector (length = ncol(data)) representing the first-order structure (numbers or labels for each item in each first-order factor or community) • higher_order — A vector (length = ncol(data) or number of lower_order communities)representing the second-order structure (numbers or labels for each item in each second-order factor or community)
verbose	Boolean (length = 1). Whether messages and (insignificant) warnings should be output. Defaults to TRUE to see all messages and warnings for every function call. Set to FALSE to ignore messages and warnings

Value

Returns a three-column data frame of the Generalized Total Entropy Fit Index using Von Neumman's entropy (VN.Entropy.Fit) (first column), as well as Lower.Order.VN - TEFI for the first-order factors (second column), and Higher.Order.VN, the equivalent for the second-order factors.

Author(s)

Hudson Golino <hfg9s at virginia.edu> and Alexander P. Christensen <alexpaulchristensen@gmail.com>

Examples

```
# Example using network scores
opt.hier <- hierEGA(
  data = optimism, scores = "network",
  plot.EGA = FALSE # No plot for CRAN checks
)
```

```
# Compute the Generalized Total Entropy Fit Index
genTEFI(opt.hier)
```

---

glla *Generalized Local Linear Approximation*

Description

Estimates the derivatives of a time series using generalized local linear approximation (GLLA). GLLA is a filtering method for estimating derivatives from data that uses time delay embedding and a variant of Savitzky-Golay filtering to accomplish the task.

Usage

```
glla(x, n.embed, tau, delta, order)
```

Arguments

x	Numeric vector. An observed time series
n.embed	Numeric (length = 1). Number of embedded dimensions (the number of observations to be used in the Embed function)
tau	Numeric (length = 1). Number of observations to offset successive embeddings in the Embed function. A tau of one uses adjacent observations. Default is 1
delta	Numeric (length = 1). The time between successive observations in the time series. Default is 1
order	Numeric (length = 1). The maximum order of the derivative to be estimated. For example, "order = 2" will return a matrix with three columns with the estimates of the observed scores and the first and second derivative for each row of the embedded matrix (i.e. the reorganization of the time series implemented via the Embed function)

Value

Returns a matrix containing *n* columns in which *n* is one plus the maximum order of the derivatives to be estimated via generalized local linear approximation

Author(s)

Hudson Golino <hfg9s at virginia.edu>

References

GLLA implementation

Boker, S. M., Deboeck, P. R., Edler, C., & Keel, P. K. (2010) Generalized local linear approximation of derivatives from time series. In S.-M. Chow, E. Ferrer, & F. Hsieh (Eds.), *The Notre Dame series on quantitative methodology. Statistical methods for modeling human dynamics: An interdisciplinary dialogue*, (p. 161-178). *Routledge/Taylor & Francis Group*.

Deboeck, P. R., Montpetit, M. A., Bergeman, C. S., & Boker, S. M. (2009) Using derivative estimates to describe intraindividual variability at multiple time scales. *Psychological Methods*, *14(4)*, 367-386.

Filtering procedure

Savitzky, A., & Golay, M. J. (1964). Smoothing and differentiation of data by simplified least squares procedures. *Analytical Chemistry*, *36(8)*, 1627-1639.

Examples

```
# A time series with 8 time points
tseries <- 49:56
deriv.tseries <- glla(tseries, n.embed = 4, tau = 1, delta = 1, order = 2)
```

---

hierEGA *Hierarchical* EGA

Description

Estimates EGA using the lower-order solution of the Louvain algorithm (cluster_louvain)to identify the lower-order dimensions and then uses factor or network loadings to estimate factor or network scores, which are used to estimate the higher-order dimensions (for more details, see Jiménez et al., 2023)

Usage

```
hierEGA(
  data,
  loading.method = c("original", "revised"),
  rotation = NULL,
  scores = c("factor", "network"),
  loading.structure = c("simple", "full"),
  impute = c("mean", "median", "none"),
  corr = c("auto", "cor_auto", "pearson", "spearman"),
  na.data = c("pairwise", "listwise"),
  model = c("BGGM", "glasso", "TMFG"),
  lower.algorithm = "louvain",
  higher.algorithm = c("leiden", "louvain", "walktrap"),
  uni.method = c("expand", "LE", "louvain"),
  plot.EGA = TRUE,
```

```
  verbose = FALSE,
  ...
)
```

Arguments

`data`	Matrix or data frame. Should consist only of variables to be used in the analysis (does not accept correlation matrices)
`loading.method`	Character (length = 1). Sets network loading calculation based on implementation described in `"original"` (Christensen & Golino, 2021) or the `"revised"` (Christensen et al., 2024) implementation. Defaults to `"revised"`
`rotation`	Character. A rotation to use to obtain a simpler structure. For a list of rotations, see `rotations` for options. Defaults to `NULL` or no rotation. By setting a rotation, `scores` estimation will be based on the rotated loadings rather than unrotated loadings
`scores`	Character (length = 1). How should scores for the higher-order structure be estimated? Defaults to `"network"` for network scores computed using the `net.scores` function. Set to `"factor"` for factor scores computed using `fa`. Factors scores will be based on **EFA** (as in Jiménez et al., 2023) *Factor scores use the number of communities from* `EGA`*. Estimated factor loadings may not align with these communities. The plots using factor scores will have higher order factors that may not completely map on to the lower order communities. Look at* `$hierarchical$higher_order$lower_loadings` *to determine the composition of the lower order factors.*
`loading.structure`	Character (length = 1). Whether simple structure or the saturated loading matrix should be used when computing scores (`scores = "network"` only). Defaults to `"simple"` `"simple"` structure more closely mirrors traditional hierarchical factor analytic methods such as CFA; `"full"` structure more closely mirrors EFA methods Simple structure is the more conservative (established) approach and is therefore the default. Treat `"full"` as experimental as proper vetting and validation has not been established
`impute`	Character (length = 1). If there are any missing data, then imputation can be implemented. Available options: • `"none"` — Default. No imputation is performed • `"mean"` — The mean value of each variable is used to replace missing data for that variable • `"median"` — The median value of each variable is used to replace missing data for that variable
`corr`	Character (length = 1). Method to compute correlations. Defaults to `"auto"`. Available options: • `"auto"` — Automatically computes appropriate correlations for the data using Pearson's for continuous, polychoric for ordinal, tetrachoric for binary, and polyserial/biserial for ordinal/binary with continuous. To change the number of categories that are considered ordinal, use `ordinal.categories` (see `polychoric.matrix` for more details)

- "cor_auto" — Uses cor_auto to compute correlations. Arguments can be passed along to the function
- "pearson" — Pearson's correlation is computed for all variables regardless of categories
- "spearman" — Spearman's rank-order correlation is computed for all variables regardless of categories

For other similarity measures, compute them first and input them into data with the sample size (n)

na.data Character (length = 1). How should missing data be handled? Defaults to "pairwise". Available options:

- "pairwise" — Computes correlation for all available cases between two variables
- "listwise" — Computes correlation for all complete cases in the dataset

model Character (length = 1). Defaults to "glasso". Available options:

- "BGGM" — Computes the Bayesian Gaussian Graphical Model. Set argument ordinal.categories to determine levels allowed for a variable to be considered ordinal. See ?BGGM::estimate for more details
- "glasso" — Computes the GLASSO with EBIC model selection. See EBICglasso.qgraph for more details
- "TMFG" — Computes the TMFG method. See TMFG for more details

lower.algorithm

Character or cluster_* function (length = 1). Defaults to the lower order "louvain" with most common consensus clustering (1000 iterations; see community.consensus for more details)

Louvain with consensus clustering is *strongly* recommended. Using any other algorithm is considered *experimental* as they have not been designed to capture lower order communities

higher.algorithm

Character or cluster_* function (length = 1). Defaults to "louvain". Three options are listed below but all are available (see community.detection for other options):

- "leiden" — See cluster_leiden for more details
- "louvain" — By default, "louvain" will implement the higher-order (order = "higher") Louvain algorithm using the consensus clustering method (see community.consensus for more information). This function will implement consensus.method = "most_common" and consensus.iter = 1000 unless specified otherwise
- "walktrap" — See cluster_walktrap for more details

Using algorithm will set only higher.algorithm and lower.algorithm will default to Louvain with most common consensus clustering (1000 iterations)

uni.method Character (length = 1). What unidimensionality method should be used? Defaults to "louvain". Available options:

- "expand" — Expands the correlation matrix with four variables correlated 0.50. If number of dimension returns 2 or less in check, then the data are

unidimensional; otherwise, regular EGA with no matrix expansion is used. This method was used in the Golino et al.'s (2020) *Psychological Methods* simulation

- `"LE"` — Applies the Leading Eigenvector algorithm (`cluster_leading_eigen`) on the empirical correlation matrix. If the number of dimensions is 1, then the Leading Eigenvector solution is used; otherwise, regular EGA is used. This method was used in the Christensen et al.'s (2023) *Behavior Research Methods* simulation
- `"louvain"` — Applies the Louvain algorithm (`cluster_louvain`) on the empirical correlation matrix. If the number of dimensions is 1, then the Louvain solution is used; otherwise, regular EGA is used. This method was validated Christensen's (2022) *PsyArXiv* simulation. Consensus clustering can be used by specifying either `"consensus.method"` or `"consensus.iter"`

`plot.EGA` Boolean. If `TRUE`, returns a plot of the network and its estimated dimensions. Defaults to `TRUE`

`verbose` Boolean (length = 1). Whether messages and (insignificant) warnings should be output. Defaults to `FALSE` (silent calls). Set to `TRUE` to see all messages and warnings for every function call

`...` Additional arguments to be passed on to `auto.correlate`, `network.estimation`, `community.detection`, `community.consensus`, `EGA`, and `rotations`

Value

Returns a list of lists containing:

`lower_order` `EGA` results for the lower order structure

`higher_order` `EGA` results for the higher order structure

`parameters` A list containing `lower_loadings` and `lower_scores` that were used to estimate scores and the higher order `EGA` results, respectively

`dim.variables` A data frame with variable names and their lower and higher order assignments

`TEFI` Generalized TEFI using `tefi`

`plot.hierEGA` Plot output if `plot.EGA = TRUE`

Author(s)

Marcos Jiménez <marcosjnezhquez@gmailcom>, Francisco J. Abad <fjose.abad@uam.es>, Eduardo Garcia-Garzon <egarcia@ucjc.edu>, Hudson Golino <hfg9s@virginia.edu>, Alexander P. Christensen <alexpaulchristensen@gmail.com>, and Luis Eduardo Garrido <luisgarrido@pucmm.edu.do>

References

Hierarchical EGA simulation
Jiménez, M., Abad, F. J., Garcia-Garzon, E., Golino, H., Christensen, A. P., & Garrido, L. E. (2023). Dimensionality assessment in bifactor structures with multiple general factors: A network psychometrics approach. *Psychological Methods*.

3+ level hierarchical EGA
Samo, A., Christensen, A. P., Abad, F. J., Garrido, L. E., Garcia-Garzon, E., Golino, H. & McAbee,

S. T. (2023). Building the structure of personality from the bottom-up using Hierarchical Exploratory Graph Analysis. *PsyArXiv*.

Conceptual implementation
Golino, H., Thiyagarajan, J. A., Sadana, R., Teles, M., Christensen, A. P., & Boker, S. M. (2020). Investigating the broad domains of intrinsic capacity, functional ability and environment: An exploratory graph analysis approach for improving analytical methodologies for measuring healthy aging. *PsyArXiv*.

Revised network loadings
Christensen, A. P., Golino, H., Abad, F. J., & Garrido, L. E. (2024). Revised network loadings. *PsyArXiv*.

See Also

`plot.EGAnet` for plot usage in

Examples

```
# Example using network scores
opt.hier <- hierEGA(
  data = optimism, scores = "network",
  plot.EGA = FALSE # No plot for CRAN checks
)


# Plot multilevel plot
plot(opt.hier, plot.type = "multilevel")

# Plot multilevel plot with higher order
# border color matching the corresponding
# lower order color
plot(opt.hier, color.match = TRUE)

# Plot levels separately
plot(opt.hier, plot.type = "separate")
```

---

igraph2matrix *Convert network to matrix*

Description

Converts network to matrix

Usage

```
igraph2matrix(igraph_network, diagonal = 0)
```

Arguments

igraph_network	network object
diagonal	Numeric (length = 1). Value to be placed on the diagonal of network. Defaults to 0

Value

Returns a network in the format

Author(s)

Hudson Golino <hfg9s at virginia.edu> & Alexander P. Christensen <alexander.christensen at Vanderbilt.Edu>

Examples

```
# Convert network to {igraph}
igraph_network <- convert2igraph(ega.wmt$network)

# Convert network back to matrix
igraph2matrix(igraph_network)
```

---

infoCluster *Information Theoretic Mixture Clustering for* dynEGA

Description

Performs hierarchical clustering using Jensen-Shannon distance followed by the Louvain algorithm with consensus clustering. The method iteratively identifies smaller and smaller clusters until there is no change in the clusters identified

Usage

```
infoCluster(dynEGA.object, plot.cluster = TRUE, ...)
```

Arguments

dynEGA.object	A dynEGA or a dynEGA.ind.pop object that is used to match the arguments of the EII object
plot.cluster	Boolean (length = 1). Should plot of optimal and hierarchical clusters be output? Defaults to TRUE. Set to FALSE to not plot
...	Additional arguments to be passed on to jsd

Value

Returns a list containing:

clusters	A vector corresponding to cluster each participant belongs to
clusterTree	The dendogram from hclust the hierarhical clustering
clusterPlot	Plot output from results
JSD	Jensen-Shannon Distance

Author(s)

Hudson Golino <hfg9s at virginia.edu> & Alexander P. Christensen <alexander.christensen at Vanderbilt.Edu>

See Also

plot.EGAnet for plot usage in EGAnet

Examples

```
# Obtain data
sim.dynEGA <- sim.dynEGA # bypasses CRAN checks

## Not run:
# Dynamic EGA individual and population structure
dyn.ega1 <- dynEGA.ind.pop(
  data = sim.dynEGA, n.embed = 5, tau = 1,
  delta = 1, id = 25, use.derivatives = 1,
  ncores = 2, corr = "pearson"
)

# Perform information-theoretic clustering
clust1 <- infoCluster(dynEGA.object = dyn.ega1)
## End(Not run)
```

---

information *Information Theory Metrics*

Description

A general function to compute several different information theory metrics

Usage

```
information(
  data,
  base = 2.718282,
  bins = floor(sqrt(nrow(data)/5)),
 statistic = c("entropy", "joint.entropy", "conditional.entropy", "total.correlation",
    "dual.total.correlation", "o.information")
)
```

Arguments

`data`	Matrix or data frame. Should consist only of variables to be used in the analysis
`base`	Numeric (length = 1). Base of logarithm to use for entropy. Common options include: • `2` — bits • `2.718282` — nats • `10` — bans Defaults to `exp(1)` or `2.718282`
`bins`	Numeric (length = 1). Number of bins if data are not discrete. Defaults to `floor(sqrt(nrow(data) / 5))`
`statistic`	Character. Information theory statistics to compute. Available options: • `"entropy"` — Shannon's entropy (Shannon, 1948) for each variable in `data`. Values range from `0` to `log(k)` where `k` is the number of categories for the variable • `"joint.entropy"` — shared uncertainty over all variables in `data`. Values range from the maximum of the individual entropies to the sum of individual entropies • `"conditional.entropy"` — uncertainty remaining after considering all other variables in `data`. Values range from `0` to the individual entropy of the conditioned variable • `"total.correlation"` — generalization of mutual information to more than two variables (Watanabe, 1960). Quantifies the redundancy of information in `data`. Values range from `0` to the sum of individual entropies minus the maximum of the individual entropies • `"dual.total.correlation"` — "shared randomness" or total uncertainty remaining in the `data` (Han, 1978). Values range from `0` to joint entropy • `"o.information"` — quantifies the extent to which the `data` is represented by lower-order (`> 0`; redundancy) or higher-order (`< 0`; synergy) constraint (Crutchfield, 1994) By default, all statistics are computed

Value

Returns list containing *only* requested `statistic`

Author(s)

Hudson F. Golino <hfg9s at virginia.edu> and Alexander P. Christensen <alexpaulchristensen@gmail.com>

References

Shannon's entropy
Shannon, C. E. (1948). A mathematical theory of communication. *The Bell System Technical Journal*, *27*(3), 379-423.

Formalization of total correlation
Watanabe, S. (1960). Information theoretical analysis of multivariate correlation. *IBM Journal of Research and Development 4*, 66-82.

Applied implementation of total correlation
Felix, L. M., Mansur-Alves, M., Teles, M., Jamison, L., & Golino, H. (2021). Longitudinal impact and effects of booster sessions in a cognitive training program for healthy older adults. *Archives of Gerontology and Geriatrics*, *94*, 104337.

Formalization of dual total correlation
Te Sun, H. (1978). Nonnegative entropy measures of multivariate symmetric correlations. *Information and Control*, *36*, 133-156.

Formalization of O-information
Crutchfield, J. P. (1994). The calculi of emergence: Computation, dynamics and induction. *Physica D: Nonlinear Phenomena*, *75*(1-3), 11-54.

Applied implementation of O-information
Marinazzo, D., Van Roozendaal, J., Rosas, F. E., Stella, M., Comolatti, R., Colenbier, N., Stramaglia, S., & Rosseel, Y. (2024). An information-theoretic approach to build hypergraphs in psychometrics. *Behavior Research Methods*, 1-23.

Examples

```
# All measures
information(wmt2[,7:24])

# One measures
information(wmt2[,7:24], statistic = "joint.entropy")
```

---

intelligenceBattery *Intelligence Data*

Description

A response matrix (n = 1152) of the International Cognitive Ability Resource (ICAR) intelligence battery developed by Condon and Revelle (2016).

Usage

```
data(intelligenceBattery)
```

Format

A 1185x125 response matrix

Examples

```
data("intelligenceBattery")
```

`invariance` *Measurement Invariance of* `EGA` *Structure*

Description

Estimates configural invariance using `bootEGA` on all data (across groups) first. After configural variance is established, then metric invariance is tested using the community structure that established configural invariance (see **Details** for more information on this process)

Usage

```
invariance(
  data,
  groups,
  structure = NULL,
  iter = 500,
  configural.threshold = 0.7,
  configural.type = c("parametric", "resampling"),
  corr = c("auto", "cor_auto", "pearson", "spearman"),
  na.data = c("pairwise", "listwise"),
  model = c("BGGM", "glasso", "TMFG"),
  algorithm = c("leiden", "louvain", "walktrap"),
  uni.method = c("expand", "LE", "louvain"),
  ncores,
  seed = NULL,
  verbose = TRUE,
  ...
)
```

Arguments

`data`	Matrix or data frame. Should consist only of variables to be used in the analysis
`groups`	Numeric or character vector (length = `nrow(data)`). Group membership corresponding to each case in data
`structure`	Numeric or character vector (length = `ncol(data)`). A vector representing the structure (numbers or labels for each item). Can be theoretical factors or the structure detected by `EGA`. If supplied, then configural invariance check is skipped (i.e., configural invariance is assumed based on the given structure)

`iter` Numeric (length = 1). Number of iterations to perform for the permutation. Defaults to `500` (recommended)

`configural.threshold`
Numeric (length = 1). Value to use a threshold in `itemStability` to determine which items should be removed during configural invariance (see **Details** for more information). Defaults to `0.70` (recommended)

`configural.type`
Character (length = 1). Type of bootstrap to use for configural invariance in `bootEGA`. Defaults to `"parametric"`

`corr` Character (length = 1). Method to compute correlations. Defaults to `"auto"`. Available options:

- `"auto"` — Automatically computes appropriate correlations for the data using Pearson's for continuous, polychoric for ordinal, tetrachoric for binary, and polyserial/biserial for ordinal/binary with continuous. To change the number of categories that are considered ordinal, use `ordinal.categories` (see `polychoric.matrix` for more details)
- `"cor_auto"` — Uses `cor_auto` to compute correlations. Arguments can be passed along to the function
- `"pearson"` — Pearson's correlation is computed for all variables regardless of categories
- `"spearman"` — Spearman's rank-order correlation is computed for all variables regardless of categories

For other similarity measures, compute them first and input them into `data` with the sample size (`n`)

`na.data` Character (length = 1). How should missing data be handled? Defaults to `"pairwise"`. Available options:

- `"pairwise"` — Computes correlation for all available cases between two variables
- `"listwise"` — Computes correlation for all complete cases in the dataset

`model` Character (length = 1). Defaults to `"glasso"`. Available options:

- `"BGGM"` — Computes the Bayesian Gaussian Graphical Model. Set argument `ordinal.categories` to determine levels allowed for a variable to be considered ordinal. See `?BGGM::estimate` for more details
- `"glasso"` — Computes the GLASSO with EBIC model selection. See `EBICglasso.qgraph` for more details
- `"TMFG"` — Computes the TMFG method. See `TMFG` for more details

`algorithm` Character or `cluster_*` function (length = 1). Defaults to `"walktrap"`. Three options are listed below but all are available (see `community.detection` for other options):

- `"leiden"` — See `cluster_leiden` for more details
- `"louvain"` — By default, `"louvain"` will implement the Louvain algorithm using the consensus clustering method (see `community.consensus` for more information). This function will implement `consensus.method = "most_common"` and `consensus.iter = 1000` unless specified otherwise

- "walktrap" — See cluster_walktrap for more details

uni.method — Character (length = 1). What unidimensionality method should be used? Defaults to "louvain". Available options:

- "expand" — Expands the correlation matrix with four variables correlated 0.50. If number of dimension returns 2 or less in check, then the data are unidimensional; otherwise, regular EGA with no matrix expansion is used. This method was used in the Golino et al.'s (2020) *Psychological Methods* simulation
- "LE" — Applies the Leading Eigenvector algorithm (cluster_leading_eigen) on the empirical correlation matrix. If the number of dimensions is 1, then the Leading Eigenvector solution is used; otherwise, regular EGA is used. This method was used in the Christensen et al.'s (2023) *Behavior Research Methods* simulation
- "louvain" — Applies the Louvain algorithm (cluster_louvain) on the empirical correlation matrix. If the number of dimensions is 1, then the Louvain solution is used; otherwise, regular EGA is used. This method was validated Christensen's (2022) *PsyArXiv* simulation. Consensus clustering can be used by specifying either "consensus.method" or "consensus.iter"

ncores — Numeric (length = 1). Number of cores to use in computing results. Defaults to ceiling(parallel::detectCores() / 2) or half of your computer's processing power. Set to 1 to not use parallel computing

If you're unsure how many cores your computer has, then type: parallel::detectCores()

seed — Numeric (length = 1). Defaults to NULL or random results. Set for reproducible results. See Reproducibility and PRNG for more details on random number generation in

verbose — Boolean (length = 1). Should progress be displayed? Defaults to TRUE. Set to FALSE to not display progress

... — Additional arguments that can be passed on to auto.correlate, network.estimation, community.detection, community.consensus, EGA, bootEGA, and net.loads

Details

In traditional psychometrics, measurement invariance is performed in sequential testing from more flexible (more free parameters) to more rigid (fewer free parameters) structures. Measurement invariance in network psychometrics is no different.

Configural Invariance

To establish configural invariance, the data are collapsed across groups and a common sample structure is identified used bootEGA and itemStability. If some variables have a replication less than 0.70 in their assigned dimension, then they are considered unstable and therefore not invariant. These variables are removed and this process is repeated until all items are considered stable (replication values greater than 0.70) or there are no variables left. If configural invariance cannot be established, then the last run of results are returned and metric invariance is not tested (because configural invariance is not met). Importantly, if any variables *are* removed, then configural invariance is not met for the original structure. Any removal would suggest only partial configural invariance is met.

Metric Invariance

The variables that remain after configural invariance are submitted to metric invariance. First, each group estimates a network and then network loadings (`net.loads`) are computed using the assigned community memberships (determined during configural invariance). Then, the difference between the assigned loadings of the groups is computed. This difference represents the empirical values. Second, the group memberships are permutated and networks are estimated based on the these permutated groups for `iter` times. Then, network loadings are computed and the difference between the assigned loadings of the group is computed, resulting in a null distribution. The empirical difference is then compared against the null distribution using a two-tailed *p*-value based on the number of null distribution differences that are greater and less than the empirical differences for each variable. Both uncorrected and false discovery rate corrected *p*-values are returned in the results. Uncorrected *p*-values are flagged for significance along with the direction of group differences.

Three or More Groups

When there are 3 or more groups, the function performs metric invariance testing by comparing all possible pairs of groups. Specifically:

- *Pairwise Comparisons*: The function generates all possible unique group pairings and computes the differences in network loadings for each pair. The same community structure, derived from configural invariance or provided by the user, is used for all groups.
- *Permutation Testing*: For each group pair, permutation tests are conducted to assess the statistical significance of the observed differences in loadings. *p*-values are calculated based on the proportion of permuted differences that are greater than or equal to the observed difference.
- *Result Compilation*: The function compiles the results for each pair including both uncorrected (`p`) and FDR-corrected (Benjamini-Hochberg; `p_BH`) *p*-values, and the direction of differences. It returns a summary of the findings for all pairwise comparisons.

This approach allows for a detailed examination of metric invariance across multiple groups, ensuring that all potential differences are thoroughly assessed while maintaining the ability to identify specific group differences.

For more details, see Jamison, Golino, and Christensen (2023)

Value

Returns a list containing:

`configural.results`
: `bootEGA` results from the final run that produced configural invariance. This output will be output on the final run of unsuccessful configural invariance runs

`memberships`
: Original memberships provided in `structure` or from `EGA` if `structure = NULL`

`EGA`
: Original `EGA` results for the full sample

`groups`
: A list containing:
  - `EGA` — `EGA` results for each group
  - `loadings` — Network loadings (`net.loads`) for each group
  - `loadingsDifference` — Difference between the dominant loadings of each group

`permutation`
: A list containing:

- `groups` — Permutated groups acorss iterations
- `loadings` — Network loadings (`net.loads`) for each group for each permutation
- `loadingsDifference` — Difference between the dominant loadings of each group for each permutation

`results` Data frame of the results (which are printed)

Author(s)

Laura Jamison <lj5yn@virginia.edu>, Hudson F. Golino <hfg9s at virginia.edu>, and Alexander P. Christensen <alexpaulchristensen@gmail.com>,

References

Original implementation
Jamison, L., Christensen, A. P., & Golino, H. F. (2024). Metric invariance in exploratory graph analysis via permutation testing. *Methodology*, *20*(2), 144-186.

See Also

`plot.EGAnet` for plot usage in

Examples

```
# Load data
wmt <- wmt2[-1,7:24]

# Groups
groups <- rep(1:2, each = nrow(wmt) / 2)

## Not run:
# Measurement invariance
results <- invariance(wmt, groups, ncores = 2)

# Plot with uncorrected alpha = 0.05
plot(results, p_type = "p", p_value = 0.05)

# Plot with BH-corrected alpha = 0.10
plot(results, p_type = "p_BH", p_value = 0.10)
## End(Not run)
```

`itemStability` *Item Stability Statistics from* `bootEGA`

Description

Based on the bootEGA results, this function computes and plots the number of times an variable is estimated in the same dimension as originally estimated by an empirical EGA structure or a theoretical/input structure. The output also contains each variable's replication frequency (i.e., proportion of bootstraps that a variable appeared in each dimension

Usage

```
itemStability(bootega.obj, IS.plot = TRUE, structure = NULL, ...)
```

Arguments

bootega.obj	A bootEGA object
IS.plot	Boolean (length = 1). Should the plot be produced for item.replication? Defaults to TRUE
structure	Numeric (length = number of variables). A theoretical or pre-defined structure. Defaults to NULL or the empirical EGA result in the bootega.obj
...	Deprecated arguments from previous versions of itemStability

Value

Returns a list containing:

membership	A list containing: • empirical — A vector of the empirical memberships from the empirical EGA result • bootstrap — A matrix of the homogenized memberships from the replicate samples in the bootEGA results • structure — A vector of the structure used in the analysis. If structure = NULL, then this output will be the same as empirical
item.stability	A list containing: • empirical.dimensions — A vector of the proportion of times each item replicated within the structure defined by structure • all.dimensions — A matrix of the proportion of times each item replicated in each of the structure defined dimensions
plot	Plot output if IS.plot = TRUE

Author(s)

Hudson Golino <hfg9s at virginia.edu> and Alexander P. Christensen <alexpaulchristensen@gmail.com>

References

Original implementation of bootEGA
Christensen, A. P., & Golino, H. (2021). Estimating the stability of the number of factors via Bootstrap Exploratory Graph Analysis: A tutorial. *Psych*, *3*(3), 479-500.

Conceptual introduction
Christensen, A. P., Golino, H., & Silvia, P. J. (2020). A psychometric network perspective on the validity and validation of personality trait questionnaires. *European Journal of Personality*, *34*(6), 1095-1108.

See Also

`plot.EGAnet` for plot usage in `EGAnet`

Examples

```
# Load data
wmt <- wmt2[,7:24]

## Not run:
# Standard EGA example
boot.wmt <- bootEGA(
  data = wmt, iter = 500,
  type = "parametric", ncores = 2
)
## End(Not run)

# Standard item stability
wmt.is <- itemStability(boot.wmt)

## Not run:
# EGA fit example
boot.wmt.fit <- bootEGA(
  data = wmt, iter = 500,
  EGA.type = "EGA.fit",
  type = "parametric", ncores = 2
)

# EGA fit item stability
wmt.is.fit <- itemStability(boot.wmt.fit)

# Hierarchical EGA example
boot.wmt.hier <- bootEGA(
  data = wmt, iter = 500,
  EGA.type = "hierEGA",
  type = "parametric", ncores = 2
)

# Hierarchical EGA item stability
wmt.is.hier <- itemStability(boot.wmt.hier)

# Random-intercept EGA example
```

```
boot.wmt.ri <- bootEGA(
  data = wmt, iter = 500,
  EGA.type = "riEGA",
  type = "parametric", ncores = 2
)

# Random-intercept EGA item stability
wmt.is.ri <- itemStability(boot.wmt.ri)
## End(Not run)
```

`jsd` *Jensen-Shannon Distance*

Description

Computes the Jensen-Shannon Distance between two networks

Usage

```
jsd(network1, network2, method = c("kld", "spectral"), signed = TRUE)
```

Arguments

`network1`	Matrix or data frame. Network to be compared
`network2`	Matrix or data frame. Second network to be compared
`method`	Character (length = 1). Method to compute Jensen-Shannon Distance. Defaults to `"spectral"`. Available options: • `"kld"` — Uses Kullback-Leibler Divergence • `"spectral"` — Uses eigenvalues of combinatorial Laplacian matrix to compute Von Neumann entropy
`signed`	Boolean. (length = 1). Should networks be remain signed? Defaults to `TRUE`

Value

Returns Jensen-Shannon Distance

Author(s)

Hudson Golino <hfg9s at virginia.edu> & Alexander P. Christensen <alexander.christensen at Vanderbilt.Edu>

Examples

```
# Obtain wmt2 data
wmt <- wmt2[,7:24]

# Set seed (for reproducibility)
set.seed(1234)

# Split data
split1 <- sample(
  1:nrow(wmt), floor(nrow(wmt) / 2)
)
split2 <- setdiff(1:nrow(wmt), split1)

# Obtain split data
data1 <- wmt[split1,]
data2 <- wmt[split2,]

# Perform EBICglasso
glas1 <- EBICglasso.qgraph(data1)
glas2 <- EBICglasso.qgraph(data2)

# Spectral JSD
jsd(glas1, glas2)
# 0.1595893

# Spectral JSS (similarity)
1 - jsd(glas1, glas2)
# 0.8404107

# Jensen-Shannon Divergence
jsd(glas1, glas2, method = "kld")
# 0.1393621
```

LCT *Loadings Comparison Test*

Description

An algorithm to identify whether data were generated from a factor or network model using factor and network loadings. The algorithm uses heuristics based on theory and simulation. These heuristics were then submitted to several deep learning neural networks with 240,000 samples per model with varying parameters.

Usage

```
LCT(
  data,
  n = NULL,
```

```
  corr = c("auto", "cor_auto", "pearson", "spearman"),
  na.data = c("pairwise", "listwise"),
  model = c("BGGM", "glasso", "TMFG"),
  algorithm = c("leiden", "louvain", "walktrap"),
  uni.method = c("expand", "LE", "louvain"),
  iter = 100,
  seed = NULL,
  verbose = TRUE,
  ...
)
```

Arguments

data	Matrix or data frame. Should consist only of variables to be used in the analysis. Can be raw data or a correlation matrix
n	Numeric (length = 1). Sample size if data provided is a correlation matrix
corr	Character (length = 1). Method to compute correlations. Defaults to "auto". Available options: • "auto" — Automatically computes appropriate correlations for the data using Pearson's for continuous, polychoric for ordinal, tetrachoric for binary, and polyserial/biserial for ordinal/binary with continuous. To change the number of categories that are considered ordinal, use ordinal.categories (see polychoric.matrix for more details) • "cor_auto" — Uses cor_auto to compute correlations. Arguments can be passed along to the function • "pearson" — Pearson's correlation is computed for all variables regardless of categories • "spearman" — Spearman's rank-order correlation is computed for all variables regardless of categories For other similarity measures, compute them first and input them into data with the sample size (n)
na.data	Character (length = 1). How should missing data be handled? Defaults to "pairwise". Available options: • "pairwise" — Computes correlation for all available cases between two variables • "listwise" — Computes correlation for all complete cases in the dataset
model	Character (length = 1). Defaults to "glasso". Available options: • "BGGM" — Computes the Bayesian Gaussian Graphical Model. Set argument ordinal.categories to determine levels allowed for a variable to be considered ordinal. See ?BGGM::estimate for more details • "glasso" — Computes the GLASSO with EBIC model selection. See EBICglasso.qgraph for more details • "TMFG" — Computes the TMFG method. See TMFG for more details
algorithm	Character or igraph cluster_* function (length = 1). Defaults to "walktrap". Three options are listed below but all are available (see community.detection for other options):

- "leiden" — See cluster_leiden for more details
- "louvain" — By default, "louvain" will implement the Louvain algorithm using the consensus clustering method (see community.consensus for more information). This function will implement consensus.method = "most_common" and consensus.iter = 1000 unless specified otherwise
- "walktrap" — See cluster_walktrap for more details

uni.method — Character (length = 1). What unidimensionality method should be used? Defaults to "louvain". Available options:

- "expand" — Expands the correlation matrix with four variables correlated 0.50. If number of dimension returns 2 or less in check, then the data are unidimensional; otherwise, regular EGA with no matrix expansion is used. This method was used in the Golino et al.'s (2020) *Psychological Methods* simulation
- "LE" — Applies the Leading Eigenvector algorithm (cluster_leading_eigen) on the empirical correlation matrix. If the number of dimensions is 1, then the Leading Eigenvector solution is used; otherwise, regular EGA is used. This method was used in the Christensen et al.'s (2023) *Behavior Research Methods* simulation
- "louvain" — Applies the Louvain algorithm (cluster_louvain) on the empirical correlation matrix. If the number of dimensions is 1, then the Louvain solution is used; otherwise, regular EGA is used. This method was validated Christensen's (2022) *PsyArXiv* simulation. Consensus clustering can be used by specifying either "consensus.method" or "consensus.iter"

iter — Numeric (length = 1). Number of replicate samples to be drawn from a multivariate normal distribution (uses MASS::mvrnorm). Defaults to 100 (recommended)

seed — Numeric (length = 1). Defaults to NULL or random results. Set for reproducible results. See Reproducibility and PRNG for more details on random number generation in EGAnet

verbose — Boolean (length = 1). Should progress be displayed? Defaults to TRUE. Set to FALSE to not display progress

... — Additional arguments that can be passed on to auto.correlate, network.estimation, community.detection, community.consensus, and EGA

Value

Returns a list containing:

empirical — Prediction of model based on empirical dataset only

bootstrap — Prediction of model based on means of the loadings across the bootstrap replicate samples

proportion — Proportions of models suggested across bootstraps

Author(s)

Hudson F. Golino <hfg9s at virginia.edu> and Alexander P. Christensen <alexpaulchristensen at gmail.com>

References

Model training and validation
Christensen, A. P., & Golino, H. (2021). Factor or network model? Predictions from neural networks. *Journal of Behavioral Data Science*, *1*(1), 85-126.

Examples

```
# Get data
data <- psych::bfi[,1:25]

## Not run: # Compute LCT
## Factor model
LCT(data)
## End(Not run)
```

`modularity` *Computes the (Signed) Modularity Statistic*

Description

Computes (signed) modularity statistic given a network and community structure. Allows the resolution parameter to be set

Usage

```
modularity(network, memberships, resolution = 1, signed = FALSE)
```

Arguments

`network`	Matrix or data frame. A symmetric matrix representing a network
`memberships`	Numeric (length = `ncol(network)`). A numeric vector of integer values corresponding to each node's community membership
`resolution`	Numeric (length = 1). A parameter that adjusts modularity to prefer smaller (`resolution` > 1) or larger (0 < `resolution` < 1) communities. Defaults to 1 (standard modularity computation)
`signed`	Boolean (length = 1). Whether signed or absolute modularity should be computed. The most common modularity metric is defined by positive values only. Gomez et al. (2009) introduced a signed version of modularity that will discount modularity for edges with negative values. This property isn't always desired for psychometric networks. If `TRUE`, then this signed modularity metric will be computed. If `FALSE`, then the absolute value of the edges in the network (using `abs`) will be used to compute modularity. Defaults to `FALSE`

Value

Returns the modularity statistic

Author(s)

Alexander P. Christensen <alexpaulchristensen@gmail.com> with assistance from GPT-4

References

Gomez, S., Jensen, P., & Arenas, A. (2009). Analysis of community structure in networks of correlated data. *Physical Review E*, *80*(1), 016114.

Examples

```
# Load data
wmt <- wmt2[,7:24]

# Estimate EGA
ega.wmt <- EGA(wmt, model = "glasso")

# Compute standard (absolute values) modularity
modularity(
  network = ega.wmt$network,
  memberships = ega.wmt$wc,
  signed = FALSE
)
# 0.1697952

# Compute signed modularity
modularity(
  network = ega.wmt$network,
  memberships = ega.wmt$wc,
  signed = TRUE
)
# 0.1701946
```

---

net.loads *Network Loadings*

Description

Computes the between- and within-community `strength` of each variable for each community

Usage

```
net.loads(
  A,
  wc,
  loading.method = c("original", "revised"),
  scaling = 2,
  rotation = NULL,
  ...
)
```

Arguments

A	Network matrix, data frame, or EGA object
wc	Numeric or character vector (length = ncol(A)). A vector of community assignments. If input into A is an EGA object, then wc is automatically detected
loading.method	Character (length = 1). Sets network loading calculation based on implementation described in "original" (Christensen & Golino, 2021) or the "revised" (Christensen et al., 2024) implementation. Defaults to "revised"
scaling	Numeric (length = 1). Scaling factor for the magnitude of the "experimental" network loadings. Defaults to 2. 10 makes loadings roughly the size of factor loadings when correlations between factors are orthogonal
rotation	Character. A rotation to use to obtain a simpler structure. For a list of rotations, see rotations for options. Defaults to NULL or no rotation. By setting a rotation, scores estimation will be based on the rotated loadings rather than unrotated loadings
...	Additional arguments to pass on to rotations

Details

Simulation studies have demonstrated that a node's strength centrality is roughly equivalent to factor loadings (Christensen & Golino, 2021; Hallquist, Wright, & Molenaar, 2019). Hallquist and colleagues (2019) found that node strength represented a combination of dominant and cross-factor loadings. This function computes each node's strength within each specified dimension, providing a rough equivalent to factor loadings (including cross-loadings; Christensen & Golino, 2021).

Value

Returns a list containing:

unstd	A matrix of the unstandardized within- and between-community strength values for each node
std	A matrix of the standardized within- and between-community strength values for each node
rotated	NULL if rotation = NULL; otherwise, a list containing the rotated standardized network loadings (loadings) and correlations between dimensions (Phi) from the rotation

Author(s)

Alexander P. Christensen <alexpaulchristensen@gmail.com> and Hudson Golino <hfg9s at virginia.edu>

References

Original implementation and simulation
Christensen, A. P., & Golino, H. (2021). On the equivalency of factor and network loadings. *Behavior Research Methods*, *53*, 1563-1580.

Demonstration of node strength similarity to CFA loadings
Hallquist, M., Wright, A. C. G., & Molenaar, P. C. M. (2019). Problems with centrality measures in psychopathology symptom networks: Why network psychometrics cannot escape psychometric theory. *Multivariate Behavioral Research*, 1-25.

Revised network loadings
Christensen, A. P., Golino, H., Abad, F. J., & Garrido, L. E. (2024). Revised network loadings. *PsyArXiv*.

Examples

```
# Load data
wmt <- wmt2[,7:24]

# Estimate EGA
ega.wmt <- EGA(
  data = wmt,
  plot.EGA = FALSE # No plot for CRAN checks
)

# Network loadings
net.loads(ega.wmt)
```

`net.scores` *Network Scores*

Description

This function computes network scores computed based on each node's `strength` within each community in the network (see `net.loads`). These values are used as "network loadings" for the weights of each variable.

Network scores are computed as a formative composite rather than a reflective factor. This composite representation is consistent with no latent factors that psychometric network theory proposes.

Scores can be computed as a "simple" structure, which is equivalent to a weighted sum scores or as a "full" structure, which is equivalent to an EFA approach. Conservatively, the "simple" structure approach is recommended until further validation

Usage

```
net.scores(
  data,
  A,
  wc,
  loading.method = c("original", "revised"),
  rotation = NULL,
 scores = c("Anderson", "Bartlett", "components", "Harman", "network", "tenBerge",
    "Thurstone"),
```

```
  loading.structure = c("simple", "full"),
  impute = c("mean", "median", "none"),
  ...
)
```

Arguments

`data`	Matrix or data frame. Should consist only of variables to be used in the analysis
`A`	Network matrix, data frame, or `EGA` object
`wc`	Numeric or character vector (length = `ncol(A)`). A vector of community assignments. If input into `A` is an `EGA` object, then `wc` is automatically detected
`loading.method`	Character (length = 1). Sets network loading calculation based on implementation described in `"original"` (Christensen & Golino, 2021) or the `"revised"` (Christensen et al., 2024) implementation. Defaults to `"revised"`
`rotation`	Character. A rotation to use to obtain a simpler structure. For a list of rotations, see `rotations` for options. Defaults to `NULL` or no rotation. By setting a rotation, `scores` estimation will be based on the rotated loadings rather than unrotated loadings
`scores`	Character (length = 1). How should scores be estimated? Defaults to `"network"` for network scores. Set to other scoring methods which will be computed using `factor.scores` (see link for arguments and explanations for other methods)
`loading.structure`	Character (length = 1). Whether simple structure or the saturated loading matrix should be used when computing scores. Defaults to `"simple"` `"simple"` structure more closely mirrors sum scores and CFA; `"full"` structure more closely mirrors EFA Simple structure is the more "conservative" (established) approach and is therefore the default. Treat `"full"` as experimental as proper vetting and validation has not been established
`impute`	Character (length = 1). If there are any missing data, then imputation can be implemented. Available options: • `"none"` — Default. No imputation is performed • `"mean"` — The mean value of each variable is used to replace missing data for that variable • `"median"` — The median value of each variable is used to replace missing data for that variable
`...`	Additional arguments to be passed on to `net.loads` and `factor.scores`

Value

Returns a list containing:

`scores`	A list containing the standardized (`std.scores`) rotated (`rot.scores`) scores. If `rotation` = `NULL`, then `rot.scores` will be `NULL`
`loadings`	Output from `net.loads`

Author(s)

Alexander P. Christensen <alexpaulchristensen@gmail.com> and Hudson F. Golino <hfg9s at virginia.edu>

References

Original implementation and simulation for loadings
Christensen, A. P., & Golino, H. (2021). On the equivalency of factor and network loadings. *Behavior Research Methods*, *53*, 1563-1580.

Preliminary simulation for scores
Golino, H., Christensen, A. P., Moulder, R., Kim, S., & Boker, S. M. (2021). Modeling latent topics in social media using Dynamic Exploratory Graph Analysis: The case of the right-wing and left-wing trolls in the 2016 US elections. *Psychometrika*.

Revised network loadings
Christensen, A. P., Golino, H., Abad, F. J., & Garrido, L. E. (2024). Revised network loadings. *PsyArXiv*.

Examples

```
# Load data
wmt <- wmt2[,7:24]

# Estimate EGA
ega.wmt <- EGA(
  data = wmt,
  plot.EGA = FALSE # No plot for CRAN checks
)

# Network scores
net.scores(data = wmt, A = ega.wmt)
```

---

network.compare *Compares Network Structures Using Permutation*

Description

A permutation implementation to determine statistical significance of whether the network structures are different from one another

Usage

```
network.compare(
  base,
  comparison,
  corr = c("auto", "cor_auto", "pearson", "spearman"),
  na.data = c("pairwise", "listwise"),
```

```
  model = c("BGGM", "glasso", "TMFG"),
  iter = 1000,
  ncores,
  verbose = TRUE,
  seed = NULL,
  ...
)
```

Arguments

base	Matrix or data frame. Should consist only of variables to be used in the analysis. First dataset
comparison	Matrix or data frame. Should consist only of variables to be used in the analysis. Second dataset
corr	Character (length = 1). Method to compute correlations. Defaults to "auto". Available options: • "auto" — Automatically computes appropriate correlations for the data using Pearson's for continuous, polychoric for ordinal, tetrachoric for binary, and polyserial/biserial for ordinal/binary with continuous. To change the number of categories that are considered ordinal, use ordinal.categories (see polychoric.matrix for more details) • "cor_auto" — Uses cor_auto to compute correlations. Arguments can be passed along to the function • "pearson" — Pearson's correlation is computed for all variables regardless of categories • "spearman" — Spearman's rank-order correlation is computed for all variables regardless of categories For other similarity measures, compute them first and input them into data with the sample size (n)
na.data	Character (length = 1). How should missing data be handled? Defaults to "pairwise". Available options: • "pairwise" — Computes correlation for all available cases between two variables • "listwise" — Computes correlation for all complete cases in the dataset
model	Character (length = 1). Defaults to "glasso". Available options: • "BGGM" — Computes the Bayesian Gaussian Graphical Model. Set argument ordinal.categories to determine levels allowed for a variable to be considered ordinal. See ?BGGM::estimate for more details • "glasso" — Computes the GLASSO with EBIC model selection. See EBICglasso.qgraph for more details • "TMFG" — Computes the TMFG method. See TMFG for more details
iter	Numeric (length = 1). Number of permutations to perform. Defaults to 1000 (recommended)

ncores	Numeric (length = 1). Number of cores to use in computing results. Defaults to `ceiling(parallel::detectCores() / 2)` or half of your computer's processing power. Set to 1 to not use parallel computing
verbose	Boolean (length = 1). Should progress be displayed? Defaults to TRUE. Set to FALSE to not display progress
seed	Numeric (length = 1). Defaults to NULL or random results. Set for reproducible results. See Reproducibility and PRNG for more details on random number generation in EGAnet
...	Additional arguments that can be passed on to auto.correlate, network.estimation, community.detection, community.consensus, EGA, and jsd

Value

Returns a list:

network	Data frame with row names of each measure, empirical value (statistic), and *p*-value based on the permutation test (p.value)
edges	List containing matrices of values for empirical values (statistic), *p*-values (p.value), and Benjamini-Hochberg corrected *p*-values (p.adjusted)

Author(s)

Hudson Golino <hfg9s at virginia.edu> and Alexander P. Christensen <alexpaulchristensen@gmail.com>

References

Frobenius Norm
Ulitzsch, E., Khanna, S., Rhemtulla, M., & Domingue, B. W. (2023). A graph theory based similarity metric enables comparison of subpopulation psychometric networks. *Psychological Methods*.

Jensen-Shannon Similarity (1 - Distance)
De Domenico, M., Nicosia, V., Arenas, A., & Latora, V. (2015). Structural reducibility of multilayer networks. *Nature Communications*, *6*(1), 1–9.

Total Network Strength
van Borkulo, C. D., van Bork, R., Boschloo, L., Kossakowski, J. J., Tio, P., Schoevers, R. A., Borsboom, D., & Waldorp, L. J. (2023). Comparing network structures on three aspects: A permutation test. *Psychological Methods*, *28*(6), 1273–1285.

Examples

```
# Load data
wmt <- wmt2[,7:24]

# Set groups (if necessary)
groups <- rep(1:2, each = nrow(wmt) / 2)

# Groups
group1 <- wmt[groups == 1,]
group2 <- wmt[groups == 2,]
```

```
## Not run: # Perform comparison
results <- network.compare(group1, group2)

# Print results
print(results)

# Plot edge differences
plot(results)
## End(Not run)
```

`network.estimation` *Apply a Network Estimation Method*

Description

General function to apply network estimation methods in `EGAnet`

Usage

```
network.estimation(
  data,
  n = NULL,
  corr = c("auto", "cor_auto", "cosine", "pearson", "spearman"),
  na.data = c("pairwise", "listwise"),
  model = c("BGGM", "glasso", "TMFG"),
  network.only = TRUE,
  verbose = FALSE,
  ...
)
```

Arguments

`data`	Matrix or data frame. Should consist only of variables to be used in the analysis
`n`	Numeric (length = 1). Sample size if `data` provided is a correlation matrix
`corr`	Character (length = 1). Method to compute correlations. Defaults to `"auto"`. Available options: • `"auto"` — Automatically computes appropriate correlations for the data using Pearson's for continuous, polychoric for ordinal, tetrachoric for binary, and polyserial/biserial for ordinal/binary with continuous. To change the number of categories that are considered ordinal, use `ordinal.categories` (see `polychoric.matrix` for more details) • `"cor_auto"` — Uses `cor_auto` to compute correlations. Arguments can be passed along to the function • `"cosine"` — Uses `cosine` to compute cosine similarity

- `"pearson"` — Pearson's correlation is computed for all variables regardless of categories
- `"spearman"` — Spearman's rank-order correlation is computed for all variables regardless of categories

For other similarity measures, compute them first and input them into `data` with the sample size (`n`)

`na.data` Character (length = 1). How should missing data be handled? Defaults to `"pairwise"`. Available options:

- `"pairwise"` — Computes correlation for all available cases between two variables
- `"listwise"` — Computes correlation for all complete cases in the dataset

`model` Character (length = 1). Defaults to `"glasso"`. Available options:

- `"BGGM"` — Computes the Bayesian Gaussian Graphical Model. Set argument `ordinal.categories` to determine levels allowed for a variable to be considered ordinal. See `?BGGM::estimate` for more details
- `"glasso"` — Computes the GLASSO with EBIC model selection. See `EBICglasso.qgraph` for more details
- `"TMFG"` — Computes the TMFG method. See `TMFG` for more details

`network.only` Boolean (length = 1). Whether the network only should be output. Defaults to `TRUE`. Set to `FALSE` to obtain all output for the network estimation method

`verbose` Boolean (length = 1). Whether messages and (insignificant) warnings should be output. Defaults to `FALSE` (silent calls). Set to `TRUE` to see all messages and warnings for every function call

`...` Additional arguments to be passed on to `auto.correlate` and the different network estimation methods (see `model` for model specific details)

Value

Returns a matrix populated with a network from the input data

Author(s)

Hudson Golino <hfg9s at virginia.edu> and Alexander P. Christensen <alexpaulchristensen@gmail.com>

References

Graphical Least Absolute Shrinkage and Selection Operator (GLASSO)
Friedman, J., Hastie, T., & Tibshirani, R. (2008). Sparse inverse covariance estimation with the graphical lasso. *Biostatistics*, *9*(3), 432–441.

GLASSO with Extended Bayesian Information Criterion (EBICglasso)
Epskamp, S., & Fried, E. I. (2018). A tutorial on regularized partial correlation networks. *Psychological Methods*, *23*(4), 617–634.

Bayesian Gaussian Graphical Model (BGGM)
Williams, D. R. (2021). Bayesian estimation for Gaussian graphical models: Structure learning, predictability, and network comparisons. *Multivariate Behavioral Research*, *56*(2), 336–352.

Triangulated Maximally Filtered Graph (TMFG)
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

```
# Load data
wmt <- wmt2[,7:24]

# EBICglasso (default for EGA functions)
glasso_network <- network.estimation(
  data = wmt, model = "glasso"
)

# TMFG
tmfg_network <- network.estimation(
  data = wmt, model = "TMFG"
)
```

---

network.predictability

Predict New Data based on Network

Description

General function to compute a network's predictive power on new data, following Haslbeck and Waldorp (2018) and Williams and Rodriguez (2022)

This implementation is different from the `predictability` in the `mgm` package (Haslbeck), which is based on (regularized) regression. This implementation uses the network directly, converting the partial correlations into an implied precision (inverse covariance) matrix. See **Details** for more information

Usage

```
network.predictability(network, original.data, newdata, ordinal.categories = 7)
```

Arguments

`network`	Matrix or data frame. A partial correlation network
`original.data`	Matrix or data frame. Must consist only of variables to be used to estimate the `network`. See **Examples**
`newdata`	Matrix or data frame. Must consist of the same variables in the same order as `original.data`. See **Examples**
`ordinal.categories`	Numeric (length = 1). *Up to* the number of categories *before* a variable is considered continuous. Defaults to `7` categories before `8` is considered continuous

Details

This implementation of network predictability proceeds in several steps with important assumptions:

1. Network was estimated using (partial) correlations (not regression like the `mgm` package!)

2. Original data that was used to estimate the network in 1. is necessary to apply the original scaling to the new data

3. (Linear) regression-like coefficients are obtained by reserve engineering the inverse covariance matrix using the network's partial correlations (i.e., by setting the diagonal of the network to -1 and computing the inverse of the opposite signed partial correlation matrix; see `EGAnet:::pcor2inv`)

4. Predicted values are obtained by matrix multiplying the new data with these coefficients

5. **Dichotomous and polytomous** data are given categorical values based on the **original data's** thresholds and these thresholds are used to convert the continuous predicted values into their corresponding categorical values

6. Evaluation metrics:

- dichotomous — `"Accuracy"` or the percent correctly predicted for the 0s and 1s and `"Kappa"` or Cohen's Kappa (see cite)
- polytomous — `"Linear Kappa"` or linearly weighted Kappa and `"Krippendorff's alpha"` (see cite)
- continuous — R-squared (`"R2"`) and root mean square error (`"RMSE"`)

Value

Returns a list containing:

`predictions`	Predicted values of `newdata` based on the `network`
`betas`	Beta coefficients derived from the `network`
`results`	Performance metrics for each variable in `newdata`

Author(s)

Hudson Golino <hfg9s at virginia.edu> and Alexander P. Christensen <alexpaulchristensen@gmail.com>

References

Original Implementation of Node Predictability
Haslbeck, J. M., & Waldorp, L. J. (2018). How well do network models predict observations? On the importance of predictability in network models. *Behavior Research Methods*, *50*(2), 853–861.

Derivation of Regression Coefficients Used (Formula 3)
Williams, D. R., & Rodriguez, J. E. (2022). Why overfitting is not (usually) a problem in partial correlation networks. *Psychological Methods*, *27*(5), 822–840.

Cohen's Kappa
Cohen, J. (1960). A coefficient of agreement for nominal scales. *Educational and Psychological Measurement*, *20*(1), 37-46.

Cohen, J. (1968). Weighted kappa: nominal scale agreement provision for scaled disagreement or partial credit. *Psychological Bulletin*, *70*(4), 213-220.

Krippendorff's alpha

Krippendorff, K. (2013). Content analysis: An introduction to its methodology (3rd ed.). Thousand Oaks, CA: Sage.

Examples

```
# Load data
wmt <- wmt2[,7:24]

# Set seed (to reproduce results)
set.seed(42)

# Split data
training <- sample(
  1:nrow(wmt), round(nrow(wmt) * 0.80) # 80/20 split
)

# Set splits
wmt_train <- wmt[training,]
wmt_test <- wmt[-training,]

# EBICglasso (default for EGA functions)
glasso_network <- network.estimation(
  data = wmt_train, model = "glasso"
)

# Check predictability
network.predictability(
  network = glasso_network, original.data = wmt_train,
  newdata = wmt_test
)
```

---

optimism *Optimism Data*

Description

A response matrix (n = 282) containing responses to 10 items of the Revised Life Orientation Test (LOT-R), developed by Scheier, Carver, & Bridges (1994).

Usage

```
data(optimism)
```

Format

A 282x10 response matrix

References

Scheier, M. F., Carver, C. S., & Bridges, M. W. (1994). Distinguishing optimism from neuroticism (and trait anxiety, self-mastery, and self-esteem): a reevaluation of the Life Orientation Test. *Journal of Personality and Social Psychology*, *67*, 1063-1078.

Examples

```
data("optimism")
```

`polychoric.matrix` *Computes Polychoric Correlations*

Description

A fast implementation of polychoric correlations in C. Uses the Beasley-Springer-Moro algorithm (Boro & Springer, 1977; Moro, 1995) to estimate the inverse univariate normal CDF, the Drezner-Wesolosky approximation (Drezner & Wesolosky, 1990) to estimate the bivariate normal CDF, and Brent's method (Brent, 2013) for optimization of rho

Usage

```
polychoric.matrix(
  data,
  na.data = c("pairwise", "listwise"),
  empty.method = c("none", "zero", "all"),
  empty.value = c("none", "point_five", "one_over"),
  ...
)
```

Arguments

`data`	Matrix or data frame. A dataset with all ordinal values (rows = cases, columns = variables). Data are required to be between `0` and 11. Proper adjustments should be made prior to analysis (e.g., scales from -3 to 3 in increments of 1 should be shifted by added 4 to all values)
`na.data`	Character (length = 1). How should missing data be handled? Defaults to `"pairwise"`. Available options: • `"pairwise"` — Computes correlation for all available cases between two variables • `"listwise"` — Computes correlation for all complete cases in the dataset
`empty.method`	Character (length = 1). Method for empty cell correction. Available options: • `"none"` — Adds no value (`empty.value = "none"`) to the empirical joint frequency table between two variables • `"zero"` — Adds `empty.value` to the cells with zero in the joint frequency table between two variables

- `"all"` — Adds `empty.value` to all in the joint frequency table between two variables

`empty.value` Character (length = 1). Value to add to the joint frequency table cells. Accepts numeric values between 0 and 1 or specific methods:

- `"none"` — Adds no value (`0`) to the empirical joint frequency table between two variables
- `"point_five"` — Adds `0.5` to the cells defined by `empty.method`
- `"one_over"` — Adds `1 / n` where *n* equals the number of cells based on `empty.method`. For `empty.method = "zero"`, *n* equals the number of *zero* cells

`...` Not used but made available for easier argument passing

Value

Returns a polychoric correlation matrix

Author(s)

Alexander P. Christensen <alexpaulchristensen@gmail.com> with assistance from GPT-4

References

Beasley-Moro-Springer algorithm
Beasley, J. D., & Springer, S. G. (1977). Algorithm AS 111: The percentage points of the normal distribution. *Journal of the Royal Statistical Society. Series C (Applied Statistics)*, *26*(1), 118-121.

Moro, B. (1995). The full monte. *Risk 8 (February)*, 57-58.

Brent optimization
Brent, R. P. (2013). Algorithms for minimization without derivatives. Mineola, NY: Dover Publications, Inc.

Drezner-Wesolowsky bivariate normal approximation
Drezner, Z., & Wesolowsky, G. O. (1990). On the computation of the bivariate normal integral. *Journal of Statistical Computation and Simulation*, *35*(1-2), 101-107.

Examples

```
# Load data (ensure matrix for missing data example)
wmt <- as.matrix(wmt2[,7:24])

# Compute polychoric correlation matrix
correlations <- polychoric.matrix(wmt)

# Randomly assign missing data
wmt[sample(1:length(wmt), 1000)] <- NA

# Compute polychoric correlation matrix
# with pairwise missing
pairwise_correlations <- polychoric.matrix(
  wmt, na.data = "pairwise"
```

```
)

# Compute polychoric correlation matrix
# with listwise missing
pairwise_correlations <- polychoric.matrix(
  wmt, na.data = "listwise"
)
```

`prime.num` *Prime Numbers through 100,000*

Description

Numeric vector of primes generated from the primes package. Used in the function `[EGAnet]{ergoInfo}`. Not for general use

Usage

```
data(prime.num)
```

Format

A 1185x24 response matrix

Examples

```
data("prime.num")
```

`riEGA` *Random-Intercept* EGA

Description

Estimates the number of substantive dimensions after controlling for wording effects. EGA is applied to a residual correlation matrix after subtracting and random intercept factor with equal unstandardized loadings from all the regular and unrecoded reversed items in the database

Usage

```
riEGA(
  data,
  n = NULL,
  corr = c("auto", "cor_auto", "cosine", "pearson", "spearman"),
  na.data = c("pairwise", "listwise"),
  model = c("glasso", "TMFG"),
  algorithm = c("leiden", "louvain", "walktrap"),
  uni.method = c("expand", "LE", "louvain"),
  plot.EGA = TRUE,
  verbose = FALSE,
  ...
)
```

Arguments

`data`	Matrix or data frame. Should consist only of variables to be used in the analysis. **Must** be raw data and not a correlation matrix
`n`	Numeric (length = 1). Sample size if `data` provided is a correlation matrix
`corr`	Character (length = 1). Method to compute correlations. Defaults to `"auto"`. Available options: • `"auto"` — Automatically computes appropriate correlations for the data using Pearson's for continuous, polychoric for ordinal, tetrachoric for binary, and polyserial/biserial for ordinal/binary with continuous. To change the number of categories that are considered ordinal, use `ordinal.categories` (see `polychoric.matrix` for more details) • `"cor_auto"` — Uses `cor_auto` to compute correlations. Arguments can be passed along to the function • `"cosine"` — Uses `cosine` to compute cosine similarity • `"pearson"` — Pearson's correlation is computed for all variables regardless of categories • `"spearman"` — Spearman's rank-order correlation is computed for all variables regardless of categories For other similarity measures, compute them first and input them into `data` with the sample size (`n`)
`na.data`	Character (length = 1). How should missing data be handled? Defaults to `"pairwise"`. Available options: • `"pairwise"` — Computes correlation for all available cases between two variables • `"listwise"` — Computes correlation for all complete cases in the dataset
`model`	Character (length = 1). Defaults to `"glasso"`. Available options: • `"BGGM"` — Computes the Bayesian Gaussian Graphical Model. Set argument `ordinal.categories` to determine levels allowed for a variable to be considered ordinal. See `?BGGM::estimate` for more details

- "glasso" — Computes the GLASSO with EBIC model selection. See EBICglasso.qgraph for more details
- "TMFG" — Computes the TMFG method. See TMFG for more details

algorithm — Character or igraph cluster_* function (length = 1). Defaults to "walktrap". Three options are listed below but all are available (see community.detection for other options):

- "leiden" — See cluster_leiden for more details
- "louvain" — By default, "louvain" will implement the Louvain algorithm using the consensus clustering method (see community.consensus for more information). This function will implement consensus.method = "most_common" and consensus.iter = 1000 unless specified otherwise
- "walktrap" — See cluster_walktrap for more details

uni.method — Character (length = 1). What unidimensionality method should be used? Defaults to "louvain". Available options:

- "expand" — Expands the correlation matrix with four variables correlated 0.50. If number of dimension returns 2 or less in check, then the data are unidimensional; otherwise, regular EGA with no matrix expansion is used. This method was used in the Golino et al.'s (2020) *Psychological Methods* simulation
- "LE" — Applies the Leading Eigenvector algorithm (cluster_leading_eigen) on the empirical correlation matrix. If the number of dimensions is 1, then the Leading Eigenvector solution is used; otherwise, regular EGA is used. This method was used in the Christensen et al.'s (2023) *Behavior Research Methods* simulation
- "louvain" — Applies the Louvain algorithm (cluster_louvain) on the empirical correlation matrix. If the number of dimensions is 1, then the Louvain solution is used; otherwise, regular EGA is used. This method was validated Christensen's (2022) *PsyArXiv* simulation. Consensus clustering can be used by specifying either "consensus.method" or "consensus.iter"

plot.EGA — Boolean (length = 1). If TRUE, returns a plot of the network and its estimated dimensions. Defaults to TRUE

verbose — Boolean (length = 1). Whether messages and (insignificant) warnings should be output. Defaults to FALSE (silent calls). Set to TRUE to see all messages and warnings for every function call

... — Additional arguments to be passed on to auto.correlate, network.estimation, community.detection, community.consensus, and EGA

Value

Returns a list containing:

EGA — Results from EGA

RI — A list containing information about the random-intercept model (if the model converged):

- fit — The fit object for the random-intercept model using cfa

- `lavaan.args` — The arguments used in `cfa`
- `loadings` — Standardized loadings from the random-intercept model
- `correlation` — Residual correlations after accounting for the random-intercept model

`TEFI`	`link[EGAnet]{tefi}` for the estimated structure
`plot.EGA`	Plot output if `plot.EGA = TRUE`

Author(s)

Alejandro Garcia-Pardina <alejandrogp97@gmail.com>, Francisco J. Abad <fjose.abad@uam.es>, Alexander P. Christensen <alexpaulchristensen@gmail.com>, Hudson Golino <hfg9s at virginia.edu>, Luis Eduardo Garrido <luisgarrido@pucmm.edu.do>, and Robert Moulder <rgm4fd@virginia.edu>

References

Selection of CFA Estimator
Rhemtulla, M., Brosseau-Liard, P. E., & Savalei, V. (2012). When can categorical variables be treated as continuous? A comparison of robust continuous and categorical SEM estimation methods under suboptimal conditions. *Psychological Methods*, *17*, 354-373.

See Also

`plot.EGAnet` for plot usage in `EGAnet`

Examples

```
# Obtain example data
wmt <- wmt2[,7:24]

# riEGA example
riEGA(data = wmt, plot.EGA = FALSE)
# no plot for CRAN checks
```

`sim.dynEGA` *sim.dynEGA Data*

Description

A simulated (multivariate time series) data with 24 variables, 100 individual observations, 50 time points per individual and 2 groups of individuals

Usage

```
data(sim.dynEGA)
```

Format

A 5000 x 26 multivariate time series

Details

Data were generated using the simDFM function with the following arguments:

Group 1

```
simDFM( variab = 12, timep = 50, nfact = 2, error = 0.125, dfm = "DAFS", loadings = EGAnet:::runif_xoshiro(
1, min = 0.50, max = 0.70 ), autoreg = 0.80, crossreg = 0.00, var.shock = 0.36, cov.shock
= 0.18 )
```

Group 2

```
simDFM( variab = 8, timep = 50, nfact = 3, error = 0.125, dfm = "DAFS", loadings = EGAnet:::runif_xoshiro(
1, min = 0.50, max = 0.70 ), autoreg = 0.80, crossreg = 0.00, var.shock = 0.36, cov.shock
= 0.18 )
```

Examples

```
data("sim.dynEGA")
```

---

simDFM *Simulate data following a Dynamic Factor Model*

Description

Function to simulate data following a dynamic factor model (DFM). Two DFMs are currently available: the direct autoregressive factor score model (Engle & Watson, 1981; Nesselroade, McArdle, Aggen, and Meyers, 2002) and the dynamic factor model with random walk factor scores.

Usage

```
simDFM(
  variab,
  timep,
  nfact,
  error,
  dfm = c("DAFS", "RandomWalk"),
  loadings,
  autoreg,
  crossreg,
  var.shock,
  cov.shock,
  burnin = 1000
)
```

Arguments

variab	Number of variables per factor.
timep	Number of time points.
nfact	Number of factors.
error	Value to be used to construct a diagonal matrix Q. This matrix is p x p covariance matrix Q that will generate random errors following a multivariate normal distribution with mean zeros. The value provided is squared before constructing Q.
dfm	A string indicating the dynamical factor model to use. Current options are: • DAFS — Simulates data using the direct autoregressive factor score model. This is the default method • RandomWalk — Simulates data using a dynamic factor model with random walk factor scores
loadings	Magnitude of the loadings.
autoreg	Magnitude of the autoregression coefficients.
crossreg	Magnitude of the cross-regression coefficients.
var.shock	Magnitude of the random shock variance.
cov.shock	Magnitude of the random shock covariance
burnin	Number of n first samples to discard when computing the factor scores. Defaults to 1000.

Author(s)

Hudson F. Golino <hfg9s at virginia.edu>

References

Engle, R., & Watson, M. (1981). A one-factor multivariate time series model of metropolitan wage rates. *Journal of the American Statistical Association*, *76*(376), 774-781.

Nesselroade, J. R., McArdle, J. J., Aggen, S. H., & Meyers, J. M. (2002). Dynamic factor analysis models for representing process in multivariate time-series. In D. S. Moskowitz & S. L. Hershberger (Eds.), *Multivariate applications book series. Modeling intraindividual variability with repeated measures data: Methods and applications*, 235-265.

Examples

```
## Not run:
# Estimate EGA network
data1 <- simDFM(variab = 5, timep = 50, nfact = 3, error = 0.05,
dfm = "DAFS", loadings = 0.7, autoreg = 0.8,
crossreg = 0.1, var.shock = 0.36,
cov.shock = 0.18, burnin = 1000)
## End(Not run)
```

simEGM *Simulate data following a Exploratory Graph Model (EGM)*

Description

Function to simulate data based on EGM

Usage

```
simEGM(
  communities,
  variables,
  loadings,
  cross.loadings = 0.01,
  correlations,
  sample.size,
  p.in = 0.95,
  p.out = 0.8,
  max.iterations = 1000
)
```

Arguments

communities	Numeric (length = 1). Number of communities to generate
variables	Numeric vector (length = 1 or communities). Number of variables per community
loadings	Numeric (length = 1). Magnitude of the assigned network loadings. Uses the same magnitude as factors loadings Uses runif(n, min = value - 0.025, max = value + 0.025) for some jitter in the loadings
cross.loadings	Numeric (length = 1). Standard deviation of a normal distribution with a mean of zero (n, mean = 0, sd = value). Defaults to 0.01
correlations	Numeric (length = 1). Magnitude of the community correlations Uses runif(n, min = value - 0.015, max = value + 0.015) for some jitter in the correlations
sample.size	Numeric (length = 1). Number of observations to generate
p.in	Numeric (length = 1). Sets the probability of retaining an edge *within* communities. Single values are applied to all communities. Defaults to 0.95
p.out	Numeric (length = 1 or communities). Sets the probability of retaining an edge *between* communities. Single values are applied to all communities. Defaults to 0.80
max.iterations	Numeric (length = 1). Number of iterations to attempt to get convergence before erroring out. Defaults to 1000

Author(s)

Hudson F. Golino <hfg9s at virginia.edu> and Alexander P. Christensen <alexpaulchristensen@gmail.com>

Examples

```
simulated <- simEGM(
  communities = 2, variables = 6,
  loadings = 0.55, # use standard factor loading sizes
  correlations = 0.30,
  sample.size = 1000
)
```

---

tefi — *Total Entropy Fit Index using Von Neumman's entropy (Quantum Information Theory) for correlation matrices*

Description

Computes the fit (TEFI) of a dimensionality structure using Von Neumman's entropy when the input is a correlation matrix. Lower values suggest better fit of a structure to the data.

Usage

```
tefi(data, structure = NULL, verbose = TRUE)
```

Arguments

`data`	Matrix, data frame, or `*EGA` class object. Matrix or data frame can be raw data or a correlation matrix. All `*EGA` objects are accepted. `hierEGA` input will produced the Generalized TEFI (see `genTEFI`)
`structure`	Numeric or character vector (length = `ncol(data)`). Can be theoretical factors or the structure detected by `EGA`
`verbose`	Boolean (length = 1). Whether messages and (insignificant) warnings should be output. Defaults to `TRUE` to see all messages and warnings for every function call. Set to `FALSE` to ignore messages and warnings

Value

Returns a data frame with columns:

Non-hierarchical Structure

`VN.Entropy.Fit`	The Total Entropy Fit Index using Von Neumman's entropy
`Total.Correlation`	The total correlation of the dataset
`Average.Entropy`	The average entropy of the dataset

Hierarchical Structure

`VN.Entropy.Fit` The Generalized Total Entropy Fit Index using Von Neumman's entropy

`Lower.Order.VN` Lower order (only) Total Entropy Fit Index

`Higher.Order.VN`
Higher order (only) Total Entropy Fit Index

Author(s)

Hudson Golino <hfg9s at virginia.edu>, Alexander P. Christensen <alexpaulchristensen@gmail.com>, and Robert Moulder <rgm4fd@virginia.edu>

References

Initial formalization and simulation
Golino, H., Moulder, R. G., Shi, D., Christensen, A. P., Garrido, L. E., Nieto, M. D., Nesselroade, J., Sadana, R., Thiyagarajan, J. A., & Boker, S. M. (2020). Entropy fit indices: New fit measures for assessing the structure and dimensionality of multiple latent variables. *Multivariate Behavioral Research*.

Examples

```
# Load data
wmt <- wmt2[,7:24]

# Estimate EGA model
ega.wmt <- EGA(
  data = wmt, model = "glasso",
  plot.EGA = FALSE # no plot for CRAN checks
)

# Compute entropy indices for empirical EGA
tefi(ega.wmt)

# User-defined structure (with `EGA` object)
tefi(ega.wmt, structure = c(rep(1, 5), rep(2, 5), rep(3, 8)))
```

`TMFG` *Triangulated Maximally Filtered Graph*

Description

Applies the Triangulated Maximally Filtered Graph (TMFG) filtering method (see Massara et al., 2016). The TMFG method uses a structural constraint that limits the number of zero-order correlations included in the network (3*n* - 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 Local-Global Inversion Method (LoGo; see Barfuss et al., 2016 for more details). See **Details** for more information

Usage

```
TMFG(
  data,
  n = NULL,
  corr = c("auto", "cor_auto", "cosine", "pearson", "spearman"),
  na.data = c("pairwise", "listwise"),
  partial = FALSE,
  returnAllResults = FALSE,
  verbose = FALSE,
  ...
)
```

Arguments

`data`	Matrix or data frame. Should consist only of variables to be used in the analysis. Can be raw data or correlation matrix
`n`	Numeric (length = 1). Sample size for when a correlation matrix is input into `data`. Defaults to `NULL`. `n` is not necessary and is provided for better functionality in `EGAnet`
`corr`	Character (length = 1). Method to compute correlations. Defaults to `"auto"`. Available options: • `"auto"` — Automatically computes appropriate correlations for the data using Pearson's for continuous, polychoric for ordinal, tetrachoric for binary, and polyserial/biserial for ordinal/binary with continuous. To change the number of categories that are considered ordinal, use `ordinal.categories` (see `polychoric.matrix` for more details) • `"cor_auto"` — Uses `cor_auto` to compute correlations. Arguments can be passed along to the function • `"cosine"` — Uses `cosine` to compute cosine similarity • `"pearson"` — Pearson's correlation is computed for all variables regardless of categories • `"spearman"` — Spearman's rank-order correlation is computed for all variables regardless of categories For other similarity measures, compute them first and input them into `data` with the sample size (`n`)
`na.data`	Character (length = 1). How should missing data be handled? Defaults to `"pairwise"`. Available options: • `"pairwise"` — Computes correlation for all available cases between two variables • `"listwise"` — Computes correlation for all complete cases in the dataset

`partial` Boolean (length = 1). Whether partial correlations should be output. Defaults to `FALSE`. The TMFG method is based on the zero-order correlations; the Local-Global Inversion Method (LoGo; see Barfuss et al., 2016 for more details) uses the decomposability of the TMFG network to obtain the inverse covariance structure of the network (which is then converted to partial correlations). Set to `TRUE` to obtain the partial correlations from the LoGo method

`returnAllResults` Boolean (length = 1). Whether all results should be returned. Defaults to `FALSE` (network only). Set to `TRUE` to access separators and cliques

`verbose` Boolean (length = 1). Whether messages and (insignificant) warnings should be output. Defaults to `FALSE` (silent calls). Set to `TRUE` to see all messages and warnings for every function call

`...` Additional arguments to be passed on to `auto.correlate`

Details

The TMFG method applies a structural constraint on the network, which restrains the network to retain a certain number of edges (3*n*-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 network or list containing:

`network` The filtered adjacency matrix

`separators` The separators (3-cliques) in the network

`cliques` The cliques (4-cliques) in the network

Author(s)

Alexander Christensen <alexpaulchristensen@gmail.com>

References

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

Psychometric network introduction to TMFG
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.

Triangulated Maximally Filtered Graph
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

```
# TMFG filtered network
TMFG(wmt2[,7:24])

# Partial correlations using the LoGo method
TMFG(wmt2[,7:24], partial = TRUE)
```

---

totalCor *Total Correlation*

Description

Computes the total correlation of a dataset

Usage

```
totalCor(data, base = 2.718282)
```

Arguments

data	Matrix or data frame. Should consist only of variables to be used in the analysis
base	Numeric (length = 1). Base to use for entropy. Defaults to `exp(1)` or `2.718282`

Value

Returns a list containing:

Ind.Entropies	Individual entropies for each variable
Joint.Entropy	The joint entropy of the dataset
Total.Cor	The total correlation of the dataset
Normalized	Total correlation divided by the sum of the individual entropies minus the maximum of the individual entropies

Author(s)

Hudson F. Golino <hfg9s at virginia.edu>

References

Formalization of total correlation
Watanabe, S. (1960). Information theoretical analysis of multivariate correlation. *IBM Journal of Research and Development 4*, 66-82.

Applied implementation
Felix, L. M., Mansur-Alves, M., Teles, M., Jamison, L., & Golino, H. (2021). Longitudinal impact and effects of booster sessions in a cognitive training program for healthy older adults. *Archives of Gerontology and Geriatrics*, *94*, 104337.

Examples

```
# Compute total correlation
totalCor(wmt2[,7:24])
```

`totalCorMat` *Total Correlation Matrix*

Description

Computes the pairwise total correlation (`totalCor`) for a dataset

Usage

```
totalCorMat(data, base = 2.718282, normalized = FALSE)
```

Arguments

`data`	Matrix or data frame. Should consist only of variables to be used in the analysis
`base`	Numeric (length = 1). Base to use for entropy. Defaults to `exp(1)` or `2.718282`
`normalized`	Boolean (length = 1). Should the normalized total correlation be computed? Defaults to `FALSE`

Value

Returns a symmetric matrix with pairwise total correlations

Author(s)

Hudson F. Golino <hfg9s at virginia.edu>

References

Formalization of total correlation
Watanabe, S. (1960). Information theoretical analysis of multivariate correlation. *IBM Journal of Research and Development 4*, 66-82.

Applied implementation
Felix, L. M., Mansur-Alves, M., Teles, M., Jamison, L., & Golino, H. (2021). Longitudinal impact and effects of booster sessions in a cognitive training program for healthy older adults. *Archives of Gerontology and Geriatrics*, *94*, 104337.

Examples

```
# Compute total correlation matrix
totalCorMat(wmt2[,7:24])
```

UVA *Unique Variable Analysis*

Description

Identifies locally dependent (redundant) variables in a multivariate dataset using the EBICglasso.qgraph network estimation method and weighted topological overlap (see Christensen, Garrido, & Golino, 2023 for more details)

Usage

```
UVA(
  data = NULL,
  network = NULL,
  n = NULL,
  key = NULL,
  uva.method = c("MBR", "EJP"),
  cut.off = 0.25,
  reduce = TRUE,
  reduce.method = c("latent", "mean", "remove", "sum"),
  auto = TRUE,
  verbose = FALSE,
  ...
)
```

Arguments

data	Matrix or data frame. Should consist only of variables to be used in the analysis. Can be raw data or a correlation matrix. Defaults to NULL
network	Symmetric matrix or data frame. A symmetric network. Defaults to NULL If both data and network are provided, then UVA will use the network with the data (rather than estimating a network from the data)

`n` Numeric (length = 1). Sample size if `data` provided is a correlation matrix. Defaults to `NULL`

`key` Character vector (length = `ncol(data)`). Item key for labeling variables in the results

`uva.method` Character (length = 1). Whether the method described in Christensen, Garrido, and Golino (2023) publication in *Multivariate Behavioral Research* (`"MBR"`) or Christensen, Golino, and Silvia (2020) publication in *European Journal of Personality* (`"EJP"`) should be used. Defaults to `"MBR"`

Based on simulation and accumulating empirical evidence, the methods described in Christensen, Golino, and Silvia (2020) such as adaptive alpha are **outdated**. Evidence supports using a single cut-off value (regardless of continuous, polytomous, or dichotomous data; Christensen, Garrido, & Golino, 2023)

`cut.off` Numeric (length = 1). Cut-off used to determine when pairwise `wto` values are considered locally dependent (or redundant). Must be values between `0` and `1`. Defaults to `0.25`

This cut-off value is **recommended** and based on extensive simulation (Christensen, Garrido, & Golino, 2023). Printing the result will provide a gradient of pairwise redundancies in increments of 0.20, 0.25, and 0.30. Use `print` or `summary` on the output rather than adjusting this cut-off value

`reduce` Logical (length = 1). Whether redundancies should be reduced in data. Defaults to `TRUE`

`reduce.method` Character (length = 1). Method to reduce redundancies. Available options:

- `"latent"` — Computes latent variables using `cfa` when there are three or more redundant variables. If variables are not all coded in the same direction, then they will be recoded as necessary. A warning will be produced for all variables that are flipped
- `"mean"` — Computes mean of redundant variables. If variables are not all coded in the same direction, then they will be recoded as necessary. A warning will be produced for all variables that are flipped
- `"remove"` — Removes all but one variable from a set of redundant variables
- `"sum"` — Computes sum of redundant variables. If variables are not all coded in the same direction, then they will be recoded as necessary. A warning will be produced for all variables that are flipped

`auto` Logical (length = 1). Whether `reduce` should occur automatically. For `reduce.method = "remove"`, the automated decision process is as follows:

- `Two variables` — The variable with the lowest maximum `wto` to all other variables (other than the one it is redundant with) is retained and the other is removed
- `Three or more variables` — The variable with the highest mean `wto` to all other variables that are redundant with one another is retained and all others are removed

`verbose` Boolean (length = 1). Whether messages and (insignificant) warnings should be output. Defaults to `FALSE` (silent calls). Set to `TRUE` to see all messages and warnings for every function call

`...` Additional arguments that should be passed on to old versions of `UVA` or to `EGA` and `cfa`

References

Most recent simulation and implementation
Christensen, A. P., Garrido, L. E., & Golino, H. (2023). Unique variable analysis: A network psychometrics method to detect local dependence. *Multivariate Behavioral Research*.

Conceptual foundation and outdated methods
Christensen, A. P., Golino, H., & Silvia, P. J. (2020). A psychometric network perspective on the validity and validation of personality trait questionnaires. *European Journal of Personality*, *34*(6), 1095-1108.

Weighted topological overlap
Nowick, K., Gernat, T., Almaas, E., & Stubbs, L. (2009). Differences in human and chimpanzee gene expression patterns define an evolving network of transcription factors in brain. *Proceedings of the National Academy of Sciences*, *106*, 22358-22363.

Selection of CFA Estimator
Rhemtulla, M., Brosseau-Liard, P. E., & Savalei, V. (2012). When can categorical variables be treated as continuous? A comparison of robust continuous and categorical SEM estimation methods under suboptimal conditions. *Psychological Methods*, *17*(3), 354-373.

Examples

```
# Perform UVA
uva.wmt <- UVA(wmt2[,7:24])

# Show summary
summary(uva.wmt)
```

---

vn.entropy *Entropy Fit Index using Von Neumman's entropy (Quantum Information Theory) for correlation matrices*

Description

Computes the fit of a dimensionality structure using Von Neumman's entropy when the input is a correlation matrix. Lower values suggest better fit of a structure to the data

Usage

```
vn.entropy(data, structure)
```

Arguments

`data`	Matrix or data frame. Contains variables to be used in the analysis
`structure`	Numeric or character vector (length = `ncol(data)`). A vector representing the structure (numbers or labels for each item). Can be theoretical factors or the structure detected by `EGA`

Value

Returns a list containing:

`VN.Entropy.Fit` The Entropy Fit Index using Von Neumman's entropy

`Total.Correlation`
: The total correlation of the dataset

`Average.Entropy`
: The average entropy of the dataset

Author(s)

Hudson Golino <hfg9s at virginia.edu>, Alexander P. Christensen <alexpaulchristensen@gmail.com>, and Robert Moulder <rgm4fd@virginia.edu>

References

Initial formalization and simulation

Golino, H., Moulder, R. G., Shi, D., Christensen, A. P., Garrido, L. E., Nieto, M. D., Nesselroade, J., Sadana, R., Thiyagarajan, J. A., & Boker, S. M. (2020). Entropy fit indices: New fit measures for assessing the structure and dimensionality of multiple latent variables. *Multivariate Behavioral Research*.

Examples

```
# Get EGA result
ega.wmt <- EGA(
  data = wmt2[,7:24], model = "glasso",
  plot.EGA = FALSE # no plot for CRAN checks
)

# Compute Von Neumman entropy
vn.entropy(ega.wmt$correlation, ega.wmt$wc)
```

---

wmt2 *WMT-2 Data*

Description

A response matrix (n = 1185) of the Wiener Matrizen-Test 2 (WMT-2).

Usage

```
data(wmt2)
```

Format

A 1185x24 response matrix

Examples

```
data("wmt2")
```

`wto` *Weighted Topological Overlap*

Description

Computes weighted topological overlap following the Novick et al. (2009) definition

Usage

```
wto(network, signed = TRUE, diagonal.zero = TRUE)
```

Arguments

`network`	Symmetric matrix or data frame. A symmetric network
`signed`	Boolean (length = 1). Whether the signed version should be used. Defaults to `TRUE`. Use `FALSE` for absolute values
`diagonal.zero`	Boolean (length = 1). Whether diagonal of overlap matrix should be set to zero. Defaults to `TRUE`. Use `FALSE` to allow overlap of a node with itself

Value

A symmetric matrix of weighted topological overlap values between each pair of variables

References

Original formalization
Nowick, K., Gernat, T., Almaas, E., & Stubbs, L. (2009). Differences in human and chimpanzee gene expression patterns define an evolving network of transcription factors in brain. *Proceedings of the National Academy of Sciences*, *106*, 22358-22363.

Examples

```
# Obtain network
network <- network.estimation(wmt2[,7:24], model = "glasso")

# Compute wTO
wto(network)
```

Index