Package ‘EGAnet’

August 12, 2022

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

Version 1.2.0

Date 2022-12-08

Maintainer Hudson Golino <hfg9s@virginia.edu>


Depends R (>= 3.5.0)

License GPL (>= 3.0)

Encoding UTF-8

LazyData true

Imports glasso, GGally, ggdendro, ggplot2, ggpubr, igraph (>= 1.3.0), lavaan, Matrix, matrixcalc, methods, network, OpenMx, pbapply, qgraph, semPlot, stats
Suggests  GPARotation, gridExtra, knitr, markdown, psych, psychTools, pwr, RColorBrewer, rmarkdown, rstudioapi, sna

VignetteBuilder knitr
RoxygenNote 7.2.1
NeedsCompilation no

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>)

Repository  CRAN
Date/Publication  2022-08-12 17:50:04 UTC

R topics documented:

- EGAnet-package  ................................................................. 3
- boot.ergoInfo  ................................................................. 5
- boot.wmt  ................................................................. 7
- bootEGA  ................................................................. 7
- CFA  ................................................................. 12
- color_palette_EGA  ................................................................. 14
- compare.EGA.plots  ................................................................. 15
- convert2igraph  ................................................................. 17
- depression  ................................................................. 17
- dimensionStability  ................................................................. 18
- dnn.weights  ................................................................. 19
- dynEGA  ................................................................. 20
- dyNEGA.ind.pop  ................................................................. 23
- EBICglasso.qgraph  ................................................................. 25
- EGA  ................................................................. 27
- EGA.estimate  ................................................................. 32
- EGA.fit  ................................................................. 35
- ega.wmt  ................................................................. 38
- Embed  ................................................................. 38
- entropyFit  ................................................................. 39
- ergoInfo  ................................................................. 41
- glia  ................................................................. 42
- hierEGA  ................................................................. 43
- infoCluster  ................................................................. 47
- intelligenceBattery  ................................................................. 48
- invariance  ................................................................. 49
- itemStability  ................................................................. 50
- jsd  ................................................................. 53
- LCT  ................................................................. 54
- louvain  ................................................................. 56
- mctest.ergoInfo  ................................................................. 57
EGAnet-package

Description

Implements the Exploratory Graph Analysis (EGA; Golino & Epskamp, 2017; Golino, Shi et al., 2020) framework for dimensionality and psychometric assessment. EGA is part of a new area called network psychometrics that uses undirected network models for the assessment of psychometric properties. EGA estimates the number of dimensions (or factors) using graphical lasso EBICglasso or Triangulated Maximally Filtered Graph (TMFG) and a weighted network community detection algorithm (Christensen, Garrido, Golino, under review A). A bootstrap method for verifying the stability of the dimensions and items in those dimensions is available (bootEGA; Christensen & Golino, 2021a). The fit of the structure suggested by EGA can be verified using Entropy Fit Indices (entropyFit, tefi; Golino, Moulder et al., 2020). A novel approach called Unique Variable Analysis (UVA) can be used to identify and reduce redundant variables in multivariate data (Christensen, Garrido, & Golino, under review B). Network loadings (net.loads), which are roughly equivalent to factor loadings when the data generating model is a factor model, are available (Christensen & Golino, 2021b, 2021c). Network scores (net.scores) can also be computed using the network loadings. Finally, dynamic EGA (dynEGA) will estimate dimensions from time series data for individual, group, and sample levels (Golino, Christensen et al., 2021).

Author(s)

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


# Related functions: EGA


# Related functions: UVA


# Related functions: bootEGA, dimensionStability, # and itemStability


# Related functions: LCT


# Related functions: LCT and net.loads


# Related functions: bootEGA, dimensionStability, # EGA, itemStability, and UVA


# Related functions: dynEGA and simDFM


# Related functions: EGA


# Related functions: EGA


# Related functions: entropyFit, tefi, and vn.entropy


# Related functions: EGA


# Related functions: EGA.fit and tefi

# Related functions: EGA.fit and tefi

---

**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 bootstrap sampling. 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.

A more efficient process, the approach applied here, is to obtain a sampling distribution of EII values as if all participants in the data have the population network structure. Sampling is not perfect and therefore random noise is added to the edges of the population structure to simulate sampling variability. This noise follows a random uniform distribution ranging from -0.10 to 0.10. In addition, a proportion of edges are rewired to allow for slight variations on the population structure. The proportion of nodes that are rewired is sampled from a random uniform distribution between 0.10 to 0.40. This process is carried out for each participant resulting in n variations of the population structure. Afterward, EII is computed. This process is carried out for i iterations (e.g., 100).

The result is a sampling distribution of EII values that would be expected if the process was ergodic. If the empirical EII value is significantly less than the distribution or not significantly different, then the empirical data can be expected to be generated from an ergodic process and the population structure is sufficient to describe all individuals. If the empirical EII value is significantly greater than the distribution, then the empirical data cannot be described by the population structure – significant information is lost when collapsing across to the population structure.

### Usage

```r
boot.ergoInfo(dynEGA.object, EII, iter = 100, ncores)
```

### Arguments

- **dynEGA.object** A `dynEGA` or a `dynEGA.ind.pop` object that is used to match the arguments of the EII object.
- **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`
- **iter** Numeric integer. Number of replica samples to generate from the bootstrap analysis. At least 100 is recommended
- **ncores** Numeric. Number of cores to use in computing results. Defaults to `parallel::detectCores() / 2` or half of your computer’s processing power. Set to 1 to not use parallel computing. Recommended to use maximum number of cores minus one

If you’re unsure how many cores your computer has, then use the following code: `parallel::detectCores()`
Value

Returns a list containing:

- `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 the expected value of the EII with small variation in the population structure.
- `effect`: Indicates whether the empirical EII is greater or less than the bootstrap distribution of EII.
- `interpretation`: How you can interpret the result of the test in plain English.
- `plot.dist`: Histogram of the bootstrapped ergodicity information index.
- `methods`: Methods to report for print/summary S3 methods and automated Methods section.

Author(s)

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

References


Examples

```r
# Dynamic EGA individual and population structures
dyn1 <- dynEGA.ind.pop(
  data = sim.dynEGA[, -c(22)], n.embed = 5, tau = 1,
  delta = 1, id = 21, use.derivatives = 1,
  model = "glasso", ncores = 2, corr = "pearson"
)

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

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

bootEGA results using the "glasso" model and 500 iterations of the Wiener Matrizen-Test 2 (WMT-2)

Usage

data(boot.wmt)

data(boot.wmt)

Format

A list with 9 objects (see bootEGA)
A list with 8 objects (see bootEGA)

Details

bootEGA results using the "glasso" model and 500 iterations of the Wiener Matrizen-Test 2 (WMT-2)

Examples

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

Description

bootEGA Estimates the number of dimensions of $n$ bootstraps using the empirical (partial) correlation matrix (parametric) or resampling from the empirical dataset (non-parametric). It also estimates a typical median network structure, which is formed by the median or mean pairwise (partial) correlations over the $n$ bootstraps.
Usage

```r
bootEGA(
  data,
  n = NULL,
  uni.method = c("expand", "LE", "louvain"),
  iter,
  type = c("parametric", "resampling"),
  seed = 1234,
  corr = c("cor_auto", "pearson", "spearman"),
  EGA.type = c("EGA", "EGA.fit", "hierEGA", "riEGA"),
  model = c("glasso", "TMFG"),
  model.args = list(),
  algorithm = c("walktrap", "leiden", "louvain"),
  algorithm.args = list(),
  consensus.method = c("highest_modularity", "most_common", "iterative", "lowest_tefi"),
  consensus.iter = 100,
  typicalStructure = TRUE,
  plot.typicalStructure = TRUE,
  plot.args = list(),
  ncores,
  ...
)
```

Arguments

- **data**: Matrix or data frame. Includes the variables to be used in the bootEGA analysis.
- **n**: Integer. Sample size if data provided is a correlation matrix.
- **uni.method**: Character. What unidimensionality method should be used? Defaults to "louvain".
  Current options are:
  - **expand**: Expands the correlation matrix with four variables correlated .50. If the number of dimensions returns 2 or less in check, then the data are unidimensional; otherwise, regular EGA with no matrix expansion is used. This is the method used in the Golino et al. (2020) *Psychological Methods* simulation.
  - **LE**: Applies the Leading Eigenvalue algorithm ([cluster_leading_eigen](https://CRAN.R-project.org/package=igraph)) on the empirical correlation matrix. If the number of dimensions is 1, then the Leading Eigenvalue solution is used; otherwise, regular EGA is used. This is the final method used in the Christensen, Garrido, and Golino (2021) simulation.
  - **louvain**: Applies the Louvain algorithm ([cluster_louvain](https://CRAN.R-project.org/package=igraph)) on the empirical correlation matrix using a resolution parameter = 0.95. If the number of dimensions is 1, then the Louvain solution is used; otherwise, regular EGA is used. This method was validated in the Christensen (2022) simulation.
- **iter**: Numeric integer. Number of replica samples to generate from the bootstrap analysis. At least 500 is recommended.
- **type**: Character. A string indicating the type of bootstrap to use. Current options are:
"parametric" Generates n new datasets (multivariate normal random distributions) based on the original dataset, via the `mvrnorm` function

"resampling" Generates n random subsamples of the original data

**seed**
Numeric. Seed to reproduce results. Defaults to 1234. For random results, set to NULL

**corr**
Character. Type of correlation matrix to compute. The default uses `cor_auto`. Current options are:

- `cor_auto` Computes the correlation matrix using the `cor_auto` function from `qgraph`
- `pearson` Computes Pearson’s correlation coefficient using the pairwise complete observations via the `cor` function
- `spearman` Computes Spearman’s correlation coefficient using the pairwise complete observations via the `cor` function

**EGA.type**
Character. Type of EGA model to use. Current options are:

- `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

**model**
Character. A string indicating the method to use. Current options are:

- `glasso` Estimates the Gaussian graphical model using graphical LASSO with extended Bayesian information criterion to select optimal regularization parameter. This is the default method
- `TMFG` Estimates a Triangulated Maximally Filtered Graph

**model.args**
List. A list of additional arguments for `EBICglasso.qgraph` or `TMFG`

**algorithm**
A string indicating the algorithm to use or a function from `igraph` Defaults to "walktrap". Current options are:

- `walktrap` Computes the Walktrap algorithm using `cluster_walktrap`
- `leiden` Computes the Leiden algorithm using `cluster_leiden`. Defaults to `objective_function = "modularity"
- `louvain` Computes the Louvain algorithm using `cluster_louvain`

**algorithm.args**
List. A list of additional arguments for `cluster_walktrap, cluster_leiden`, or some other community detection algorithm function (see examples)

**consensus.method**
Character. What consensus clustering method should be used? Defaults to "highest_modularity". Current options are:

- `highest_modularity` Uses the community solution that achieves the highest modularity across iterations
- `most_common` Uses the community solution that is found the most across iterations
• **iterative** Identifies the most common community solutions across iterations and determines how often nodes appear in the same community together. A threshold of 0.30 is used to set low proportions to zero. This process repeats iteratively until all nodes have a proportion of 1 in the community solution.

• **lowest_tefi** Uses the community solution that achieves the lowest $t_{efi}$ across iterations

**consensus.iter** Numeric. Number of iterations to perform in consensus clustering for the Louvain algorithm (see Lancichinetti & Fortunato, 2012). Defaults to 100

**typicalStructure** Boolean. If TRUE, returns the typical network of partial correlations (estimated via graphical lasso or via TMFG) and estimates its dimensions. The "typical network" is the median of all pairwise correlations over the $n$ bootstraps. Defaults to TRUE

**plot.typicalStructure** Boolean. If TRUE, returns a plot of the typical network (partial correlations), which is the median of all pairwise correlations over the $n$ bootstraps, and its estimated dimensions. Defaults to TRUE

**plot.args** List. A list of additional arguments for the network plot. See ggnet2 for full list of arguments:

• `vsize` Size of the nodes. Defaults to 6.
• `label.size` Size of the labels. Defaults to 5.
• `alpha` The level of transparency of the nodes, which might be a single value or a vector of values. Defaults to 0.7.
• `edge.alpha` The level of transparency of the edges, which might be a single value or a vector of values. Defaults to 0.4.
• `legend.names` A vector with names for each dimension
• `color.palette` The color palette for the nodes. For custom colors, enter HEX codes for each dimension in a vector. See color_palette_EGA for more details and examples

**ncores** Numeric. Number of cores to use in computing results. Defaults to `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 use the following code: `parallel::detectCores()`

... Additional arguments. Used for deprecated arguments from previous versions of EGA

**Value**

Returns a list containing:

- **iter** Number of replica samples in bootstrap
- **boot.ndim** Number of dimensions identified in each replica sample
- **boot.wc** Item allocation for each replica sample
bootEGA

bootGraphs Networks of each replica sample
summary.table Summary table containing number of replica samples, median, standard deviation, standard error, 95% confidence intervals, and quantiles (lower = 2.5% and upper = 97.5%)
frequency Proportion of times the number of dimensions was identified (e.g., .85 of 1,000 = 850 times that specific number of dimensions was found)
EGA Output of the original EGA results
typicalGraph A list containing:
  • graph Network matrix of the median network structure
  • typical.dim.variables An ordered matrix of item allocation
  • wc Item allocation of the median network

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

References
# Original implementation of bootEGA
# Structural consistency (see dimensionStability)

See Also
EGA to estimate the number of dimensions of an instrument using EGA and CFA to verify the fit of the structure suggested by EGA using confirmatory factor analysis.

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

# Standard EGA example
boot.wmt <- bootEGA(
  data = wmt, iter = 100, # recommended 500
  plot.typicalStructure = FALSE, # No plot for CRAN checks
  type = "parametric", ncores = 2
)

# Produce Methods section
methods.section(boot.wmt)

# Louvain example
boot.wmt.louvain <- bootEGA(
data = wmt, iter = 100, # recommended 500
algorithm = "louvain",
plot.typicalStructure = FALSE, # No plot for CRAN checks
type = "parametric", ncores = 2
)

# Spinglass example
boot.wmt.spinglass <- bootEGA(
  data = wmt, iter = 100, # recommended 500
  algorithm = igraph::cluster_spinglass, # use any function from {igraph}
  plot.typicalStructure = FALSE, # No plot for CRAN checks
type = "parametric", ncores = 2
)

# EGA fit example
boot.wmt.fit <- bootEGA(
  data = wmt, iter = 100, # recommended 500
  EGA.type = "EGA.fit",
  plot.typicalStructure = FALSE, # No plot for CRAN checks
type = "parametric", ncores = 2
)

# Hierarchical EGA example
boot.wmt.hier <- bootEGA(
  data = wmt, iter = 100, # recommended 500
  EGA.type = "hierEGA",
  plot.typicalStructure = FALSE, # No plot for CRAN checks
type = "parametric", ncores = 2
)

# Random-intercept EGA example
boot.wmt.ri <- bootEGA(
  data = wmt, iter = 100, # recommended 500
  EGA.type = "riEGA",
  plot.typicalStructure = FALSE, # No plot for CRAN checks
type = "parametric", ncores = 2
)

---

**CFA**

*Description*

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

*Usage*

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

---

**CFA Fit of EGA Structure**
**Arguments**

- `ega.obj`: An EGA object
- `data`: A dataframe with the 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**


**See Also**

EGA to estimate the number of dimensions of an instrument using EGA and bootEGA to investigate the stability of EGA’s estimation via bootstrap.

**Examples**

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

# Estimate EGA
ega.wmt <- EGA(
  data = wmt,
  estimator = 'WLSMV',
  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")

---

color_palette_EGA EGA Color Palettes

**Description**

Color palettes for plotting ggnet2 EGA network plots

**Usage**

color_palette_EGA(name, wc, sorted = FALSE)

**Arguments**

- **name** Character. Name of color scheme (see RColorBrewer). Defaults to "polychrome". EGA palettes:
  - "polychrome" Default 20 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** 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 TRUE

**Value**

Vector of colors for community memberships
compare.EGA.plots

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

Examples

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

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

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

Description
Organizes EGA plots for comparison. Ensures that nodes are placed in the same layout to maximize comparison. Community memberships are also homogenized across EGA outputs to enhance interpretation.

Usage

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

Arguments

... 
EGA objects
input.list List. Bypasses ... argument in favor of using a list as an input
base.plot Numeric. 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 vector. Labels for each EGA object
rows Numeric. Number of rows to spread plots across
columns Numeric. Number of columns to spread plots down
plot.args  List. A list of additional arguments for the network plot. For plot.type = "qgraph":

- vsize Size of the nodes. Defaults to 6.

(see ggnet2 for full list of arguments):

- vsize Size of the nodes. Defaults to 6.
- label.size Size of the labels. Defaults to 5.
- alpha The level of transparency of the nodes, which might be a single value or a vector of values. Defaults to 0.7.
- edge.alpha The level of transparency of the edges, which might be a single value or a vector of values. Defaults to 0.4.
- legend.names A vector with names for each dimension
- color.palette The color palette for the nodes. For custom colors, enter HEX codes for each dimension in a vector. See color_palette_EGA for more details and examples

Value

Visual comparison of EGAnet objects

Author(s)

Alexander Christensen <alexpaulchristensen@gmail.com>

Examples

# Obtain SAPA items
items <- psychTools::spi[,c(11:20)]

# Draw random samples
sample1 <- items[sample(1:nrow(items), 1000),]
sample2 <- items[sample(1:nrow(items), 1000),]

# Estimate EGAs
g1 <- EGA(sample1)
g2 <- EGA(sample2)

# Compare EGAs via plot
compare.EGA.plots(
  g1, g2,
  base.plot = 1, # use "g1" as base for comparison
  labels = c("Sample 1", "Sample 2"),
  rows = 1, columns = 2
)
convert2igraph

Convert networks to igraph format

Usage

correct2igraph(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 Van-
derbilt.Edu>

Examples

convert2igraph(ega.wmt$network)

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)

data(depression)
 dimensionStability

Format

A 574x78 response matrix
A 574x78 response matrix

Examples

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

dimensionStability  
*Dimension Stability Statistics from bootEGA*

Description

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

Usage

```r
dimensionStability(bootega.obj, ...)
```

Arguments

- `bootega.obj` A bootEGA object
- `...` Additional arguments. Used for deprecated arguments from previous versions of dimStability

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


See Also

EGA to estimate the number of dimensions of an instrument using EGA and CFA to verify the fit of the structure suggested by EGA using confirmatory factor analysis.

Examples

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

# Estimate bootstrap EGA
boot.wmt <- bootEGA(
  data = wmt, iter = 100, # recommended 500
  plot.typicalStructure = FALSE, # No plot for CRAN checks
  type = "parametric", ncores = 2
)

# Estimate stability statistics
res <- dimensionStability(boot.wmt)
res$dimension.stability

# Produce Methods section
methods.section(
  boot.wmt,
  stats = "dimensionStability"
)
```

---

**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)

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)
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

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

Format

A list of with a length of 4
A list of with a length of 4

Examples

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

dynEGA  

**Dynamic Exploratory Graph Analysis**

Description

Estimates dynamic factors in multivariate time series (i.e. longitudinal data, panel data, intensive longitudinal data) at multiple time scales, in different levels of analysis: individuals (intraindividual structure), groups or population (structure of the population). Exploratory graph analysis is applied in the derivatives estimated using generalized local linear approximation (glla). Instead of estimating factors by modeling how variables are covarying, as in traditional EGA, dynEGA is a dynamic model that estimates the factor structure by modeling how variables are changing together. 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

```r
dynEGA(
  data,  
n.embed,  
tau = 1,  
delta = 1,  
level = c("individual", "group", "population"),  
id = NULL,  
group = NULL,  
use.derivatives = 1,  
model = c("glasso", "TMFG"),  
model.args = list(),
)```
algorithm = c("walktrap", "leiden", "louvain"),
algorithm.args = list(),
corr = c("cor_auto", "pearson", "spearman"),
ncores,
...)

Arguments

data A dataframe with the variables to be used in the analysis. The dataframe should be in a long format (i.e. observations for the same individual (for example, individual 1) are placed in order, from time 1 to time t, followed by the observations from individual 2, also ordered from time 1 to time t.)
n.embed Integer. 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 Integer. Number of observations to offset successive embeddings in the Embed function. A tau of one uses adjacent observations. Default is "tau = 1".
delta Integer. The time between successive observations in the time series. Default is "delta = 1".
level Character. A string indicating the level of analysis. If the interest is in modeling the intraindividual structure only (one dimensionality structure per individual), then level should be set to "individual". If the interest is in the structure of a group of individuals, then level should be set to "group". Finally, if the interest is in the population structure, then level should be set to "population".

Current options are:

• individual Estimates the dynamic factors per individual. This should be the preferred method if one is interested in the factor structure of individuals. An additional parameter ("id") needs to be provided identifying each individual.
• group Estimates the dynamic factors for each group. An additional parameter ("group") needs to be provided identifying the group membership.
• population Estimates the dynamic factors of the population

id Numeric. Number of the column identifying each individual.
group Numeric or character. Number of the column identifying group membership. Must be specified only if level = "group".

use.derivatives Integer. The order of the derivative to be used in the EGA procedure. Default to 1.

model Character. A string indicating the method to use. Current options are:

• glasso Estimates the Gaussian graphical model using graphical LASSO with extended Bayesian information criterion to select optimal regularization parameter. This is the default method
• TMFG Estimates a Triangulated Maximally Filtered Graph

model.args List. A list of additional arguments for EBICglasso.qgraph or TMFG
algorithm A string indicating the algorithm to use or a function from `igraph` Defaults to "walktrap". Current options are:

- `walktrap` Computes the Walktrap algorithm using `cluster_walktrap`
- `leiden` Computes the Leiden algorithm using `cluster_leiden`. Defaults to `objective_function = "modularity"`
- `louvain` Computes the Louvain algorithm using `cluster_louvain`

algorithm.args List. A list of additional arguments for `cluster_walktrap, cluster_louvain,` or some other community detection algorithm function (see examples)

corr Type of correlation matrix to compute. The default uses "pearson". Current options are:

- `cor_auto` Computes the correlation matrix using the `cor_auto` function from `qgraph`.
- `pearson` Computes Pearson’s correlation coefficient using the pairwise complete observations via the `cor` function.
- `spearman` Computes Spearman’s correlation coefficient using the pairwise complete observations via the `cor` function.

ncores Numeric. Number of cores to use in computing results. Defaults to `parallel::detectCores() / 2` or half of your computer’s processing power. Set to 1 to not use parallel computing. Recommended to use maximum number of cores minus one.

If you’re unsure how many cores your computer has, then use the following code: `parallel::detectCores()`

... Additional arguments. Used for deprecated arguments from previous versions of `EGA`

Author(s)

Hudson Golino <hfg9s at virginia.edu>

References


Examples

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

# Population structure
dyn.random <- dynEGA(
  data = sim.dynEGA, n.embed = 5, tau = 1,
  delta = 1, id = 21, group = 22, use.derivatives = 1,
  level = "population", ncores = 2, corr = "pearson"
)

# Plot population structure
plot(dyn.random)

# Group structure
dyn.group <- dynEGA(
  data = sim.dynEGA, n.embed = 5, tau = 1,
  delta = 1, id = 21, group = 22, use.derivatives = 1,
  level = "group", ncores = 2, corr = "pearson"
)

# Plot group structure
plot(dyn.group, ncol = 2, nrow = 1)

# Intraindividual structure
dyn.individual <- dynEGA(
  data = sim.dynEGA, n.embed = 5, tau = 1,
  delta = 1, id = 21, group = 22, use.derivatives = 1,
  level = "individual", ncores = 2, corr = "pearson"
)

# Plot individual structure (participant 1)
plot(dyn.individual, id = 1)
```

---

**Description**

DynEGA estimates dynamic factors in multivariate time series (i.e. longitudinal data, panel data, intensive longitudinal data) at multiple time scales, in different levels of analysis: individuals (intraindividual structure) and population (structure of the population). Exploratory graph analysis is applied in the derivatives estimated using generalized local linear approximation (glla). Instead of estimating factors by modeling how variables are covarying, as in traditional EGA, dynEGA is a dynamic model that estimates the factor structure by modeling how variables are changing together. 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

dynEGA.ind.pop(
  data,
  n.embed,
  tau = 1,
  delta = 1,
  id = NULL,
  use.derivatives = 1,
  model = c("glasso", "TMFG"),
  model.args = list(),
  algorithm = c("walktrap", "leiden", "louvain"),
  algorithm.args = list(),
  corr = c("cor_auto", "pearson", "spearman"),
  ncores,
  ...
)

Arguments

data
  A data frame with the variables to be used in the analysis. The data frame should
  be in a long format (i.e. observations for the same individual (for example, indi-
  vidual 1) are placed in order, from time 1 to time t, followed by the observations
  from individual 2, also ordered from time 1 to time t.)

n.embed
  Integer. 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
  Integer. Number of observations to offset successive embeddings in the Embed
  function. A tau of one uses adjacent observations. Default is tau = 1.

delta
  Integer. The time between successive observations in the time series. Default is
  delta = 1.

id
  Numeric. Number of the column identifying each individual.

use.derivatives
  Integer. The order of the derivative to be used in the EGA procedure. Default to
  1.

model
  Character. A string indicating the method to use. Defaults to glasso. Current
  options are:
    • glasso Estimates the Gaussian graphical model using graphical LASSO
      with extended Bayesian information criterion to select optimal regulariza-
      tion parameter. This is the default method
    • TMFG Estimates a Triangulated Maximally Filtered Graph

model.args
  List. A list of additional arguments for EBICglasso.qgraph or TMFG

algorithm
  A string indicating the algorithm to use or a function from igraph Defaults to
  "walktrap". Current options are:
    • walktrap Computes the Walktrap algorithm using cluster_walktrap
• leiden Computes the Leiden algorithm using `cluster_leiden`. Defaults to objective_function = "modularity"
• louvain Computes the Louvain algorithm using `cluster_louvain`

algorithm.args List. A list of additional arguments for `cluster_walktrap`, `cluster_louvain`, or some other community detection algorithm function (see examples)
corr Type of correlation matrix to compute. The default uses `cor_auto`. Current options are:
  • cor_auto Computes the correlation matrix using the `cor_auto` function from `qgraph`.
  • pearson Computes Pearson’s correlation coefficient using the pairwise complete observations via the `cor` function.
  • spearman Computes Spearman’s correlation coefficient using the pairwise complete observations via the `cor` function.

ncores Numeric. Number of cores to use in computing results. Defaults to `parallel::detectCores() / 2` or half of your computer’s processing power. Set to 1 to not use parallel computing. Recommended to use maximum number of cores minus one

... Additional arguments. Used for deprecated arguments from previous versions of `EGA`

Author(s)

Hudson Golino <hfg9s at virginia.edu>

Examples

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

# Dynamic EGA individual and population structure
dyn.ega1 <- dynEGA.ind.pop(
  data = sim.dynEGA, n.embed = 5, tau = 1,
  delta = 1, id = 21, use.derivatives = 1,
  ncores = 2, corr = "pearson"
)

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 criterium (EBIC) described by Foygel & Drton (2010).
Usage

EBICglasso.qgraph(
  data,
  n = NULL,
  gamma = 0.5,
  penalize.diagonal = FALSE,
  nlambda = 100,
  lambda.min.ratio = 0.01,
  returnAllResults = FALSE,
  penalizeMatrix,
  countDiagonal = FALSE,
  refit = FALSE,
  ...
)

Arguments

  data  Data matrix
  n     Number of participants
  gamma EBIC tuning parameter. 0.5 is generally a good choice. Setting to zero will
         cause regular BIC to be used.
  penalize.diagonal Should the diagonal be penalized?
  nlambda Number of lambda values to test.
  lambda.min.ratio Ratio of lowest lambda value compared to maximal lambda
  returnAllResults If TRUE this function does not return a network but the results of the entire glasso
                     path.
  penalizeMatrix Optional logical matrix to indicate which elements are penalized
  countDiagonal Should diagonal be counted in EBIC computation? Defaults to FALSE. Set to
                   TRUE to mimic qgraph < 1.3 behavior (not recommended!).
  refit Logical, should the optimal graph be refitted without LASSO regularization?
           Defaults to FALSE.
  ...   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
EGA

Author(s)

Sacha Epskamp <mail@sachaepskamp.com>

References

# Instantiation of GLASSO

# Tutorial on EBICglasso

# glasso package

# glasso + EBIC

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

*Applies the Exploratory Graph Analysis technique*

Description

Estimates the number of dimensions of a given dataset or correlation matrix using the graphical lasso (*EBICglasso.qgraph*) or the Triangulated Maximally Filtered Graph (*TMFG*) network estimation methods.
Usage

EGA(
  data,
  n = NULL,
  corr = c("cor_auto", "pearson", "spearman"),
  uni.method = c("expand", "LE", "louvain"),
  model = c("glasso", "TMFG"),
  model.args = list(),
  algorithm = c("walktrap", "leiden", "louvain"),
  algorithm.args = list(),
  consensus.method = c("highest_modularity", "most_common", "iterative", "lowest_tefi"),
  consensus.iter = 100,
  plot.EGA = TRUE,
  plot.args = list(),
  ...
)

Arguments

data Matrix or data frame. Variables (down columns) or correlation matrix. If the input is a correlation matrix, then argument n (number of cases) is required

n Integer. Sample size if data provided is a correlation matrix

corr Type of correlation matrix to compute. The default uses cor_auto. Current options are:
  • cor_auto Computes the correlation matrix using the cor_auto function from qgraph.
  • pearson Computes Pearson’s correlation coefficient using the pairwise complete observations via the cor function.
  • spearman Computes Spearman’s correlation coefficient using the pairwise complete observations via the cor function.

uni.method Character. What unidimensionality method should be used? Defaults to "LE". Current options are:
  • expand Expands the correlation matrix with four variables correlated .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 is the method used in the Golino et al. (2020) Psychological Methods simulation.
  • LE Applies the Leading Eigenvalue algorithm (clusterLeadingEigen) on the empirical correlation matrix. If the number of dimensions is 1, then the Leading Eigenvalue solution is used; otherwise, regular EGA is used. This is the final method used in the Christensen, Garrido, and Golino (2021) simulation.
  • louvain Applies the Louvain algorithm (clusterLouvain) on the empirical correlation matrix using a resolution parameter = 0.95. If the number of dimensions is 1, then the Louvain solution is used; otherwise, regular EGA is used. This method was validated in the Christensen (2022) simulation.
EGA

model Character. A string indicating the method to use. Defaults to "glasso". Current options are:

- glasso Estimates the Gaussian graphical model using graphical LASSO with extended Bayesian information criterion to select optimal regularization parameter
- TMFG Estimates a Triangulated Maximally Filtered Graph

model.args List. A list of additional arguments for EBICglasso.qgraph or TMFG

algorithm A string indicating the algorithm to use or a function from igraph Defaults to "walktrap". Current options are:

- walktrap Computes the Walktrap algorithm using cluster_walktrap
- leiden Computes the Leiden algorithm using cluster_leiden. Defaults to objective_function = "modularity"
- louvain Computes the Louvain algorithm using cluster_louvain

algorithm.args List. A list of additional arguments for cluster_walktrap, cluster_leiden, or some other community detection algorithm function (see examples)

consensus.method Character. What consensus clustering method should be used? Defaults to "highest_modularity". Current options are:

- highest_modularity Uses the community solution that achieves the highest modularity across iterations
- most_common Uses the community solution that is found the most across iterations
- iterative Identifies the most common community solutions across iterations and determines how often nodes appear in the same community together. A threshold of 0.30 is used to set low proportions to zero. This process repeats iteratively until all nodes have a proportion of 1 in the community solution.
- lowest_tefi Uses the community solution that achieves the lowest tefi across iterations

consensus.iter Numeric. Number of iterations to perform in consensus clustering for the Louvain algorithm (see Lancichinetti & Fortunato, 2012). Defaults to 100

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

plot.args List. A list of additional arguments for the network plot. For plot.type = "qgraph":

- vsize Size of the nodes. Defaults to 6.

For plot.type = "Ggally" (see ggnet2 for full list of arguments):

- vsize Size of the nodes. Defaults to 6.
- label.size Size of the labels. Defaults to 5.
- alpha The level of transparency of the nodes, which might be a single value or a vector of values. Defaults to 0.7.
- edge.alpha The level of transparency of the edges, which might be a single value or a vector of values. Defaults to 0.4.
EGA

- **legend.names**: A vector with names for each dimension
- **color.palette**: The color palette for the nodes. For custom colors, enter HEX codes for each dimension in a vector. See `color_palette_EGA` for more details and examples

... Additional arguments. Used for deprecated arguments from previous versions of EGA

**Details**

Two community detection algorithms, Walktrap (Pons & Latapy, 2006) and Louvain (Blondel et al., 2008), are pre-programmed because of their superior performance in simulation studies on psychological data generated from factor models (Christensen & Golino; 2020; Golino et al., 2020). Notably, any community detection algorithm from the `igraph` can be used to estimate the number of communities (see examples).

**Value**

Returns a list containing:

- **network**: A symmetric network estimated using either the `EBICglasso.qgraph` or `TMFG`.
- **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.

**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**

# Louvain algorithm

# Comprehensive unidimensionality simulation

# Compared all `igraph` community detections algorithms, introduced Louvain algorithm, simulation with continuous and polytomous data
# Also implements the Leading Eigenvalue unidimensional method

# Original simulation and implementation of EGA

# Current implementation of EGA, introduced unidimensional checks, continuous and dichotomous data

# Walktrap algorithm

See Also

bootEGA to investigate the stability of EGA’s estimation via bootstrap and CFA to verify the fit of the structure suggested by EGA using confirmatory factor analysis.

Examples

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

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

# Summary statistics
summary(ega.wmt)

# Produce Methods section
methods.section(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 Leiden algorithm
ega.wmt.leiden <- EGA(
data = wmt, algorithm = "leiden",
  plot.EGA = FALSE # No plot for CRAN checks
)
```
# Estimate EGA with Spinglass algorithm
ega.wmt.spinglass <- EGA(
  data = wmt,
  algorithm = igraph::cluster_spinglass, # any {igraph} algorithm
  plot.EGA = FALSE # No plot for CRAN checks
)

EGA.estimate

A Sub-routine Function for EGA

Description

Estimates the number of dimensions of a given dataset or correlation matrix using the graphical lasso (EBICglasso.qgraph) or the Triangulated Maximally Filtered Graph (TMFG) network estimation methods.

Usage

EGA.estimate(
  data,
  n = NULL,
  corr = c("cor_auto", "pearson", "spearman"),
  model = c("glasso", "TMFG"),
  model.args = list(),
  algorithm = c("walktrap", "leiden", "louvain"),
  algorithm.args = list(),
  consensus.method = c("highest_modularity", "most_common", "iterative", "lowest_tei"),
  consensus.iter = 100,
  ...
)

Arguments

data Matrix or data frame. Variables (down columns) or correlation matrix. If the input is a correlation matrix, then argument n (number of cases) is required

n Integer. Sample size if data provided is a correlation matrix

corr Type of correlation matrix to compute. The default uses cor_auto. Current options are:

  * cor_auto Computes the correlation matrix using the cor_auto function from qgraph.
  * pearson Computes Pearson’s correlation coefficient using the pairwise complete observations via the cor function.
  * spearman Computes Spearman’s correlation coefficient using the pairwise complete observations via the cor function.
EGA.estimate

model Character. A string indicating the method to use.
Current options are:
- glasso Estimates the Gaussian graphical model using graphical LASSO with extended Bayesian information criterion to select optimal regularization parameter. This is the default method
- TMFG Estimates a Triangulated Maximally Filtered Graph

model.args List. A list of additional arguments for EBICglasso.qgraph or TMFG

algorithm A string indicating the algorithm to use or a function from igraph. Current options are:
- walktrap Computes the Walktrap algorithm using cluster_walktrap
- leiden Computes the Leiden algorithm using cluster_leiden
- louvain Computes the Louvain algorithm using cluster_louvain

algorithm.args List. A list of additional arguments for cluster_walktrap, cluster_louvain, or some other community detection algorithm function (see examples)

consensus.method Character. What consensus clustering method should be used? Defaults to “highest_modularity”. Current options are:
- highest_modularity Uses the community solution that achieves the highest modularity across iterations
- most_common Uses the community solution that is found the most across iterations
- iterative Identifies the most common community solutions across iterations and determines how often nodes appear in the same community together. A threshold of 0.30 is used to set low proportions to zero. This process repeats iteratively until all nodes have a proportion of 1 in the community solution.
- lowest_tefi Uses the community solution that achieves the lowest tefi across iterations

consensus.iter Numeric. Number of iterations to perform in consensus clustering for the Louvain algorithm (see Lancichinetti & Fortunato, 2012). Defaults to 100

Details

Two community detection algorithms, Walktrap (Pons & Latapy, 2006) and Louvain (Blondel et al., 2008), are pre-programmed because of their superior performance in simulation studies on psychological data generated from factor models (Christensen & Golino; 2020; Golino et al., 2020). Notably, any community detection algorithm from the igraph can be used to estimate the number of communities (see examples).

Value

Returns a list containing:
estimated.network

A symmetric network estimated using either the `EBICglasso.qgraph` or `TMFG`.

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.

Author(s)

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

References

# Louvain algorithm

# Compared all `igraph` community detections algorithms, introduced Louvain algorithm, simulation with continuous and polytomous data

# Original simulation and implementation of EGA


# Current implementation of EGA, introduced unidimensional checks, continuous and dichotomous data

# Walktrap algorithm

See Also

`bootEGA` to investigate the stability of EGA’s estimation via bootstrap and `CFA` to verify the fit of the structure suggested by EGA using confirmatory factor analysis.

Examples

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

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

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

# Estimate EGA with Louvain algorithm
ega.wmt.louvain <- EGA.estimate(data = wmt, algorithm = "louvain")

# Estimate EGA with Spinglass algorithm
ega.wmt.spinglass <- EGA.estimate(
    data = wmt,
    algorithm = igraph::cluster_spinglass # any (igraph) algorithm
)

---

**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**

```r
EGA.fit(
  data,
  n = NULL,
  uni.method = c("expand", "LE"),
  corr = c("cor_auto", "pearson", "spearman"),
  model = c("glasso", "TMFG"),
  algorithm = c("leiden", "walktrap"),
  algorithm.args = list(steps = c(3:8), resolution_parameter = seq(0, 2, 0.001))
)
```

**Arguments**

- `data` Matrix or data frame. Dataset or correlation matrix
- `n` Integer. Sample size (if the data provided is a correlation matrix)
- `uni.method` Character. What unidimensionality method should be used? Defaults to "LE". Current options are:
  - `expand` Expands the correlation matrix with four variables correlated .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 is the method used in the Golino et al. (2020) *Psychological Methods* simulation.
• LE Applies the leading eigenvalue algorithm (`cluster_leading_eigen`) on the empirical correlation matrix. If the number of dimensions is 1, then the leading eigenvalue solution is used; otherwise, regular EGA is used. This is the final method used in the Christensen, Garrido, and Golino (2021) simulation.

corr Type of correlation matrix to compute. The default uses `cor_auto`. Current options are:
  • `cor_auto` Computes the correlation matrix using the `cor_auto` function from `qgraph`.
  • `pearson` Computes Pearson’s correlation coefficient using the pairwise complete observations via the `cor` function.
  • `spearman` Computes Spearman’s correlation coefficient using the pairwise complete observations via the `cor` function.

model Character. A string indicating the method to use. Defaults to "glasso". Current options are:
  • "glasso" Estimates the Gaussian graphical model using graphical LASSO with extended Bayesian information criterion to select optimal regularization parameter. See `EBICglasso.qgraph`
  • "TMFG" Estimates a Triangulated Maximally Filtered Graph. See `TMFG`

algorithm A string indicating the algorithm to use or a function from `igraph`. Defaults to "walktrap". Current options are:
  • `walktrap` Computes the Walktrap algorithm using `cluster_walktrap`
  • `leiden` Computes the Leiden algorithm using `cluster_louvain`

algorithm.args List. A list of additional arguments for `cluster_walktrap` or `cluster_leiden`. Options are:
  • `steps` Number of steps used in the Walktrap algorithm. Defaults to `c(3, 8)`
  • `leiden` Resolution parameter used in the Leiden algorithm. Defaults to `seq(0, 2, .001)`. Higher values lead to smaller communities, lower values lead to larger communities

Value

Returns a list containing:

EGA The EGA output for the best fitting model
steps The number of steps used in the best fitting model from the `cluster_walktrap` algorithm
resolution_parameter The resolution parameter used in the best fitting model from the `cluster_leiden` algorithm
EntropyFit The tefi Index for the unique solutions given the range of steps (vector names represent the number of steps)
Lowest.EntropyFit The lowest value for the tefi Index
Author(s)
Hudson Golino <hfg9s at virginia.edu> and Alexander P. Christensen <alexpaulchristensen@gmail.com>

References

# Entropy fit measures

# Simulation for EGA.fit

# Leiden algorithm

# Walktrap algorithm

See Also

bootEGA to investigate the stability of EGA’s estimation via bootstrap, EGA to estimate the number of dimensions of an instrument using EGA, and CFA to verify the fit of the structure suggested by EGA using confirmatory factor analysis.

Examples

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

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

# Estimate optimal EGA
fit.wmt <- EGA.fit(data = wmt)

# Plot optimal fit
plot(fit.wmt$EGA)

# Estimate CFAs
cfa.ega <- CFA(ega.wmt, estimator = "WLSMV", data = wmt)
cfa.fit <- CFA(fit.wmt$EGA, estimator = "WLSMV", data = wmt)

# Compare CFAs
lavaan::lavTestLRT(
```

cfa.ega$fit, cfa.fit$fit,
method = "satorra.bentler.2001"
)

ega.wmt

EGA WMT-2 Data

Description

EGA Network of wmt2Data

An EGA using the "glasso" model of the Wiener Matrizen-Test 2 (WMT-2)

Usage

data(ega.wmt)

data(ega.wmt)

Format

A 17 x 17 adjacency matrix
A 17 x 17 adjacency matrix

Details

An EGA using the "glasso" model of the Wiener Matrizen-Test 2 (WMT-2)

Examples

data("ega.wmt")

data("ega.wmt")

---

Embed

Time-delay Embedding

Description

Reorganizes an individual’s 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

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

### Arguments

- **x**: Vector. An observed time series to be reorganized into a time-delayed embedded matrix.
- **E**: Integer. Number of embedded dimensions or the number of observations to be used. For example, an "E = 5" will generate a matrix with five columns, meaning that five consecutive observations are used to create each row of the embedded matrix.
- **tau**: Integer. Number of observations to offset successive embeddings. A tau of one uses adjacent observations. Default is "tau = 1".

### Value

Returns a matrix containing the embedded matrix.

### Author(s)

Pascal Deboeck <pascal.deboeck at psych.utah.edu>

### References


### Examples

```r
# A time series with 8 time points
tseries <- 49:56
embed.tseries <- Embed(tseries, E = 4, 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 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@virginia.edu>, Alexander P. Christensen <alexpaulchristensen@gmail.com> and Robert Moulder <rgm4fd@virginia.edu>

References


See Also

EGA to estimate the number of dimensions of an instrument using EGA and CFA to verify the fit of the structure suggested by EGA using confirmatory factor analysis.

Examples

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

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

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

**Ergodicity Information Index**

**Description**

Computes the Ergodicity Information Index

**Usage**

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

**Arguments**

- `dynEGA.object`: A `dynEGA.ind.pop` object
- `use`: Character. 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 = "weighted"`. 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 network. 0 = edge absent and 1 = edge present
  - "weighted" Calculates the algorithm complexity using the weights of the network.

**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>
**Examples**

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

# Dynamic EGA individual and population structure
dyn.ega1 <- dynEGA.ind.pop(
  data = sim.dynEGA, n.embed = 5, tau = 1,
  delta = 1, id = 21, use.derivatives = 1,
  ncores = 2, corr = "pearson"
)

# Compute empirical ergodicity information index
eii <- ergoInfo(
  dynEGA.object = dyn.ega1,
  use = "weighted"
)
```

---

**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**

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

**Arguments**

- `x` Vector. An observed time series.
- `n.embed` Integer. Number of embedded dimensions (the number of observations to be used in the `Embed` function).
- `tau` Integer. Number of observations to offset successive embeddings in the `Embed` function. A tau of one uses adjacent observations. Default is "tau = 1".
- `delta` Integer. The time between successive observations in the time series. Default is "delta = 1".
- `order` Integer. 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@virginia.edu>

References


Examples

```r
# 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 `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

Usage

```r
hierEGA(
  data,
  scores = c("factor", "network"),
  consensus.iter = 1000,
  consensus.method = c("highest_modularity", "most_common", "iterative", "lowest_tefi"),
  uni.method = c("expand", "LE", "louvain"),
  corr = c("cor_auto", "pearson", "spearman"),
)```
model = c("glasso", "TMFG"),
model.args = list(),
algorithm = c("walktrap", "leiden", "louvain"),
algorithm.args = list(),
plot.EGA = TRUE,
plot.args = list()
)

Arguments

data Matrix or data frame. Variables (down columns) only. Does not accept correlation matrices

scores Character. 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 are assumed to be correlated using the "oblimin" rotation. NOTE: Factor scores use the number of communities from EGA. Estimated factor may not align with these communities. The plots using factor scores with have higher order factors that may not completely map onto the lower order communities. Look at the $hierarchical$higher_order$lower_loadings to determine the composition of the lower order factors.

By default, both factor and network scores are computed and stored in the output. The selected option only appears in the main output ($hierarchical)

consensus.iter Numeric. Number of iterations to perform in consensus clustering (see Lancichinetti & Fortunato, 2012). Defaults to 1000

consensus.method Character. What consensus clustering method should be used? Defaults to "highest_modularity". Current options are:

• highest_modularity Uses the community solution that achieves the highest modularity across iterations
• most_common Uses the community solution that is found the most across iterations
• iterative Identifies the most common community solutions across iterations and determines how often nodes appear in the same community together. A threshold of 0.30 is used to set low proportions to zero. This process repeats iteratively until all nodes have a proportion of 1 in the community solution.
• lowest_tefi Uses the community solution that achieves the lowest tefi across iterations

By default, all consensus.method options are computed and stored in the output. The selected method will be used to plot and appear in the main output ($hierarchical)

uni.method Character. What unidimensionality method should be used? Defaults to "LE". Current options are:

• expand Expands the correlation matrix with four variables correlated .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
is the method used in the Golino et al. (2020) *Psychological Methods* simulation.

- **LE** Applies the Leading Eigenvalue algorithm (*cluster_leading_eigen*) on the empirical correlation matrix. If the number of dimensions is 1, then the Leading Eigenvalue solution is used; otherwise, regular EGA is used. This is the final method used in the Christensen, Garrido, and Golino (2021) simulation.

- **louvain** Applies the Louvain algorithm (*cluster_louvain*) on the empirical correlation matrix using a resolution parameter = 0.95. If the number of dimensions is 1, then the Louvain solution is used; otherwise, regular EGA is used. This method was validated in the Christensen (2022) simulation.

**corr** Type of correlation matrix to compute. The default uses *cor_auto*. Current options are:

- **cor_auto** Computes the correlation matrix using the *cor_auto* function from *qgraph*.
- **pearson** Computes Pearson’s correlation coefficient using the pairwise complete observations via the *cor* function.
- **spearman** Computes Spearman’s correlation coefficient using the pairwise complete observations via the *cor* function.

**model** Character. A string indicating the method to use. Defaults to "glasso". Current options are:

- **glasso** Estimates the Gaussian graphical model using graphical LASSO with extended Bayesian information criterion to select optimal regularization parameter
- **TMFG** Estimates a Triangulated Maximally Filtered Graph

**model.args** List. A list of additional arguments for *EBICglasso.qgraph* or *TMFG*

**algorithm** A string indicating the algorithm to use or a function from *igraph* Defaults to "walktrap". Current options are:

- **walktrap** Computes the Walktrap algorithm using *cluster_walktrap*
- **leiden** Computes the Leiden algorithm using *cluster_leiden*. Defaults to *objective_function = "modularity"
- **louvain** Computes the Louvain algorithm using *cluster_louvain*

**algorithm.args** List. A list of additional arguments for *cluster_walktrap*, *cluster_louvain*, or some other community detection algorithm function (see examples)

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

**plot.args** List. A list of additional arguments for the network plot. See *ggnet2* for full list of arguments:

- **vsize** Size of the nodes. Defaults to 6.
- **label.size** Size of the labels. Defaults to 5.
- **alpha** The level of transparency of the nodes, which might be a single value or a vector of values. Defaults to 0.7.
- **edge.alpha** The level of transparency of the edges, which might be a single value or a vector of values. Defaults to 0.4.
• `legend.names` A vector with names for each dimension
• `color.palette` The color palette for the nodes. For custom colors, enter HEX codes for each dimension in a vector. See `color_palette_EGA` for more details and examples

Value

Returns a list of lists containing:

Main Results

`hierarchical` The main results list containing:
• `lower_order` Lower order EGA results for the selected methods
• `higher_order` Higher order EGA results for the selected methods
  If `plot.EGA = TRUE`, then:
  • `lower_plot` Plot of the lower order results
  • `higher_plot` Plot of the higher order results
  • `hier_plot` Plot of the lower and higher order results together, side-by-side

Secondary Results

`lower_ega` A list containing the lower order EGA results. The `$wc` does not contain valid results. Do not use its output.

`lower_wc` A list containing consensus clustering results:
• `highest_modularity` Community memberships based on the highest modularity across the `cluster_louvain` applications
• `most_common` Community memberships based on the most commonly found memberships across the `cluster_louvain` applications
• `iterative` Community memberships based on consensus clustering described by Lancichinetti & Fortunato (2012)
• `lowest_tefi` Community memberships based on the lowest `tefi` across the `cluster_louvain` applications
• `summary_table` A data frame summarizing the unique community solutions across the iterations. Down the columns indicate: number of dimensions (`N_Dimensions`), proportion of times each community solution was identified (`Proportion`), modularity of each community solution (`Modularity`), total entropy fit index of each community solution (`tefi`), and the memberships for each item. Across the rows indicate each unique community solution

`factor_results` A list containing higher order results based on factor scores. A list for each consensus.method is provided with their EGA results

`network_results` A list containing higher order results based on network scores. A list for each consensus.method is provided with their EGA results
**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**

```r
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. Should plot of optimal and hierarchical clusters be output? Defaults to TRUE. Set to FALSE to not plot.

**Examples**

```r
# Obtain example data
data <- optimism

# hierEGA example
opt.hier <- hierEGA(
  data = optimism,
  algorithm = "louvain",
  plot.EGA = FALSE # no plots for CRAN check
)
```

**References**

**intelligenceBattery**  

**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 Van-derbilt.Edu>

**Examples**

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

# Dynamic EGA individual and population structure
dyn.ega1 <- dynEGA.ind.pop(
  data = sim.dynEGA, n.embed = 5, tau = 1,
  delta = 1, id = 21, use.derivatives = 1,
  ncores = 2, corr = "pearson"
)

# Perform information-theoretic clustering
clust1 <- infoCluster(
  dynEGA.object = dyn.ega1,
  plot.cluster = FALSE  # No plot for CRAN checks
)
```

---

**intelligenceBattery**  

**Intelligence Data**

**Description**

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

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

**Usage**

```r
data(intelligenceBattery)
```

```r
data(intelligenceBattery)
```
**invariance**

**Format**
- A 1185x125 response matrix
- A 1185x125 response matrix

**Examples**
```r
data("intelligenceBattery")
data("intelligenceBattery")
```

---

**invariance**

**Measurement Invariance of EGA Structure**

**Description**
Estimates metric invariance of EGA or specified structure

**Usage**
```r
invariance(
  data,
  groups,
  memberships = NULL,
  type = c("loadings"),
  iter = 500,
  ncores,
  ...
)
```

**Arguments**
- **data**
  - Matrix or data frame. Variables to be used in the analysis
- **groups**
  - Vector. Group membership corresponding to each case in data
- **memberships**
  - Vector. Node membership for each community or factor. Defaults to NULL. When NULL, EGA is used to compute node memberships
- **type**
  - Character. Type of measurement invariance to estimate. Only includes "loadings" at the moment
- **iter**
  - Numeric. Number of iterations to perform for the permutation. Defaults to 500
- **ncores**
  - Numeric. Number of cores to use in computing results. Defaults to `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 use the following code: `parallel::detectCores()`
- **...**
  - Arguments passed to EGA
**Value**

Returns a list containing:

- **memberships**: Original memberships provided in `memberships` or from `EGA` if NULL
- **EGA**: Original `EGA` results for the sample
- **groups**
  - `EGA` EGA results for each group
  - `loadings` Network loadings for each group
  - `loadingsDifference` Difference between the dominant loadings of each group
- **permutation**
  - `groups` Permutated groups across iterations
  - `loadings` Loadings 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>, Alexander P. Christensen <alexpaulchristensen@gmail.com>, and Hudson F. Golino <hfg9s at virginia.edu>

**Examples**

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

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

# Measurement invariance
results <- invariance(wmt, groups, ncores = 2)
```

**Description**

Based on the `bootEGA` results, this function computes and plots the number of times an item (variable) is estimated in the same factor/dimension as originally estimated by `EGA` (`item.replication`). The output also contains each item’s replication frequency (i.e., proportion of bootstraps that an item appeared in each dimension; `item.dim.rep`) as well as the average network loading for each item in each dimension (`item.loadings`).

**Usage**

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

- **bootega.obj**  A `bootEGA` object
- **IS.plot**  Should the plot be produced for `item.replication`? If `TRUE`, then a plot for the `item.replication` output will be produced. Defaults to `TRUE`
- **structure**  User specified dimensionality structure.
- **...**  Additional arguments. Used for deprecated arguments from previous versions of `itemStability`

**Value**

Returns a list containing:

- **membership**  A list containing:
  - `empirical`  The empirical memberships from the empirical `EGA` result
  - `unique`  The unique dimensions from the empirical `EGA` result
  - `bootstrap`  The memberships from the replicate samples in the `bootEGA` results

- **item.stability**  A list containing:
  - `empirical.dimensions`  The proportion of times each item replicated within the empirical `EGA` defined dimension. This EGA result is defined using the input from `bootEGA`
  - `all.dimensions`  The proportion of times each item replicated in each of the empirical `EGA` defined dimensions. This EGA result is defined using the input from `bootEGA`

- **plot**  A plot of the number of times each item replicated within the empirical `EGA` defined dimension.

- **mean.loadings**  Matrix of the average standardized network loading (computed using `net.loads`) for each item in each dimension

**Author(s)**

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

**References**


**See Also**

EGA to estimate the number of dimensions of an instrument using EGA and CFA to verify the fit of the structure suggested by EGA using confirmatory factor analysis.
Examples

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

# Standard EGA example
boot.wmt <- bootEGA(
  data = wmt, iter = 100, # recommended 500
  plot.typicalStructure = FALSE, # No plot for CRAN checks
  type = "parametric", ncores = 2
)

# Standard item stability
wmt.is <- itemStability(
  boot.wmt,
  IS.plot = FALSE # NO plot for CRAN checks
)

# Produce Methods section
methods.section(
  boot.wmt,
  stats = "itemStability"
)

# EGA fit example
boot.wmt.fit <- bootEGA(
  data = wmt, iter = 100, # recommended 500
  EGA.type = "EGA.fit",
  plot.typicalStructure = FALSE, # No plot for CRAN checks
  type = "parametric", ncores = 2
)

# EGA fit item stability
wmt.is.fit <- itemStability(
  boot.wmt.fit,
  IS.plot = FALSE # NO plot for CRAN checks
)

# Hierarchical EGA example
boot.wmt.hier <- bootEGA(
  data = wmt, iter = 100, # recommended 500
  EGA.type = "hierEGA",
  plot.typicalStructure = FALSE, # No plot for CRAN checks
  type = "parametric", ncores = 2
)

# Hierarchical EGA item stability
wmt.is.hier <- itemStability(
  boot.wmt.hier,
  IS.plot = FALSE # NO plot for CRAN checks
)

# Random-intercept EGA example
```
boot.wmt.ri <- bootEGA(
  data = wmt, iter = 100, # recommended 500
  EGA.type = "riEGA",
  plot.typicalStructure = FALSE, # No plot for CRAN checks
  type = "parametric", ncores = 2
)

# Random-intercept EGA item stability
wmt.is.ri <- itemStability(
  boot.wmt.ri,
  IS.plot = FALSE # NO plot for CRAN checks
)

---

**Jensen-Shannon Distance**

**Description**

Computes the Jensen-Shannon Distance between two networks

**Usage**

```r
gsd(network1, network2, method = c("kld", "spectral"))
```

**Arguments**

- `network1`: Matrix or data frame. Network to be compared
- `network2`: Matrix or data frame. Second network to be compared
- `method`: Character. Method to compute Jensen-Shannon Distance. Defaults to "spectral". Options:
  - "kld" Uses Kullback-Leibler Divergence
  - "spectral" Uses eigenvalues of combinatorial Laplacian matrix to compute Von Neumann entropy

**Value**

Returns Jensen-Shannon Distance

**Author(s)**

Hudson Golino <hfg9s at virginia.edu> & Alexander P. Christensen <alexander.christensen at Van-derbilt.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) # 0.1618195

# Spectral JSS (similarity)
1 - jsd(glas1, glas2) # 0.8381805

# Jensen-Shannon Divergence
jsd(glas1, glas2, method = "kld") # 0.1923636

---

**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,
  iter = 100,
  dynamic = FALSE,
  dynamic.args = list(n.embed = 4, tau = 1, delta = 1, use.derivatives = 1)
)
Arguments

data Matrix or data frame. A data frame with the variables to be used in the test or a correlation matrix. If the data used is a correlation matrix, the argument n will need to be specified

n Integer. Sample size (if the data provided is a correlation matrix)

iter Integer. Number of replicate samples to be drawn from a multivariate normal distribution (uses \texttt{mvtnorm::mvrnorm}). Defaults to 100

dynamic Boolean. Is the dataset a time series where rows are time points and columns are variables? Defaults to \texttt{FALSE}.

dynamic.args List. Arguments to be used in \texttt{dynEGA}. Defaults:
  - \texttt{n.embed} Number of embeddings: 4
  - \texttt{tau} Lag: 1
  - \texttt{delta} Delta: 1
  - \texttt{use.derivatives} Derivatives: 1

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


Examples

# Compute LCT
## Network model
LCT(data = wmt2[,7:24])

## Factor model
LCT(data = psychTools::bfi[,1:25])

# Dynamic LCT
LCT(sim.dynEGA[sim.dynEGA$ID == 1,1:20], dynamic = TRUE)
louvain community detection algorithm

Description

Computes the Louvain community detection algorithm (Blondel et al., 2008)

Usage

louvain(A, method = c("modularity", "tefi"), resolution = 1, corr = NULL)

Arguments

- **A**: Matrix or data frame. A network adjacency matrix
- **method**: Character. Whether modularity or tefi should be used to optimize communities. Defaults to "modularity"
- **resolution**: Numeric. Resolution parameter for computing modularity. Defaults to 1. Values smaller than 1 favor larger communities; values larger than 1 favor smaller communities
- **corr**: Matrix or data frame. Correlation matrix to be used when method = "tefi"

Details

This version was adapted from the Matlab code available here: https://perso.uclouvain.be/vincent.blondel/research/louvain.html. The code was adjusted to mirror the results of cluster_louvain. The Louvain algorithm’s results can vary depending on node ordering. In this version, nodes are not shuffled so that consistent results can be achieved with the same node ordering. Results from cluster_louvain will shuffle nodes within the function and therefore will sometimes produce similar results and sometimes produce slightly different results. This version is based all in R and therefore is slower than the version in igraph.

Value

Returns a list containing:

- **wc**: A matrix of lower to higher order community membership detected in the network
- **modularity**: A vector of modularity values corresponding the rows of the wc matrix

Author(s)

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

References

Examples

# Load data
dep <- depression[,24:44]

# Estimate correlations
corr <- qgraph::cor_auto(dep)

# Estimate network
net <- EBICglasso.qgraph(corr, n = nrow(dep))

# Estimate communities using modularity
louvain(net, method = "modularity")

# Estimate communities using tefi
louvain(net, method = "tefi", corr = corr)

mctest.ergoInfo

Monte-Carlo Test for the Ergodicity Information Index

Description

Computes a Monte-Carlo Test for the Ergodicity Information Index, comparing the empirical Ergodicity Information index to values obtained in a Monte-Carlo simulation in which all individuals have a similar latent structure. The p-values in the Monte-Carlo test can be calculated as 
\( (\text{sum}(\text{EII} \geq \text{MC.EII}) + 1)/(\text{iter} + 1) \)
and as 
\( (\text{sum}(\text{EII} \leq \text{MC.EII}) + 1)/(\text{iter} + 1) \), where EII is the empirical Ergodicity Information Index, MC.EII is the values of the Ergodicity Information Index obtained in the simulation, and iter is the number of random samples generated in the simulation. The two-sided p-value is computed as two times the lowest p-value. In the Monte-Carlo Test for the Ergodicity Information Index, the null hypothesis is that the empirical value of EII is equal to the Monte-Carlo value of EII obtained in multiple individuals with a similar latent structure. Small values of p indicate that is very unlikely to obtain an EII as large as the one obtained in the empirical sample if the null hypothesis is true, thus there is convincing evidence that the empirical Ergodicity Information Index is different than it could be expected if all individuals had a similar latent structure, conditioned on the parameters used to simulate the data.

Usage

mctest.ergoInfo(
  iter,
  N,
  EII,
  use,
  variab,
  timep,
  nfact,
  error,
  dfm,
loadings,
autoreg,
crossreg,
var.shock,
cov.shock,
embed,
tau,
delta,
derivatives,
model,
model.args = list(),
algorithm = c("walktrap", "louvain"),
algorithm.args = list(),
corr,
ncores,
...)

Arguments

iter  Numeric integer. Number of random samples to generate in the Monte-Carlo simulation. At least 500 is recommended

N  Numeric integer. Number of individuals to simulate data from, using the simDFM function.

EII  Numeric. Empirical Ergodicity Information Index obtained via the ergoInfo function.

use  Character. A string indicating what network element will be used to compute the algorithm complexity in the ergoInfo function, the list of edges or the weights of the network. Defaults to use = "edge.list". Current options are:

- edge.list Calculates the algorithm complexity using the list of edges.
- weights Calculates the algorithm complexity using the weights of the network.

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. Defaults to "DAFS". 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. Default is "autoreg = 0.8".
crossreg Magnitude of the cross-regression coefficients. Default is "crossreg = 0.1".
var.shock Magnitude of the random shock variance. Default is "var.shock = 0.18".
cov.shock Magnitude of the random shock covariance. Default is "cov.shock = 0.36".
embed    Integer. Number of embedded dimensions (the number of observations to be used in the Embed function). For example, an "embed = 5" will use five observations to estimate a single derivative. Defaults to embed = 5.
tau      Integer. Number of observations to offset successive embeddings in the Embed function. A tau of one uses adjacent observations. Default is "tau = 1".
delta    Integer. The time between successive observations in the time series. Default is "delta = 1".
derivatives Integer. The order of the derivative to be used in the EGA procedure. Default to 1.
model    Character. A string indicating the method to use. Defaults to glasso. Current options are:
          • glasso Estimates the Gaussian graphical model using graphical LASSO with extended Bayesian information criterion to select optimal regularization parameter. This is the default method
          • TMFG Estimates a Triangulated Maximally Filtered Graph
model.args List. A list of additional arguments for EBICglasso.qgraph or TMFG
algorithm A string indicating the algorithm to use or a function from igraph Current options are:
          • walktrap Computes the Walktrap algorithm using cluster_walktrap
          • louvain Computes the Walktrap algorithm using cluster_louvain
algorithm.args List. A list of additional arguments for cluster_walktrap, cluster_louvain, or some other community detection algorithm function (see examples)
corr     Type of correlation matrix to compute. The default uses cor_auto. Current options are:
          • cor_auto Computes the correlation matrix using the cor_auto function from qgraph.
          • pearson Computes Pearson’s correlation coefficient using the pairwise complete observations via the cor function.
          • spearman Computes Spearman’s correlation coefficient using the pairwise complete observations via the cor function.
ncores   Numeric. Number of cores to use in computing results. Defaults to parallel::detectCores() / 2 or half of your computer’s processing power. Set to 1 to not use parallel computing. Recommended to use maximum number of cores minus one. If you’re unsure how many cores your computer has, then use the following code: parallel::detectCores()
... Additional arguments. Used for deprecated arguments from previous versions of EGA
Value

Returns a list containing:

- `mc.ergoInfo` The values of the Ergodicity Information Index obtained in the Monte-Carlo Simulation
- `p.value.twosided` The p-value of the Monte-Carlo test for the Ergodicity Information Index. The null hypothesis is that the empirical Ergodicity Information index is equal to the expected value of the EII if the all individuals had similar latent structures.
- `effect` Indicates whether the empirical EII is greater or less than the Monte-Carlo obtained EII.
- `plot.dist` Histogram of the bootstrapped ergodicity information index

Author(s)

Hudson Golino <hfg9s@virginia.edu>

Examples

```r
## Not run:
\donttest{
dyn1 <- dynEGA.ind.pop(data = sim.dynEGA, n.embed = 5, tau = 1,
  delta = 1, id = 21, group = 22, use.derivatives = 1,
  model = "glasso", ncores = 2, corr = "pearson")

eii1 <- ergoInfo(data = dyn1)$EII

dist.ergoinfo <- mctest.ergoInfo(iter = 10, N = 10, EII = eii1,
  variab = 4,
  timep = 100, nfact = 2, error = 0.05, dfm = "DAFS", loadings = 0.55, autoreg = 0.8,
  crossreg = 0.1, var.shock = 0.18, cov.shock = 0.36, embed = 5, tau=1, delta=1, derivatives=1,
  model = "glasso", ncores = 2, corr = "pearson")
}
## End(Not run)
```

Description

This function accepts EGA objects and generates a Methods section for your analysis. The output is an HTML page containing the descriptions of the methods and parameters as well as a Reference section for appropriate citation.
Usage

methods.section(
  ...,  
  stats = c("net.loads", "net.scores", "dimensionStability", "itemStability")
)

Arguments

... EGAnet objects. Available methods (more methods will be added soon!):

- EGA Exploratory graph analysis
- bootEGA Bootstrap exploratory graph analysis
- UVA Unique variable analysis

stats Methods section for statistics in EGAnet. Multiple statistics can be input. Available statistics:

- net.loads Network loadings. Requires EGA object to be input
- net.scores Network scores. Requires EGA object to be input
- dimensionStability Structural consistency. Requires bootEGA object to be input
- itemStability Item stability. Requires bootEGA object to be input

Value

Automated HTML Methods section in your default browser

Examples

# Estimate EGA
## plot.type = "qgraph" used for CRAN checks
## plot.type = "GGally" is the default
ega.wmt <- EGA(data = wmt2[,7:24], plot.type = "qgraph")

# EGA Methods section
if(interactive()){
  methods.section(ega.wmt)
}

# Estimate standardized network loadings
wmt.loads <- net.loads(ega.wmt)$std

# EGA Methods section with network loadings
if(interactive()){
  methods.section(ega.wmt, stats = "net.loads")
}

## Not run: # bootEGA example
## plot.type = "qgraph" used for CRAN checks
## plot.type = "GGally" is the default
boot.wmt <- bootEGA(data = wmt2[,7:24], iter = 500, plot.type = "qgraph", type = "parametric", ncores = 2)
## End(Not run)

EGA and bootEGA Methods section

```r
if(interactive()){
  methods.section(ega.wmt, boot.wmt)
}
```

# Estimate structural consistency
```r
sc.wmt <- dimensionStability(boot.wmt)
```

EGA and bootEGA Methods section with structural consistency and item stability

```r
if(interactive()){
  methods.section(boot.wmt, stats = c("dimensionStability", "itemStability"))
}
```

EGA with network loadings and

EGA and bootEGA Methods section with structural consistency and item stability

```r
if(interactive()){
  methods.section(ega.wmt, boot.wmt, stats = c("net.loads", "dimensionStability", "itemStability"))
}
```

---

### net.loads: Network Loadings

**Description**

Computes the between- and within-community strength of each item for each community. This function uses the `comcat` and `stable` functions to calculate the between- and within-community strength of each item, respectively.

**Usage**

```r
net.loads(A, wc, pos.manifold = FALSE, min.load = 0)
```

**Arguments**

- **A**: Matrix, data frame, or EGA object. A network adjacency matrix
- **wc**: Numeric or character vector. A vector of community assignments. If input into `A` is an EGA object, then `wc` is automatically detected
- **pos.manifold**: Boolean. Should a positive manifold be applied (i.e., should all dimensions be positively correlated)? Defaults to `FALSE`. Set to `TRUE` for a positive manifold
- **min.load**: Numeric. Sets the minimum loading allowed in the standardized network loading matrix. Values equal or greater than the minimum loading are kept in the output. Values less than the minimum loading are removed. This matrix can be viewed using `print()` or `summary()`. Defaults to 0
Details

Simulation studies have demonstrated that a node’s strength centrality is roughly equivalent to factor loadings (Christensen, Golino, & Silvia, 2019; Hallquist, Wright, & Molenaar, in press). Hallquist and colleagues (in press) 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). For more details, type vignette("Network_Scores")

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
- minLoad: The minimum loading to appear in summary of network loadings. Use print() or summary() to view

Author(s)

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

References


Examples

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

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

# Network loadings
net.loads(ega.wmt)

# Produce Methods section
methods.section(
    ega.wmt,
    stats = "net.loads"
)

---

### net.scores

#### Network Scores

**Description**

This function computes network scores computed based on each node's strength within each community (i.e., factor) in the network (see net.loads). These values are used as network "factor loadings" for the weights of each item. Notably, network analysis allows nodes to contribution to more than one community. These loadings are considered in the network scores. In addition, if the construct is a hierarchy (e.g., personality questionnaire; items in facet scales in a trait domain), then an overall score can be computed (see argument global). An important difference is that the network scores account for cross-loadings in their estimation of scores.

**Usage**

```r
net.scores(data, A, wc, global = FALSE, impute, ...)
```

**Arguments**

- **data** Matrix or data frame. Must be a dataset
- **A** Matrix, data frame, or EGA object. An adjacency matrix of network data
- **wc** Numeric. A vector of community assignments. Not necessary if an EGA object is input for argument A
- **global** Boolean. Should general network loadings be computed in scores? Defaults to FALSE. If there is more than one dimension and there is theoretically one global dimension, then general loadings of the dimensions onto the global dimension can be included in the weighted scores
- **impute** Character. In the presence of missing data, imputation can be implemented. Currently, three options are available:
  - **none** No imputation is performed. This is the default.
  - **mean** The "mean" value of the columns are used to replace the missing data.
  - **median** The "median" value of the columns are used to replace the missing data.
- **...** Additional arguments for EGA

**Details**

For more details, type `vignette("Network_Scores")`
Value

Returns a list containing:

unstd.scores  The unstandardized network scores for each participant and community (including the overall score)
std.scores    The standardized network scores for each participant and community (including the overall score)
commCor       Partial correlations between the specified or identified communities
loads         Standardized network loadings for each item in each dimension (computed using net.loads)

Author(s)

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

References


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)

# Produce Methods section
methods.section(
  ega.wmt,
  stats = "net.scores"
)
network.descriptives  

**Description**

Computes descriptive statistics for network models

**Usage**

```
network.descriptives(network)
```

**Arguments**

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>network</td>
<td>Matrix, data frame, qgraph, or EGA object</td>
</tr>
</tbody>
</table>

**Value**

Numeric vector including:

- **Mean_weight**: The average of the edge weights in the network
- **SD_weight**: The standard deviation of the edge weights in the network
- **Min_weight**: The minimum of the edge weights in the network
- **Max_weight**: The maximum of the edge weights in the network
- **Density**: The density of the network
- **ASPL**: The average shortest path length (ASPL) of the network (computed as unweighted)
- **CC**: The clustering coefficient (CC) of the network (computed as unweighted)
- **swn.rand**: Small-worldness measure based on random networks:
  
  \[
  \text{swn.rand} = \frac{\text{ASPL}_{\text{random}}}{\text{ASPL}} / \frac{\text{CC}_{\text{random}}}{\text{CC}}
  \]
  
  \(\text{swn.rand} > 1\) suggests the network is small-world
- **swn.HG**: Small-worldness measure based on Humphries & Gurney (2008):
  
  \[
  \text{swn.HG} = \frac{\text{transitivity}}{\text{transitivity}_{\text{random}}} / \frac{\text{ASPL}_{\text{random}}}{\text{ASPL}}
  \]
  
  \(\text{swn.HG} > 1\) suggests the network is small-world
- **swn.TJHBL**: Small-worldness measure based on Telesford, Joyce, Hayasaka, Burdette, & Laurienti (2011):
  
  \[
  \text{swn.TJHBL} = \frac{\text{ASPL}_{\text{random}}}{\text{ASPL}} - \frac{\text{CC}}{\text{CC}_{\text{lattice}}}
  \]
  
  \(\text{swn.TJHBL} \text{ near } 0\) suggests the network is small-world, positive values suggest more random network characteristics, negative values suggest more lattice network characteristics
- **scale-free_R-sq**: The R-squared fit of whether the degree distribution follows the power-law (many small degrees, few large degrees)
optimism

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

References

# swn.HG

# swn.TJHBL
Telesford, Q. K., Joyce, K. E., Hayasaka, S., Burdette, J. H., & Laurienti, P. J. (2011). The ubiquity

# scale-free_R-sq
analysis. *BMC Bioinformatics*, 9, 559

Examples

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

# EGA example
ego.wmt <- EGA(
  data = wmt,
  plot.EGA = FALSE # No plot for CRAN
)

# Compute descriptives
network.descriptives(ego.wmt)

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).

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)

data(optimism)
Format

A 282x10 response matrix
A 282x10 response matrix

References


Examples

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

```r
plot(x, title = "", plot.args = list(), produce = TRUE, ...)
```

```r
## S3 method for class 'bootEGA'
plot(x, title = "", plot.args = list(), produce = TRUE, ...)
```

```r
## S3 method for class 'CFA'
plot(x, layout = "spring", vsize = 6, ...)
```

```r
## S3 method for class 'dynEGA'
plot(x, title = "", plot.args = list(), produce = TRUE, ...)
```

```r
## S3 method for class 'dynEGA.Groups'
plot(x, ncol, nrow, title = "", plot.args = list(), produce = TRUE, ...)
```

```r
## S3 method for class 'dynEGA.Individuals'
plot(x, title = "", id = NULL,
```
plot.args = list(), produce = TRUE, ...)  

## S3 method for class 'EGA'
plot(x, title = "", plot.args = list(), produce = TRUE, ...)  

Arguments

x Object from EGAnet package  

title Character. Title of the plot. Defaults to ""  

plot.args List. A list of additional arguments for the network plot. See ggnet2 for full list of arguments:

- vsize Size of the nodes. Defaults to 6.
- label.size Size of the labels. Defaults to 5.
- alpha The level of transparency of the nodes, which might be a single value or a vector of values. Defaults to 0.7.
- edge.alpha The level of transparency of the edges, which might be a single value or a vector of values. Defaults to 0.4.
- legend.names A vector with names for each dimension
- color.palette The color palette for the nodes. For custom colors, enter HEX codes for each dimension in a vector. See color_palette_EGA for more details and examples

produce Boolean. This argument is used internally. Should plot be produced? Defaults to TRUE  

... Arguments passed on to  

- semPaths Functions: CFA

vsize Numeric. Size of vertices in CFA plots. Defaults to 6  

layout Character. Layout of plot (see semPaths). Defaults to "spring"  

ncol Numeric. Number of columns  

nrow Numeric. Number of rows  

id Numeric. An integer or character indicating the ID of the individual to plot

Value

Plots of EGAnet object

Author(s)

Hudson Golino <hfg9s at virginia.edu> and Alexander P. Christensen <alexpaulchristensen@gmail.com>
**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**
```r
data(prime.num)
data(prime.num)
```

**Format**
A 1185x24 response matrix

**Examples**
```r
data("prime.num")
data("prime.num")
```

---

**prints**  
*S3Methods for Printing*

**Description**
Prints for EGAnet objects

**Usage**
```r
## S3 method for class 'dynEGA'
print(x, ...)

## S3 method for class 'dynEGA.Groups'
print(x, ...)

## S3 method for class 'dynEGA.Individuals'
print(x, ...)
```
residualEGA

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

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

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

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

Arguments

x Object from EGAnet package
...

Additional arguments

Value

Prints EGAnet object

Author(s)

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

---

residualEGA Residualized EGA

Description

residualEGA Estimates the number of dimensions after controlling for wording effects. EGA is applied in the residual of a random intercept item factor model (RIIFA) with one method factor and one substantive factor.

Usage

residualEGA(data, manifests, lat, negative.items)

Arguments

data Matrix or data frame. Includes the variables to be used in the residualEGA analysis

manifests Character vector. Vector indicating the names of the variables (items) to be used in the analysis.

lat Numeric integer. Number of latent factors to be estimated. Only one substantive latent factor is recommended in the current version of the function.

negative.items Numeric vector A numeric vector indicating the column of the negative items.
Value
Returns a list containing:

openMx.model  OpenMX model
openMx.result OpenMX results
openMx.std.par OpenMX standardized parameters
ResidualMatrix Residual matrix
EGA.Residuals Results of the residualized EGA
Fit  Fit metrics of the network structure, calculated using the ggmfit function of the qgraph package
WordLoads Loadings of the wording effects

Author(s)
Hudson F. Golino <hfg9s@virginia.edu> and Robert Moulder <rgm4fd@virginia.edu>

See Also
EGA to estimate the number of dimensions of an instrument using EGA and CFA to verify the fit of the structure suggested by EGA using confirmatory factor analysis.

Examples

```r
data <- optimism

## Not run:
# resEGA example
opt.res <- residualEGA(data = data, manifests = colnames(optimism), lat = 1, negative.items = c(3,7,9))

# Fit:
opt.res$Fit

## End(Not run)
```

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,
  uni.method = c("expand", "LE", "louvain"),
  corr = c("cor_auto", "pearson", "spearman"),
  model = c("glasso", "TMFG"),
  model.args = list(),
  algorithm = c("walktrap", "louvain"),
  algorithm.args = list(),
  consensus.iter = 100,
  consensus.method = c("highest_modularity", "most_common", "iterative", "lowest_tefi"),
  plot.EGA = TRUE,
  plot.args = list(),
  estimator = c("auto", "WLSMV", "MLR"),
  lavaan.args = list()
)

Arguments

data Matrix or data frame. Variables (down columns) or correlation matrix. If the
  input is a correlation matrix, then argument n (number of cases) is required. Variables
  MUST be unrecoded – reversed items should remain reversed

n Integer. Sample size if data provided is a correlation matrix

uni.method Character. What unidimensionality method should be used? Defaults to "louvain". Current options are:

  • expand Expands the correlation matrix with four variables correlated .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 is the method used in the Golino et al. (2020) Psychological Methods simulation.

  • LE Applies the Leading Eigenvalue algorithm (cluster_leading_eigen) on the empirical correlation matrix. If the number of dimensions is 1, then the Leading Eigenvalue solution is used; otherwise, regular EGA is used. This is the final method used in the Christensen, Garrido, and Golino (2021) simulation.

  • louvain Applies the Louvain algorithm (cluster_louvain) on the empirical correlation matrix using a resolution parameter = 0.95. If the number of dimensions is 1, then the Louvain solution is used; otherwise, regular EGA is used. This method was validated in the Christensen (2022) simulation.

corr Type of correlation matrix to compute. The default uses cor_auto. Current options are:

  • cor_auto Computes the correlation matrix using the cor_auto function from qgraph.

  • pearson Computes Pearson’s correlation coefficient using the pairwise complete observations via the cor function.
• spearman Computes Spearman’s correlation coefficient using the pairwise complete observations via the cor function.

model Character. A string indicating the method to use. Defaults to "glasso". Current options are:
  • glasso Estimates the Gaussian graphical model using graphical LASSO with extended Bayesian information criterion to select optimal regularization parameter
  • TMFG Estimates a Triangulated Maximally Filtered Graph

model.args List. A list of additional arguments for EBICglasso.qgraph or TMFG

algorithm A string indicating the algorithm to use or a function from igraph. Defaults to "walktrap". Current options are:
  • walktrap Computes the Walktrap algorithm using cluster_walktrap
  • louvain Computes the Louvain algorithm using cluster_louvain

algorithm.args List. A list of additional arguments for cluster_walktrap, cluster_louvain, or some other community detection algorithm function (see examples)

consensus.iter Numeric. Number of iterations to perform in consensus clustering for the Louvain algorithm (see Lancichinetti & Fortunato, 2012). Defaults to 100

consensus.method Character. What consensus clustering method should be used? Defaults to "highest_modularity". Current options are:
  • highest_modularity Uses the community solution that achieves the highest modularity across iterations
  • most_common Uses the community solution that is found the most across iterations
  • iterative Identifies the most common community solutions across iterations and determines how often nodes appear in the same community together. A threshold of 0.30 is used to set low proportions to zero. This process repeats iteratively until all nodes have a proportion of 1 in the community solution.
  • lowest_teffi Uses the community solution that achieves the lowest teffi across iterations

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

plot.args List. A list of additional arguments for the network plot. For plot.type = "qgraph":
  • vsize Size of the nodes. Defaults to 6.
For plot.type = "GGally" (see ggnet2 for full list of arguments):
  • vsize Size of the nodes. Defaults to 6.
  • label.size Size of the labels. Defaults to 5.
  • alpha The level of transparency of the nodes, which might be a single value or a vector of values. Defaults to 0.7.
  • edge.alpha The level of transparency of the edges, which might be a single value or a vector of values. Defaults to 0.4.
• legend.names A vector with names for each dimension
• color.palette The color palette for the nodes. For custom colors, enter HEX codes for each dimension in a vector. See color_palette_EGA for more details and examples
estimator Character. Estimator to use for random-intercept model (see Estimators for more details). Defaults to "auto", which selects "MLR" for continuous data and "WLSMV" for mixed and categorical data. Data are considered continuous data if they have 6 or more categories (see Rhemtulla, Brosseau-Liard, & Savalei, 2012)
lavaan.args List. If reduce.method = "latent", then lavaan's cfa function will be used to create latent variables to reduce variables. Arguments should be input as a list. Some example arguments (see lavOptions for full details)

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

Author(s)
Alejandro Garcia-Pardina <alejandrogp97@gmail.com>, Francisco J. Abad <fjose.abad@uam.es>, Alexander P. Christensen <lexpaulchristensen@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

Examples
# Obtain example data
data <- optimism

# riEGA example
opt.res <- riEGA(data = optimism)
**sim.dynEGA**

*sim.dynEGA Data*

**Description**

A simulated (multivariate time series) data with 20 variables, 200 individual observations, 50 time points per individual and 2 groups of individuals.

**Usage**

```r
data(sim.dynEGA)
```

**Format**

A 10000x22 multivariate time series

**Examples**

```r
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**

```r
simDFM(
  variab,
  timep,
  nfact,
  error,
  dfm = c("DAFS", "RandomWalk"),
)```
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


Examples

```r
## Not run:
\dontrtest{
  # 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 = 0.18,
                  cov.shock = 0.36, burnin = 1000)
}
## End(Not run)
```

### Description

**Summarys** for **EGA** objects

### Usage

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

## S3 method for class 'dynEGA.Groups'
summary(object, ...)

## S3 method for class 'dynEGA.Individuals'
summary(object, ...)

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

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

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

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

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

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

**Arguments**

- **object**: Object from EGAnet package
- **...**: Additional arguments

**Value**

- **summary**: EGAnet object

**Author(s)**

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

**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**

```r
tefi(data, structure)
```

**Arguments**

- **data**: A dataframe or correlation matrix
- **structure**: 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@virginia.edu>, Alexander P. Christensen <alexpaulchristensen@gmail.com>, and Robert Moulder <rgm4fd@virginia.edu>
References

See Also
EGA to estimate the number of dimensions of an instrument using EGA and CFA to verify the fit of the structure suggested by EGA using confirmatory factor analysis.

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

## Not run:
# Estimate EGA model
tei(data = ega.wmt$correlation, structure = ega.wmt$wc)

## End(Not run)
```

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

Usage
```
TMFG(cormat)
```

Arguments
```
cormat   A correlation matrix
```
TMFG

Details

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

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

Value

Returns a list containing:

- \(A\) The filtered adjacency matrix
- separators The separators (3-cliques) in the network
- cliques The cliques (4-cliques) in the network

Author(s)

Alexander Christensen <alexpaulchristensen@gmail.com>

References


Examples

# Pearson's correlation only for CRAN checks
A <- TMFG(cor(wmt2[,7:24]))$A
totalCor  

**Total Correlation**

**Description**
Computes the total correlation of a dataset

**Usage**
```r
totalCor(data)
```

**Arguments**
- `data`  
  Matrix or data frame. Variables to be used in the analysis

**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

**Author(s)**
Hudson F. Golino <hfg9s@virginia.edu>

**References**


**Examples**
```r
# Compute total correlation
totalCor(wmt2[,7:24])
```
totalCorMat

Total Correlation Matrix

Description
Computes the pairwise total correlation for a dataset

Usage
totalCorMat(data)

Arguments
data Matrix or data frame. Variables to be used in the analysis

Value
Returns a square matrix with pairwise total correlations

Author(s)
Hudson F. Golino <hfg9s at virginia.edu>

References


Examples
```r
## Not run:
\donttest{
  # Compute total correlation
  totalCorMat(wmt2[,7:24])
}
## End(Not run)
```
toy.example  Toy Example Data

Description

A simulated dataset with 2 factors, three items per factor and n = 500.

Usage

data(toy.example)

Format

A 500x6 response matrix

Examples

data("toy.example")

UVA  Unique Variable Analysis

Description

Identifies redundant variables in a multivariate dataset using a number of different association methods and types of significance values (see Christensen, Garrido, & Golino, 2020 for more details)

Usage

UVA(
    data,
    n = NULL,
    model = c("glasso", "TMFG"),
    corr = c("cor_auto", "pearson", "spearman"),
    method = c("cor", "pcor", "wTO"),
    type = c("adapt", "alpha", "threshold"),
    sig,
    key = NULL,
    reduce = TRUE,
    auto = TRUE,
    label.latent = FALSE,
    reduce.method = c("latent", "remove", "sum"),
    lavaan.args = list(),
    adhoc = TRUE,
    plot.redundancy = FALSE,
    plot.args = list()
  )
Arguments

data Matrix or data frame. Input can either be data or a correlation matrix

n Numeric. If input in data is a correlation matrix, then sample size is required. Defaults to NULL

model Character. A string indicating the method to use. Current options are:

- glasso Estimates the Gaussian graphical model using graphical LASSO with extended Bayesian information criterion to select optimal regularization parameter. This is the default method
- TMFG Estimates a Triangulated Maximally Filtered Graph

corr Type of correlation matrix to compute. The default uses cor_auto. Current options are:

- cor_auto Computes the correlation matrix using the cor_auto function from qgraph.
- pearson Computes Pearson’s correlation coefficient using the pairwise complete observations via the cor function.
- spearman Computes Spearman’s correlation coefficient using the pairwise complete observations via the cor function.

method Character. Computes weighted topological overlap ("wTO" using EBICglasso), partial correlations ("pcor"), or correlations ("cor") Defaults to "wTO"

type Character. Type of significance. Computes significance using the standard p-value ("alpha"), adaptive alpha p-value (adapt.a), or some threshold “threshold”. Defaults to "threshold"

sig Numeric. p-value for significance of overlap (defaults to .05). Defaults for “threshold” for each method:

- "wTO" .25
- "pcor" .35
- "cor" .50

key Character vector. A vector with variable descriptions that correspond to the order of variables input into data. Defaults to NULL or the column names of data

reduce Boolean. Should redundancy reduction be performed? Defaults to TRUE. Set to FALSE for redundancy analysis only

auto Boolean. Should redundancy reduction be automated? Defaults to TRUE. Set to FALSE for manual selection

label.latent Boolean. Should latent variables be labelled? Defaults to TRUE. Set to FALSE for arbitrary labelling (i.e., "LV")

reduce.method Character. How should data be reduced? Defaults to "latent"

- "latent" Redundant variables will be combined into a latent variable
- "remove" All but one redundant variable will be removed
- "sum" Redundant variables are combined by summing across cases (rows)
lavaan.args List. If reduce.method = "latent", then lavaan's cfa function will be used to create latent variables to reduce variables. Arguments should be input as a list. Some example arguments (see lavOptions for full details):

• estimator Estimator to use for latent variables (see Estimators) for more details. Defaults to "MLR" for continuous data and "WLSMV" for mixed and categorical data. Data are considered continuous data if they have 6 or more categories (see Rhemtulla, Brosseau-Liard, & Savalei, 2012)

• missing How missing data should be handled. Defaults to "fiml"

• std.lv If TRUE, the metric of each latent variable is determined by fixing their (residual) variances to 1.0. If FALSE, the metric of each latent variable is determined by fixing the factor loading of the first indicator to 1.0. If there are multiple groups, std.lv = TRUE and "loadings" is included in the group.label argument, then only the latent variances of the first group will be fixed to 1.0, while the latent variances of other groups are set free. Defaults to TRUE

adhoc Boolean. Should adhoc check of redundancies be performed? Defaults to TRUE. If TRUE, adhoc check will run the redundancy analysis on the reduced variable set to determine if there are any remaining redundancies. This check is performed with the arguments: method = "wTO", type = "threshold", and sig = .20. This check is based on Christensen, Garrido, and Golino’s (2020) simulation where these parameters were found to be the most conservative, demonstrating few false positives and false negatives

plot.redundancy Boolean. Should redundancies be plotted in a network plot? Defaults to FALSE

plot.args List. Arguments to be passed onto ggnet2. Defaults:

• vsize = 6 Changes node size
• alpha = 0.4 Changes transparency
• label.size = 5 Changes label size
• edge.alpha = 0.7 Changes edge transparency

Value

Returns a list:

redundancy A list containing several objects:

• redundant Vectors nested within the list corresponding to redundant nodes with the name of object in the list
• data Original data
• correlation Correlation matrix of original data
• weights Weights determine by weighted topological overlap, partial correlation, or zero-order correlation
• network If method = "wTO", then the network computed following EGA with EBICglasso network estimation
• plot If redundancy.plot = TRUE, then a plot of all redundancies found
• descriptives
basic A vector containing the mean, standard deviation, median, median absolute deviation (MAD), 3 times the MAD, 6 times the MAD, minimum, maximum, and critical value for the overlap measure (i.e., weighted topological overlap, partial correlation, or threshold)

centralTendency A matrix for all (absolute) non-zero values and their respective standard deviation from the mean and median absolute deviation from the median

• method Returns method argument
• type Returns type argument
• distribution If type = "threshold", then distribution that was used to determine significance

reduced If reduce = TRUE, then a list containing:

• data New data with redundant variables merged or removed
• merged A matrix containing the variables that were decided to be redundant with one another
• method Method used to perform redundancy reduction

adhoc If adhoc = TRUE, then the adhoc check containing the same objects as in the redundancy list object in the output

Author(s)

Alexander Christensen <alexpaulchristensen@gmail.com>

References

# Simulation using UVA

# Implementation of UVA (formally node.redundant)

# wTO measure

# Selection of CFA Estimator

Examples

# Select Five Factor Model personality items only
idx <- na.omit(match(gsub("-", "", unlist(psychTools::spi.keys[1:5])), colnames(psychTools::spi)))
items <- psychTools::spi[,idx]
# Change names in redundancy output to each item's description
key.ind <- match(colnames(items), as.character(psychTools::spi.dictionary$item_id))
key <- as.character(psychTools::spi.dictionary$item[key.ind])

# Automated selection of local dependence (default)
uva.results <- UVA(data = items, key = key)

# Produce Methods section
methods.section(uva.results)

# Manual selection of local dependence
if(interactive()){
  uva.results <- UVA(data = items, key = key, auto = FALSE)
}

---

### vn.entropy

**Entropy Fit Index using Von Neumann's entropy (Quantum Information Theory) for correlation matrices**

**Description**

Computes the fit of a dimensionality structure using Von Neumann's entropy when the input is a correlation matrix. Lower values suggest better fit of a structure to the data.

**Usage**

```r
vn.entropy(data, structure)
```

**Arguments**

- `data` A dataframe or a correlation matrix
- `structure` 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 Neumann's entropy
- **Total.Correlation** The total correlation of the dataset
- **Average.Entropy** The average entropy of the dataset

**Author(s)**

Hudson Golino <hfg9s@virginia.edu>, Alexander P. Christensen <alexpaulchristensen@gmail.com>, and Robert Moulder <rgm4jd@virginia.edu>
References


See Also

EGA to estimate the number of dimensions of an instrument using EGA and CFA to verify the fit of the structure suggested by EGA using confirmatory factor analysis.

Examples

```r
# Select Five Factor Model personality items only
idx <- na.omit(match(gsub("-", "", unlist(psychTools::spi.keys[1:5])), colnames(psychTools::spi)))
items <- psychTools::spi[,idx]

# Estimate EGA
ega.spi <- EGA(
data = items, model = "glasso",
plot.EGA = FALSE # No plot for CRAN checks
)

# Compute entropy indices
vn.entropy(
data = ega.spi$correlation,
structure = ega.spi$wc
)
```

---

**wmt2**  
*WMT-2 Data*

Description

A response matrix (n = 1185) of the Wiener Matrizen-Test 2 (WMT-2).

Usage

```r
data(wmt2)
```

Format

A 1185x24 response matrix

A 1185x24 response matrix
Examples

```r
data("wmt2")
data("wmt2")
data("wmt2")
```
Index

* datasets
  - boot.wmt, 7
  - depression, 17
  - dnn.weights, 19
  - ega.wmt, 38
  - intelligenceBattery, 48
  - optimism, 67
  - prime.num, 70
  - sim.dynEGA, 76
  - toy.example, 84
  - wmt2, 89
  - boot.ergoInfo, 5
  - boot.wmt, 7
  - bootEGA, 3, 4, 7–14, 18, 19, 22, 25, 27, 30, 33, 35–38, 40, 43, 44, 46, 49–51, 59–62, 64, 66, 71, 72, 75, 79, 80, 86, 88, 89
  - EGA.estimate, 32
  - EGA.fit, 4, 5, 9, 35
  - ega.wmt, 38
  - EGANet, 15, 16, 60, 61
  - EGANet (EGANet-package), 3
  - EGANet-package, 3
  - Embed, 21, 24, 38, 42, 59
  - entropyFit, 3, 4, 39
  - ergoInfo, 5, 41, 58
  - fa, 44
  - fitMeasures, 13
  - ggnet2, 10, 14, 16, 29, 45, 69, 74, 86
  - glasso, 25, 26
  - glla, 20, 23, 42
  - hclust, 48
  - hierEGA, 9, 43
  - igraph, 9, 17, 22, 24, 29, 30, 33, 36, 45, 56, 59, 74
  - infoCluster, 47
  - intelligenceBattery, 48
  - invariance, 49
  - itemStability, 4, 18, 50, 51, 61
  - jsd, 53
  - lavaan, 75, 86
  - lavOptions, 13, 75, 86
  - LCT, 4, 54
  - louvain, 56
  - mctest.ergoInfo, 57
  - methods.section, 60
  - mvnorm, 9
net.loads, 3, 4, 51, 61, 62, 64, 65
net.scores, 3, 44, 61, 64
network.descriptives, 66
optimism, 67
plot.bootEGA (plots), 68
plot.CFA (plots), 68
plot.dynEGA (plots), 68
plot.EGA (plots), 68
plots, 68
prime.num, 70
print.dynEGA (prints), 70
print.EGA (prints), 70
print.hierEGA (prints), 70
print.invariance (prints), 70
print.NetLoads (prints), 70
prints, 70
qgraph, 9, 14, 22, 25, 28, 32, 36, 45, 59, 66, 72, 73, 85
RCColorBrewer, 14
residualEGA, 71
riEGA, 9, 72
semPaths, 13, 69
sim.dynEGA, 76
simDFM, 4, 58, 76
summary.dynEGA (summaries), 78
summary.EGA (summaries), 78
summary.hierEGA (summaries), 78
summary.invariance (summaries), 78
summary.Net Loads (summaries), 78
summary.NetLoads (summaries), 78
summaries, 78
tefi, 3–5, 9, 10, 29, 33, 35, 36, 44, 46, 56, 74, 79
TMFG, 3, 9, 21, 24, 27, 29, 30, 32–34, 36, 59, 80
totalCor, 82
totalCorMat, 83
toy.example, 84
UVA, 3, 4, 61, 84
vn.entropy, 4, 88
wi2net, 26
wmt2, 7, 38, 89