Package 'latentFactoR'

November 22, 2022

Title Data Simulation Based on Latent Factors

Version 0.0.4

Date 2022-11-22

Maintainer Alexander Christensen <alexpaulchristensen@gmail.com>

Description Generates data based on latent factor models. Data can be continuous, polytomous, dichotomous, or mixed. Skews, cross-loadings, wording effects, population errors, and local dependencies can be added. All parameters can be manipulated. Data categorization is based on Garrido, Abad, and Ponsoda (2011) <doi:10.1177/0013164410389489>.

Depends R (>= 3.6.0)

License GPL (>= 3.0)

Imports BBmisc, EGAnet, fspe, googledrive, ineq, Matrix, methods, mlr, mvtnorm, psych, qgraph, rstudioapi, xgboost

Encoding UTF-8

RoxygenNote 7.2.2

NeedsCompilation no

Author Alexander Christensen [aut, cre] (<https://orcid.org/0000-0002-9798-7037>), Maria Dolores Nieto Canaveras [aut], Hudson Golino [aut] (<https://orcid.org/0000-0002-1601-1447>), Luis Eduardo Garrido [aut] (<https://orcid.org/0000-0001-8932-6063>), Marcos Jimenez [aut], Francisco Abad [ctb], Eduardo Garcia-Garzon [ctb], Vithor Franco [aut]

Repository CRAN

Date/Publication 2022-11-22 16:00:02 UTC

R topics documented:

`latentFactoR-package` *latentFactoR–package*

Description

Generates data based on latent factor models. Data can be continuous, polytomous, dichotomous, or mixed. Skew, cross-loadings, and population error can be added. All parameters can be manipulated. Data categorization is based on Garrido, Abad, and Ponsoda (2011).

Author(s)

Alexander P. Christensen <alexpaulchristensen@gmail.com>, Maria Dolores Nieto Canaveras <mnietoca@nebrija.es>, Hudson Golino <hfg9s@virginia.edu>, Luis Eduardo Garrido <luisgarrido@pucmm.edu>

References

Christensen, A. P., Garrido, L. E., & Golino, H. (2022).
Unique variable analysis: A network psychometrics method to detect local dependence.
PsyArXiv

Garrido, L. E., Abad, F. J., & Ponsoda, V. (2011).
Performance of Velicer's minimum average partial factor retention method with categorical variables.
Educational and Psychological Measurement, *71*(3), 551-570.

Golino, H., Shi, D., Christensen, A. P., Garrido, L. E., Nieto, M. D., Sadana, R., ... & Martinez-Molina, A. (2020). Investigating the performance of exploratory graph analysis and traditional techniques to identify the number of latent factors: A simulation and tutorial. *Psychological Methods*, *25*(3), 292-320.

add_cross_loadings *Adds (Substantial) Cross-loadings to* simulate_factors *Data*

Description

Intended to add substantial cross-loadings to simulated data from simulate_factors. See examples to get started

Usage

```
add_cross_loadings(
  lf_object,
  proportion_cross_loadings,
  proportion_cross_loadings_range = NULL,
  magnitude_cross_loadings,
  magnitude_cross_loadings_range = NULL,
  leave_cross_loadings = FALSE
)
```

Arguments

lf_object	Data object from simulate_factors
proportion_cross_loadings	Numeric (length = 1 or factors). Proportion of variables that should be cross-loaded randomly onto one other factor. Accepts number of variables to cross-load onto one other factor as well
proportion_cross_loadings_range	Numeric (length = 2). Range of proportion of variables that should be cross-loaded randomly onto one other factor. Accepts number of variables to cross-load onto one other factor as well
magnitude_cross_loadings	Numeric (length = 1, factors, or total number of variables to cross-load across all factors). The magnitude or size of the cross-loadings. Must range between -1 and 1.
magnitude_cross_loadings_range	Numeric (length = 2). The range of the magnitude or size of the cross-loadings. Defaults to NULL
leave_cross_loadings	Boolean. Should cross-loadings be kept? Defaults to FALSE. Convergence problems can arise if cross-loadings are kept, so setting them to zero is the default. Only set to TRUE with careful consideration of the structure. Make sure to perform additional checks that the data are adequate

Value

Returns a list containing the same parameters as the original lf_object but with updated data, population_correlation, and parameters (specifically, loadings matrix). Also returns original lf_object in original_results

Author(s)

Alexander P. Christensen <alexpaulchristensen@gmail.com>, Hudson Golino <hfg9s@virginia.edu>, Luis Eduardo Garrido <luisgarrido@pucmm.edu>

References

Christensen, A. P., Garrido, L. E., & Golino, H. (2022). Unique variable analysis: A network psychometrics method to detect local dependence. *PsyArXiv*

Examples

```
# Generate factor data
two_factor <- simulate_factors(
  factors = 2, # factors = 2
  variables = 6, # variables per factor = 6
  loadings = 0.55, # loadings between = 0.45 to 0.65
  cross_loadings = 0.05, # cross-loadings N(0, 0.05)
  correlations = 0.30, # correlation between factors = 0.30
  sample_size = 1000 # number of cases = 1000
)

# Add substantial cross-loadings
two_factor_CL <- add_cross_loadings(
  lf_object = two_factor,
  proportion_cross_loadings = 0.25,
  magnitude_cross_loadings = 0.35
)

# Randomly vary proportions
two_factor_CL <- add_cross_loadings(
  lf_object = two_factor,
  proportion_cross_loadings_range = c(0, 0.25),
  magnitude_cross_loadings = 0.35
)

# Randomly vary magnitudes
two_factor_CL <- add_cross_loadings(
  lf_object = two_factor,
  proportion_cross_loadings = 0.25,
  magnitude_cross_loadings_range = c(0.35, 0.45)
)

# Set number of cross-loadings per factor (rather than proportion)
two_factor_CL <- add_cross_loadings(
  lf_object = two_factor,
  proportion_cross_loadings = 2,
  magnitude_cross_loadings = 0.35
)
```

add_local_dependence *Adds Local Dependence to* simulate_factors *Data*

Description

Adds local dependence to simulated data from simulate_factors. See examples to get started

Usage

```
add_local_dependence(
  lf_object,
  method = c("correlate_residuals", "minor_factors", "threshold_shifts"),
  proportion_LD,
  proportion_LD_range = NULL,
  add_residuals = NULL,
  add_residuals_range = NULL,
  allow_multiple = FALSE
)
```

Arguments

lf_object	Data object from simulate_factors
method	Character (length = 1). Method to generate local dependence between variables. Only "correlate_residuals" at the moment. Future developments will include minor factor and threshold-shift methods. Description of methods: • "correlate_residuals" Adds residuals directly to the population correlation matrix prior to data generation (uses population correlation matrix from simulate_factors) • "minor_factors" Coming soon... • "threshold_shifts" Coming soon...
proportion_LD	Numeric (length = 1 or factors). Proportion of variables that should be locally dependent across all or each factor. Accepts number of locally dependent values as well
proportion_LD_range	Numeric (length = 2). Range of proportion of variables that are randomly selected from a random uniform distribution. Accepts number of locally dependent values as well. Defaults to NULL
add_residuals	Numeric (length = 1, factors, or total number of locally dependent variables). Amount of residual to add to the population correlation matrix between two variables. Only used when method = "correlated_residuals". Magnitudes are drawn from a random uniform distribution using +/- 0.05 of value input. Can also be specified directly (same length as total number of locally dependent variables). General effect sizes range from small (0.20), moderate (0.30), to large (0.40)

`add_residuals_range`
: Numeric (length = 2). Range of the residuals to add to the correlation matrix are randomly selected from a random uniform distribution. Defaults to `NULL`

`allow_multiple`
: Boolean. Whether a variable should be allowed to be locally dependent with more than one other variable. Defaults to `FALSE`. Set to `TRUE` for more complex locally dependence patterns

Value

Returns a list containing:

`data`
: Simulated data from the specified factor model

`population_correlation`
: Population correlation matrix with local dependence added

`original_correlation`
: Original population correlation matrix *before* local dependence was added

`correlated_residuals`
: A data frame with the first two columns specifying the variables that are locally dependent and the third column specifying the magnitude of the added residual for each locally dependent pair

`original_results`
: Original `lf_object` input into function

Author(s)

Alexander P. Christensen <alexpaulchristensen@gmail.com>, Hudson Golino <hfg9s@virginia.edu>, Luis Eduardo Garrido <luisgarrido@pucmm.edu>

References

Christensen, A. P., Garrido, L. E., & Golino, H. (2022). Unique variable analysis: A network psychometrics method to detect local dependence. *PsyArXiv*

Examples

```
# Generate factor data
two_factor <- simulate_factors(
  factors = 2, # factors = 2
  variables = 6, # variables per factor = 6
  loadings = 0.55, # loadings between = 0.45 to 0.65
  cross_loadings = 0.05, # cross-loadings N(0, 0.05)
  correlations = 0.30, # correlation between factors = 0.30
  sample_size = 1000 # number of cases = 1000
)

# Add local dependence
two_factor_LD <- add_local_dependence(
  lf_object = two_factor,
  proportion_LD = 0.25,
  add_residuals = 0.20,
```

```
  allow_multiple = FALSE
)

# Randomly vary proportions
two_factor_LD <- add_local_dependence(
  lf_object = two_factor,
  proportion_LD_range = c(0.10, 0.50),
  add_residuals = 0.20,
  allow_multiple = FALSE
)

# Randomly vary residuals
two_factor_LD <- add_local_dependence(
  lf_object = two_factor,
  proportion_LD = 0.25,
  add_residuals_range = c(0.20, 0.40),
  allow_multiple = FALSE
)

# Randomly vary proportions, residuals, and allow multiple
two_factor_LD <- add_local_dependence(
  lf_object = two_factor,
  proportion_LD_range = c(0.10, 0.50),
  add_residuals_range = c(0.20, 0.40),
  allow_multiple = TRUE
)
```

`add_population_error` *Adds Population Error to* `simulate_factors` *Data*

Description

Adds population error to simulated data from `simulate_factors`. See examples to get started

Usage

```
add_population_error(
  lf_object,
  cfa_method = c("minres", "ml"),
  fit = c("cfi", "rmsea", "rmsr", "raw"),
  misfit = c("close", "acceptable"),
  error_method = c("cudeck", "yuan"),
  tolerance = 0.01,
  convergence_iterations = 10,
  leave_cross_loadings = FALSE
)
```

Arguments

`lf_object`	Data object from `simulate_factors`
`cfa_method`	Character (length = 1). Method to generate population error. Defaults to `"minres"`. Available options: • `"minres"` Minimum residual • `"ml"` Maximum likelihood
`fit`	Character (length = 1). Fit index to control population error. Defaults to `"rmsr"`. Available options: • `"cfi"` Comparative fit index • `"rmsea"` Root mean square error of approximation • `"rmsr"` Root mean square residuals • `"raw"` Direct application of error
`misfit`	Character or numeric (length = 1). Magnitude of error to add. Defaults to `"close"`. Available options: • `"close"` Slight deviations from original population correlation matrix • `"acceptable"` Moderate deviations from original population correlation matrix While numbers can be used, they are **not** recommended. They can be used to specify misfit but the level of misfit will vary depending on the factor structure
`error_method`	Character (length = 1). Method to control population error. Defaults to `"cudeck"`. Description of methods: • `"cudeck"` Description coming soon... see Cudeck & Browne, 1992 for more details • `"yuan"` Description coming soon...
`tolerance`	Numeric (length = 1). Tolerance of SRMR difference between population error correlation matrix and the original population correlation matrix. Ensures that appropriate population error was added. Similarly, verifies that the MAE of the loadings are not greater than the specified amount, ensuring proper convergence. Defaults to `0.01`
`convergence_iterations`	Numeric (length = 1). Number of iterations to reach parameter convergence within the specified 'tolerance'. Defaults to `10`
`leave_cross_loadings`	Boolean. Should cross-loadings be kept? Defaults to `FALSE`. Convergence problems can arise if cross-loadings are kept, so setting them to zero is the default. Only set to `TRUE` with careful consideration of the structure. Make sure to perform additional checks that the data are adequate

Value

Returns a list containing:

`data`	Simulated data from the specified factor model

`population_correlation`
: Population correlation matrix with local dependence added

`population_error`
: A list containing the parameters used to generate population error:

  - `error_correlation` Correlation matrix with population error added (same as `population_correlation`)
  - `fit` Fit measure used to control population error
  - `delta` Minimum of the objective function corresponding to the misfit value
  - `misfit` Specified misfit value
  - `loadings` Estiamted CFA loadings after error has been added

`original_results`
: Original `lf_object` input into function

Author(s)

`bifactor` authors
Marcos Jimenez, Francisco J. Abad, Eduardo Garcia-Garzon, Vithor R. Franco, Luis Eduardo Garrido <luisgarrido@pucmm.edu>

`latentFactoR` authors
Alexander P. Christensen <alexpaulchristensen@gmail.com>, Hudson Golino <hfg9s@virginia.edu>, Luis Eduardo Garrido <luisgarrido@pucmm.edu>, Marcos Jimenez, Francisco J. Abad, Eduardo Garcia-Garzon, Vithor R. Franco

References

Christensen, A. P., Garrido, L. E., & Golino, H. (2022). Unique variable analysis: A network psychometrics method to detect local dependence. *PsyArXiv*

Cudeck, R., & Browne, M.W. (1992). Constructing a covariance matrix that yields a specified minimizer and a specified minimum discrepancy function value. *Psychometrika*, *57*, 357–369.

Jimenez, M., Abad, F. J., Garcia-Garzon, E., Golino, H., Christensen, A. P., & Garrido, L. E. (2022). Dimensionality assessment in generalized bi-factor structures: A network psychometrics approach. *PsyArXiv*

Examples

```
# Generate factor data
two_factor <- simulate_factors(
  factors = 2, # factors = 2
  variables = 6, # variables per factor = 6
  loadings = 0.55, # loadings between = 0.45 to 0.65
  cross_loadings = 0.05, # cross-loadings N(0, 0.05)
  correlations = 0.30, # correlation between factors = 0.30
  sample_size = 1000 # number of cases = 1000
)

# Add small population error using Cudeck method
two_factor_Cudeck <- add_population_error(
  lf_object = two_factor,
```

```
  cfa_method = "minres",
  fit = "rmsr", misfit = "close",
  error_method = "cudeck"
)

# Add small population error using Yuan method
two_factor_Yuan <- add_population_error(
  lf_object = two_factor,
  cfa_method = "minres",
  fit = "rmsr", misfit = "close",
  error_method = "yuan"
)
```

`categorize` *Categorize Continuous Data*

Description

Categorizes continuous data based on Garrido, Abad and Ponsoda (2011; see references). Categorical data with 2 to 6 categories can include skew between -2 to 2 in increments of 0.05

Usage

```
categorize(data, categories, skew_value = 0)
```

Arguments

`data`	Numeric (length = n). A vector of continuous data with *n* values. For matrices, use `apply`
`categories`	Numeric (length = 1). Number of categories to create. Between 2 and 6 categories can be used with skew
`skew_value`	Numeric (length = 1). Value of skew. Ranges between -2 to 2 in increments of 0.05. Skews not in this sequence will be converted to the nearest value in this sequence. Defaults to `0` or no skew

Value

Returns a numeric vector of the categorize data

Author(s)

Maria Dolores Nieto Canaveras <mnietoca@nebrija.es>, Luis Eduardo Garrido <luisgarrido@pucmm.edu>, Hudson Golino <hfg9s@virginia.edu>, Alexander P. Christensen <alexpaulchristensen@gmail.com>

References

Garrido, L. E., Abad, F. J., & Ponsoda, V. (2011).
Performance of Velicer's minimum average partial factor retention method with categorical variables.
Educational and Psychological Measurement, *71*(3), 551-570.

Golino, H., Shi, D., Christensen, A. P., Garrido, L. E., Nieto, M. D., Sadana, R., ... & Martinez-Molina, A. (2020). Investigating the performance of exploratory graph analysis and traditional techniques to identify the number of latent factors: A simulation and tutorial. *Psychological Methods*, *25*(3), 292-320.

Examples

```
# Dichotomous data (no skew)
dichotomous <- categorize(
  data = rnorm(1000),
  categories = 2
)

# Dichotomous data (with positive skew)
dichotomous_skew <- categorize(
  data = rnorm(1000),
  categories = 2,
  skew_value = 1.25
)

# 5-point Likert scale (no skew)
five_likert <- categorize(
  data = rnorm(1000),
  categories = 5
)

# 5-point Likert scale (negative skew)
five_likert <- categorize(
  data = rnorm(1000),
  categories = 5,
  skew_value = -0.45
)
```

---

data_to_zipfs *Transforms* simulate_factors *Data to Zipf's Distribution*

Description

Zipf's distribution is commonly found for text data. Closely related to the Pareto and power-law distributions, the Zipf's distribution produces highly skewed data. This transformation is intended to mirror the data generating process of Zipf's law seen in semantic network and topic modeling data.

Usage

```
data_to_zipfs(lf_object, beta = 2.7, alpha = 1, dichotomous = FALSE)
```

Arguments

`lf_object`	Data object from `simulate_factors`
`beta`	Numeric (length = 1). Sets the shift in rank. Defaults to `2.7`
`alpha`	Numeric (length = 1). Sets the power of the rank. Defaults to `1`
`dichotomous`	Boolean (length = 1). Whether data should be dichotomized rather than frequencies (e.g., semantic network analysis). Defaults to `FALSE`

Details

The formula used to transform data is (Piantadosi, 2014):

f(r) proportional to 1 / (r + beta)^alpha

where *f(r)* is the *r*th most frequency, *r* is the rank-order of the data, *beta* is a shift in the rank (following Mandelbrot, 1953, 1962), and *alpha* is the power of the rank with greater values suggesting greater differences between the largest frequency to the next, and so forth.

The function will transform continuous data output from `simulate_factors`. See examples to get started

Value

Returns a list containing:

`data`	Simulated data that has been transform to follow Zipf's distribution
`RMSE`	A vector of root mean square errors for transformed data and data assumed to follow theoretical Zipf's distribution and Spearman's correlation matrix of the transformed data compared to the original population correlation matrix
`spearman_correlation`	Spearman's correlation matrix of the transformed data
`original_correlation`	Original population correlation matrix *before* the data were transformed
`original_results`	Original `lf_object` input into function

Author(s)

Alexander P. Christensen <alexpaulchristensen@gmail.com>, Hudson Golino <hfg9s@virginia.edu>, Luis Eduardo Garrido <luisgarrido@pucmm.edu>

References

Mandelbrot, B. (1953). An informational theory of the statistical structure of language. *Communication Theory*, *84*, 486–502.

Mandelbrot, B. (1962). On the theory of word frequencies and on related Markovian models of discourse. *Structure of Language and its Mathematical Aspects*, 190–219.

Piantadosi, S. T. (2014). Zipf's word frequency law in natural language: A critical review and future directions. *Psychonomic Bulletin & Review*, *21*(5), 1112-1130.

Zipf, G. (1936). *The psychobiology of language*. London, UK: Routledge.

Zipf, G. (1949). *Human behavior and the principle of least effort*. New York, NY: Addison-Wesley.

Examples

```
# Generate factor data
two_factor <- simulate_factors(
  factors = 2, # factors = 2
  variables = 6, # variables per factor = 6
  loadings = 0.55, # loadings between = 0.45 to 0.65
  cross_loadings = 0.05, # cross-loadings N(0, 0.05)
  correlations = 0.30, # correlation between factors = 0.30
  sample_size = 1000 # number of cases = 1000
)

# Transform data to Mandelbrot's Zipf's
two_factor_zipfs <- data_to_zipfs(
  lf_object = two_factor,
  beta = 2.7,
  alpha = 1
)

# Transform data to Mandelbrot's Zipf's (dichotomous)
two_factor_zipfs_binary <- data_to_zipfs(
  lf_object = two_factor,
  beta = 2.7,
  alpha = 1,
  dichotomous = TRUE
)
```

EKC *Estimate Number of Dimensions using Empirical Kaiser Criterion*

Description

Estimates the number of dimensions in data using Empirical Kaiser Criterion (Braeken & Van Assen, 2017). See examples to get started

Usage

```
EKC(data, sample_size)
```

Arguments

data	Matrix or data frame. Either a dataset with all numeric values (rows = cases, columns = variables) or a symmetric correlation matrix
sample_size	Numeric (length = 1). If input into data is a correlation matrix, then specifying the sample size is required

Value

Returns a list containing:

dimensions	Number of dimensions identified
eigenvalues	Eigenvalues
reference	Reference values compared against eigenvalues

Author(s)

Alexander P. Christensen <alexpaulchristensen@gmail.com>, Hudson Golino <hfg9s@virginia.edu>, Luis Eduardo Garrido <luisgarrido@pucmm.edu>

References

Braeken, J., & Van Assen, M. A. (2017). An empirical Kaiser criterion. *Psychological Methods*, *22*(3), 450–466.

Examples

```
# Generate factor data
two_factor <- simulate_factors(
  factors = 2, # factors = 2
  variables = 6, # variables per factor = 6
  loadings = 0.55, # loadings between = 0.45 to 0.65
  cross_loadings = 0.05, # cross-loadings N(0, 0.05)
  correlations = 0.30, # correlation between factors = 0.30
  sample_size = 1000 # number of cases = 1000
)

# Perform Empirical Kaiser Criterion
EKC(two_factor$data)
```

`estimate_dimensions` *Estimates Dimensions using Several State-of-the-art Methods*

Description

Estimates dimensions using Exploratory Graph Analysis (`EGA`), Empirical Kaiser Criterion (`EKC`), Factor Forest (`factor_forest`), Exploratory Factor Analysis with out-of-sample prediction (`fspe`), Next Eigenvalue Sufficiency Test (`NEST`), and parallel analysis (`fa.parallel`)

Usage

```
estimate_dimensions(
  data,
  sample_size,
  EGA_args = list(corr = "cor_auto", uni.method = "louvain", model = "glasso",
    consensus.method = "most_common", plot.EGA = FALSE),
  FF_args = list(maximum_factors = 8),
  FSPE_args = list(maxK = 8, rep = 1, method = "PE", pbar = FALSE),
 NEST_args = list(iterations = 1000, maximum_iterations = 500, alpha = 0.05, convergence
    = 1e-05),
 PA_args = list(fm = "minres", fa = "both", cor = "cor", n.iter = 20, sim = FALSE, plot
    = FALSE)
)
```

Arguments

`data`	Matrix or data frame. Either a dataset with all numeric values (rows = cases, columns = variables) or a symmetric correlation matrix
`sample_size`	Numeric (length = 1). If input into `data` is a correlation matrix, then specifying the sample size is required
`EGA_args`	List. List of arguments to be passed along to `EGA`. Defaults are listed
`FF_args`	List. List of arguments to be passed along to `factor_forest`. Defaults are listed
`FSPE_args`	List. List of arguments to be passed along to `fspe`. Defaults are listed
`NEST_args`	List. List of arguments to be passed along to `NEST`. Defaults are listed
`PA_args`	List. List of arguments to be passed along to `fa.parallel`. Defaults are listed

Value

Returns a list containing:

`dimensions`	Dimensions estimated from each method

A list of each methods output (see their respective functions for their outputs)

Author(s)

Maria Dolores Nieto Canaveras <mnietoca@nebrija.es>, Alexander P. Christensen <alexpaulchristensen@gmail.com>, Hudson Golino <hfg9s@virginia.edu>, Luis Eduardo Garrido <luisgarrido@pucmm.edu>

Examples

```
# Generate factor data
two_factor <- simulate_factors(
  factors = 2, # factors = 2
  variables = 6, # variables per factor = 6
  loadings = 0.55, # loadings between = 0.45 to 0.65
  cross_loadings = 0.05, # cross-loadings N(0, 0.05)
  correlations = 0.30, # correlation between factors = 0.30
  sample_size = 1000 # number of cases = 1000
)

## Not run:
# Estimate dimensions
estimate_dimensions(two_factor$data)
## End(Not run)
```

---

factor_forest *Estimate Number of Dimensions using Factor Forest*

Description

Estimates the number of dimensions in data using the pre-trained Random Forest model from Goretzko and Buhner (2020, 2022). See examples to get started

Usage

```
factor_forest(data, sample_size, maximum_factors = 8)
```

Arguments

`data`	Matrix or data frame. Either a dataset with all numeric values (rows = cases, columns = variables) or a symmetric correlation matrix
`sample_size`	Numeric (length = 1). If input into `data` is a correlation matrix, then specifying the sample size is required
`maximum_factors`	Numeric (length = 1). Maximum number of factors to search over. Defaults to `8`

Value

Returns a list containing:

`dimensions`	Number of dimensions identified
`probabilities`	Probability that the number of dimensions is most likely

Author(s)

Authors of Factor Forest
David Goretzko and Markus Buhner

Authors of latentFactoR
Alexander P. Christensen <alexpaulchristensen@gmail.com>, Hudson Golino <hfg9s@virginia.edu>, Luis Eduardo Garrido <luisgarrido@pucmm.edu>

References

Goretzko, D., & Buhner, M. (2022). Factor retention using machine learning with ordinal data. *Applied Psychological Measurement*, 01466216221089345.

Goretzko, D., & Buhner, M. (2020). One model to rule them all? Using machine learning algorithms to determine the number of factors in exploratory factor analysis. *Psychological Methods*, *25*(6), 776-786.

Examples

```
# Generate factor data
two_factor <- simulate_factors(
  factors = 2, # factors = 2
  variables = 6, # variables per factor = 6
  loadings = 0.55, # loadings between = 0.45 to 0.65
  cross_loadings = 0.05, # cross-loadings N(0, 0.05)
  correlations = 0.30, # correlation between factors = 0.30
  sample_size = 1000 # number of cases = 1000
)

## Not run:
# Perform Factor Forest
factor_forest(two_factor$data)
## End(Not run)
```

NEST *Estimate Number of Dimensions using Next Eigenvalue Sufficiency Test*

Description

Estimates the number of dimensions in data using NEST (Achim, 2017). See examples to get started

Usage

```
NEST(
  data,
  sample_size,
  iterations = 1000,
  maximum_iterations = 500,
```

```
  alpha = 0.05,
  convergence = 1e-05
)
```

Arguments

`data`	Matrix or data frame. Either a dataset with all numeric values (rows = cases, columns = variables) or a symmetric correlation matrix
`sample_size`	Numeric (length = 1). If input into `data` is a correlation matrix, then specifying the sample size is required
`iterations`	Numeric (length = 1). Number of iterations to estimate rank. Defaults to `1000`
`maximum_iterations`	Numeric (length = 1). Maximum umber of iterations to obtain convergence of eigenvalues. Defaults to `500`
`alpha`	Numeric (length = 1). Significance level for determine sufficient eigenvalues. Defaults to `0.05`
`convergence`	Numeric (length = 1). Value necessary to be less than or equal to when establishing convergence of eigenvalues

Value

Returns a list containing:

`dimensions`	Number of dimensions identified
`loadings`	Loading matrix
`converged`	Whether estimation converged. If `FALSE`, then results are reported from last convergence point. Interpret results with caution.

Author(s)

Alexander P. Christensen <alexpaulchristensen@gmail.com>, Hudson Golino <hfg9s@virginia.edu>, Luis Eduardo Garrido <luisgarrido@pucmm.edu>

References

Achim, A. (2017). Testing the number of required dimensions in exploratory factor analysis. *The Quantitative Methods for Psychology*, *13*(1), 64–74.

Brandenburg, N., & Papenberg, M. (2022). Reassessment of innovative methods to determine the number of factors: A simulation-Based comparison of Exploratory Graph Analysis and Next Eigenvalue Sufficiency Test. *Psychological Methods*.

Examples

```
# Generate factor data
two_factor <- simulate_factors(
  factors = 2, # factors = 2
  variables = 6, # variables per factor = 6
  loadings = 0.55, # loadings between = 0.45 to 0.65
```

```
  cross_loadings = 0.05, # cross-loadings N(0, 0.05)
  correlations = 0.30, # correlation between factors = 0.30
  sample_size = 1000 # number of cases = 1000
)

## Not run:
# Perform NEST
NEST(two_factor$data)
## End(Not run)
```

`obtain_zipfs_parameters` *Obtain Zipf's Distribution Parameters from Data*

Description

Zipf's distribution is commonly found for text data. Closely related to the Pareto and power-law distributions, the Zipf's distribution produces highly skewed data. This function obtains the best fitting parameters to Zipf's distribution

Usage

```
obtain_zipfs_parameters(data)
```

Arguments

`data`	Numeric vector, matrix, or data frame. Numeric data to determine Zipf's distribution parameters

Details

The best parameters are optimized by minimizing the aboslute difference between the original frequencies and the frequencies obtained by the *beta* and *alpha* parameters in the following formula (Piantadosi, 2014):

f(r) proportional to 1 / (r + beta)^alpha

where *f(r)* is the *r*th most frequency, *r* is the rank-order of the data, *beta* is a shift in the rank (following Mandelbrot, 1953, 1962), and *alpha* is the power of the rank with greater values suggesting greater differences between the largest frequency to the next, and so forth.

Value

Returns a vector containing the estimated `beta` and `alpha` parameters. Also contains `zipfs_sse` which corresponds to the sum of square error between frequencies based on the parameter values estimated and the original data frequencies

Author(s)

Alexander P. Christensen <alexpaulchristensen@gmail.com>, Hudson Golino <hfg9s@virginia.edu>, Luis Eduardo Garrido <luisgarrido@pucmm.edu>

References

Mandelbrot, B. (1953). An informational theory of the statistical structure of language. *Communication Theory*, *84*, 486–502.

Mandelbrot, B. (1962). On the theory of word frequencies and on related Markovian models of discourse. *Structure of Language and its Mathematical Aspects*, 190–219.

Piantadosi, S. T. (2014). Zipf's word frequency law in natural language: A critical review and future directions. *Psychonomic Bulletin & Review*, *21*(5), 1112-1130.

Examples

```
# Generate factor data
two_factor <- simulate_factors(
  factors = 2, # factors = 2
  variables = 6, # variables per factor = 6
  loadings = 0.55, # loadings between = 0.45 to 0.65
  cross_loadings = 0.05, # cross-loadings N(0, 0.05)
  correlations = 0.30, # correlation between factors = 0.30
  sample_size = 1000 # number of cases = 1000
)

# Transform data to Mandelbrot's Zipf's
two_factor_zipfs <- data_to_zipfs(
  lf_object = two_factor,
  beta = 2.7,
  alpha = 1
)

# Obtain Zipf's distribution parameters
obtain_zipfs_parameters(two_factor_zipfs$data)
```

`simulate_factors` *Simulates Latent Factor Data*

Description

Simulates data from a latent factor model based on many manipulable parameters. Parameters do not have default values and must each be set. See examples to get started

Usage

```
simulate_factors(
  factors,
  variables,
  variables_range = NULL,
  loadings,
  loadings_range = NULL,
  cross_loadings,
  cross_loadings_range = NULL,
  correlations,
  correlations_range = NULL,
  sample_size,
  variable_categories = Inf,
  categorical_limit = 6,
  skew = 0,
  skew_range = NULL
)
```

Arguments

factors	Numeric (length = 1). Number of factors
variables	Numeric (length = 1 or factors). Number of variables per factor. Can be a single value or as many values as there are factors. Minimum three variables per factor
variables_range	Numeric (length = 2). Range of variables to randomly select from a random uniform distribution. Minimum three variables per factor
loadings	Numeric or matrix (length = 1, factors, total number of variables (factors x variables), or factors x total number of variables. Loadings drawn from a random uniform distribution using +/- 0.10 of value input. Can be a single value or as many values as there are factors (corresponding to the factors). Can also be a loading matrix. Columns must match factors and rows must match total variables (factors x variables) General effect sizes range from small (0.40), moderate (0.55), to large (0.70)
loadings_range	Numeric (length = 2). Range of loadings to randomly select from a random uniform distribution. General effect sizes range from small (0.40), moderate (0.55), to large (0.70)
cross_loadings	Numeric or matrix(length = 1, factors, or factors x total number of variables. Cross-loadings drawn from a random normal distribution with a mean of 0 and standard deviation of value input. Can be a single value or as many values as there are factors (corresponding to the factors). Can also be a loading matrix. Columns must match factors and rows must match total variables (factors x variables)
cross_loadings_range	Numeric (length = 2). Range of cross-loadings to randomly select from a random uniform distribution

`correlations` Numeric (length = 1 or `factors` x `factors`). Can be a single value that will be used for all correlations between factors. Can also be a square matrix (`factors` x `factors`). General effect sizes range from orthogonal (0.00), small (0.30), moderate (0.50), to large (0.70)

`correlations_range` Numeric (length = 2). Range of correlations to randomly select from a random uniform distribution. General effect sizes range from orthogonal (0.00), small (0.30), moderate (0.50), to large (0.70)

`sample_size` Numeric (length = 1). Number of cases to generate from a random multivariate normal distribution using `rmvnorm`

`variable_categories` Numeric (length = 1 or total variables (`factors` x `variables`)). Number of categories for each variable. `Inf` is used for continuous variables; otherwise, values reflect number of categories

`categorical_limit` Numeric (length = 1). Values greater than input value are considered continuous. Defaults to `6` meaning that 7 or more categories are considered continuous (i.e., variables are *not* categorized from continuous to categorical)

`skew` Numeric (length = 1 or categorical variables). Skew to be included in categorical variables. It is randomly sampled from provided values. Can be a single value or as many values as there are (total) variables. Current skew implementation is between -2 and 2 in increments of 0.05. Skews that are not in this sequence will be converted to their nearest value in the sequence. Not recommended to use with `variables_range`. Future versions will incorporate finer skews

`skew_range` Numeric (length = 2). Randomly selects skews within in the range. Somewhat redundant with `skew` but more flexible. Compatible with `variables_range`

Value

Returns a list containing:

`data` Simulated data from the specified factor model

`population_correlation` Population correlation matrix

`parameters` A list containing the parameters used to generate the data:

- `factors` Number of factors
- `variables` Variables on each factor
- `loadings` Loading matrix
- `factor_correlations` Correlations between factors
- `categories` Categories for each variable
- `skew` Skew for each variable

Author(s)

Maria Dolores Nieto Canaveras <mnietoca@nebrija.es>, Alexander P. Christensen <alexpaulchristensen@gmail.com>, Hudson Golino <hfg9s@virginia.edu>, Luis Eduardo Garrido <luisgarrido@pucmm.edu>

References

Garrido, L. E., Abad, F. J., & Ponsoda, V. (2011).
Performance of Velicer's minimum average partial factor retention method with categorical variables.
Educational and Psychological Measurement, *71*(3), 551-570.

Golino, H., Shi, D., Christensen, A. P., Garrido, L. E., Nieto, M. D., Sadana, R., ... & Martinez-Molina, A. (2020). Investigating the performance of exploratory graph analysis and traditional techniques to identify the number of latent factors: A simulation and tutorial. *Psychological Methods*, *25*(3), 292-320.

Examples

```
# Generate factor data
two_factor <- simulate_factors(
  factors = 2, # factors = 2
  variables = 6, # variables per factor = 6
  loadings = 0.55, # loadings between = 0.45 to 0.65
  cross_loadings = 0.05, # cross-loadings N(0, 0.05)
  correlations = 0.30, # correlation between factors = 0.30
  sample_size = 1000 # number of cases = 1000
)

# Randomly vary loadings
two_factor_loadings <- simulate_factors(
  factors = 2, # factors = 2
  variables = 6, # variables per factor = 6
  loadings_range = c(0.30, 0.80), # loadings between = 0.30 to 0.80
  cross_loadings = 0.05, # cross-loadings N(0, 0.05)
  correlations = 0.30, # correlation between factors = 0.30
  sample_size = 1000 # number of cases = 1000
)

# Generate dichotomous data
two_factor_dichotomous <- simulate_factors(
  factors = 2, # factors = 2
  variables = 6, # variables per factor = 6
  loadings = 0.55, # loadings between = 0.45 to 0.65
  cross_loadings = 0.05, # cross-loadings N(0, 0.05)
  correlations = 0.30, # correlation between factors = 0.30
  sample_size = 1000, # number of cases = 1000
  variable_categories = 2 # dichotomous data
)

# Generate dichotomous data with skew
two_factor_dichotomous_skew <- simulate_factors(
  factors = 2, # factors = 2
  variables = 6, # variables per factor = 6
  loadings = 0.55, # loadings between = 0.45 to 0.65
  cross_loadings = 0.05, # cross-loadings N(0, 0.05)
  correlations = 0.30, # correlation between factors = 0.30
  sample_size = 1000, # number of cases = 1000
```

```
  variable_categories = 2, # dichotomous data
  skew = 1 # all variables with have a skew of 1
)

# Generate dichotomous data with variable skew
two_factor_dichotomous_skew <- simulate_factors(
  factors = 2, # factors = 2
  variables = 6, # variables per factor = 6
  loadings = 0.55, # loadings between = 0.45 to 0.65
  cross_loadings = 0.05, # cross-loadings N(0, 0.05)
  correlations = 0.30, # correlation between factors = 0.30
  sample_size = 1000, # number of cases = 1000
  variable_categories = 2, # dichotomous data
  skew_range = c(-2, 2) # skew = -2 to 2 (increments of 0.05)
)
```

`skew_tables` *Skew Tables*

Description

Tables for skew based on the number of categories (2, 3, 4, 5, or 6) in the data

Usage

```
data(skew_tables)
```

Format

A list (length = 5)

Examples

```
data("skew_tables")
```

Index