Package: EFAfactors (via r-universe)

October 28, 2026

Type Package

Title Determining the Number of Factors in Exploratory Factor Analysis

Version 1.2.4

Date 2025-10-13

Author Haijiang Qin [aut, cre, cph] (ORCID: <https://orcid.org/0009-0000-6721-5653>), Lei Guo [aut, cph] (ORCID: <https://orcid.org/0000-0002-8273-3587>)

Maintainer Haijiang Qin <haijiang133@outlook.com>

Description Provides a collection of standard factor retention methods in Exploratory Factor Analysis (EFA), making it easier to determine the number of factors. Traditional methods such as the scree plot by Cattell (1966) <doi:10.1207/s15327906mbr0102_10>, Kaiser-Guttman Criterion (KGC) by Guttman (1954) <doi:10.1007/BF02289162> and Kaiser (1960) <doi:10.1177/001316446002000116>, and flexible Parallel Analysis (PA) by Horn (1965) <doi:10.1007/BF02289447> based on eigenvalues form PCA or EFA are readily available. This package also implements several newer methods, such as the Empirical Kaiser Criterion (EKC) by Braeken and van Assen (2017) <doi:10.1037/met0000074>, Comparison Data (CD) by Ruscio and Roche (2012) <doi:10.1037/a0025697>, and Hull method by Lorenzo-Seva et al. (2011) <doi:10.1080/00273171.2011.564527>, as well as some AI-based methods like Comparison Data Forest (CDF) by Goretzko and Ruscio (2024) <doi:10.3758/s13428-023-02122-4> and Factor Forest (FF) by Goretzko and Buhner (2020) <doi:10.1037/met0000262>. Additionally, it includes a deep neural network (DNN) trained on large-scale datasets that can efficiently and reliably determine the number of factors.

License GPL-3

Depends R (>= 4.3.0)

Imports BBmisc, checkmate, ddpcr, ineq, MASS, Matrix, mlr, proxy, psych, ranger, reticulate, Rcpp, RcppArmadillo, SimCorMultRes, xgboost

LinkingTo Rcpp, RcppArmadillo

RoxygenNote 7.3.2

Encoding UTF-8

NeedsCompilation yes

Collate 'CD.R' 'CDF.R' 'check_python_libraries.R' 'data.bfi.R' 'data.DAPCS.R' 'data.datasets.DNN.R' 'data.datasets.LSTM.R' 'data.scaler.DNN.R' 'data.scaler.LSTM.R' 'NN.R' 'EFAhclust.R' 'EFAindex.R' 'EFAkmeans.R' 'EFAvote.R' 'EKC.R' 'EFAscreet.R' 'EFAsim.data.R' 'extractor.feature.NN.R' 'extractor.feature.FF.R' 'factor.analysis.R' 'FF.R' 'GenData.R' 'get.runs.R' 'Hull.R' 'KGC.R' 'load.R' 'MAP.R' 'model.xgb.R' 'normalizor.R' 'PA.R' 'ParamHelpers.R' 'plot.R' 'print.R' 'RcppExports.R' 'af.softmax.R' 'utils.R' 'zzz.R' 'STOC.R'

URL https://haijiangqin.com/EFAfactors/

Repository https://cran.r-universe.dev

Date/Publication 2025-10-14 14:40:27 UTC

RemoteUrl https://github.com/cran/EFAfactors

RemoteRef HEAD

RemoteSha d0b8d80fd30aa52dc61e9393f69775229c191b85

Contents

af.softmax . . . 3
CD . . . 4
CDF . . . 6
check_python_libraries . . . 9
data.bfi . . . 9
data.DAPCS . . . 11
data.datasets.DNN . . . 12
data.datasets.LSTM . . . 13
data.scaler.DNN . . . 13
data.scaler.LSTM . . . 14
EFAhclust . . . 15
EFAindex . . . 17
EFAkmeans . . . 19
EFAscreet . . . 20
EFAsim.data . . . 22
EFAvote . . . 24
EKC . . . 25
extractor.feature.FF . . . 28
extractor.feature.NN . . . 30
factor.analysis . . . 32
FF . . . 34
GenData . . . 38

Hull 40
KGC 42
load.NN 44
load.scaler 45
load.xgb 46
MAP 47
model.xgb 48
NN 49
normalizor 53
PA 54
plot 57
predictLearner.classif.xgboost.earlystop 59
print 60
STOC 62

Index **65**

`af.softmax` *An Activation Function: Softmax*

Description

This function computes the softmax of a numeric vector. The softmax function transforms a vector of real values into a probability distribution, where each element is between 0 and 1 and the sum of all elements is 1. @seealso NN

Usage

```
af.softmax(x)
```

Arguments

x	A numeric vector for which the softmax transformation is to be computed.

Details

The softmax function is calculated as:

$$softmax(x_i) = \frac{exp(x_i)}{\sum_j exp(x_j)}$$

In the case of overflow (i.e., when $exp(x_i)$ is too large), this function handles `Inf` values by assigning `1` to the corresponding positions and `0` to the others before Softmax. @seealso NN

Value

A numeric vector representing the softmax-transformed values of `x`.

Examples

```
x <- c(1, 2, 3)
af.softmax(x)
```

CD *the Comparison Data (CD) Approach*

Description

This function runs the Comparison Data (CD) approach of Ruscio & Roche (2012).

Usage

```
CD(
  response,
  nfact.max = 10,
  N.pop = 10000,
  N.Samples = 500,
  Alpha = 0.3,
  cor.type = "pearson",
  use = "pairwise.complete.obs",
  vis = TRUE,
  plot = TRUE
)
```

Arguments

response	A required N × I matrix or data.frame consisting of the responses of N individuals to × I items.
nfact.max	The maximum number of factors discussed by CD approach. (default = 10)
N.pop	Size of finite populations of simulating.. (default = 10,000)
N.Samples	Number of samples drawn from each population. (default = 500) Each Sample is consisted of a N × I matrix, just like the empirical data.
Alpha	Alpha level when testing statistical significance (Wilcoxon Rank Sum and Signed Rank Tests) of improvement with additional factor. (default = .30)
cor.type	A character string indicating which correlation coefficient (or covariance) is to be computed. One of "pearson" (default), "kendall", or "spearman". @seealso cor.
use	an optional character string giving a method for computing covariances in the presence of missing values. This must be one of the strings "everything", "all.obs", "complete.obs", "na.or.complete", or "pairwise.complete.obs" (default). @seealso cor.
vis	A Boolean variable that will print the factor retention results when set to TRUE, and will not print when set to FALSE. (default = TRUE)
plot	A Boolean variable that will print the CD plot when set to TRUE, and will not print it when set to FALSE. @seealso plot.CD. (Default = TRUE)

Details

Ruscio and Roche (2012) proposed a method for determining the number of factors through comparison data (CD). This method identifies the appropriate number of factors by finding the solution that best reproduces the pattern of eigenvalues. CD employs an iterative procedure when generating comparison data with a known number of factors. Initially, CD compares whether the simulated comparison data with one latent factor (j=1) reproduces the empirical eigenvalue pattern significantly worse than the two-factor solution (j+1). If so, CD increases the value of j until further improvements are no longer significant or a preset maximum number of factors is reached. Specifically, CD involves five steps:

1. Generate random data with either j or j+1 latent factors and calculate the eigenvalues of the respective correlation matrices.

2. Compute the root mean square error (RMSE) of the difference between the empirical and simulated eigenvalues using the formula

$$RMSE = \sqrt{\frac{1}{p}\sum_{i=1}^{p}(\lambda_{emp,i} - \lambda_{sim,i})^2}$$

where:

- $\lambda_{emp,i}$: The i-th empirical eigenvalue.
- $\lambda_{sim,i}$: The i-th simulated eigenvalue.
- p: The number of items or eigenvalues.

This step produces two RMSEs, corresponding to the different numbers of latent factors.

3. Repeat steps 1 and 2 for 500 times (default in the Package).

4. Use a one-sided Wilcoxon test (alpha = 0.30) to assess whether the RMSEs is significantly reduced under the two-factor condition.

5. If the difference in RMSEs is not significant, CD suggests selecting j factors. Otherwise, j is increased by 1, and steps 1 to 4 are repeated.

The code is implemented based on the resources available at:

- https://ruscio.pages.tcnj.edu/quantitative-methods-program-code/
- https://osf.io/gqma2/?view_only=d03efba1fd0f4c849a87db82e6705668
- https://osf.io/mvrau/

Since the CD approach requires extensive data simulation and computation, C++ code is used to speed up the process.

Value

An object of class `CD` is a `list` containing the following components:

`nfact`	The number of factors to be retained.
`RMSE.Eigs`	A matrix containing the root mean square error (RMSE) of the eigenvalues produced by each simulation for every discussed number of factors.
`Sig`	A boolean variable indicating whether the significance level of the Wilcoxon Rank Sum and Signed Rank Tests has reached Alpha.

Author(s)

Haijiang Qin <Haijiang133@outlook.com>

References

Auerswald, M., & Moshagen, M. (2019). How to determine the number of factors to retain in exploratory factor analysis: A comparison of extraction methods under realistic conditions. Psychological methods, 24(4), 468-491. https://doi.org/https://doi.org/10.1037/met0000200.

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. Psychol Methods, 25(6), 776-786. https://doi.org/10.1037/met0000262.

Ruscio, J., & Roche, B. (2012). Determining the number of factors to retain in an exploratory factor analysis using comparison data of known factorial structure. Psychological Assessment, 24, 282–292. http://dx.doi.org/10.1037/a0025697.

See Also

GenData

CDF *the Comparison Data Forest (CDF) Approach*

Description

The Comparison Data Forest (CDF; Goretzko & Ruscio, 2019) approach is a combination of Random Forest with the Comparison Data (CD) approach.

Usage

```
CDF(
  response,
  num.trees = 500,
  mtry = "sqrt",
  nfact.max = 10,
  N.pop = 10000,
  N.Samples = 500,
  cor.type = "pearson",
  use = "pairwise.complete.obs",
  vis = TRUE,
  plot = TRUE
)
```

Arguments

response	A required N × I matrix or data.frame consisting of the responses of N individuals to × I items.
num.trees	the number of trees in the Random Forest. (default = 500) See details.
mtry	the maximum depth for each tree, can be a number or a character ("sqrt"). When mtry = "sqrt", it means that the maximum depth of each tree will be determined by the square root of the number of available features (converted to an integer by round). default = "sqrt". See details.
nfact.max	The maximum number of factors discussed by CDF approach. (default = 10)
N.pop	Size of finite populations of simulating. (default = 10,000)
N.Samples	Number of samples drawn from each population. (default = 500) Each Sample is consisted of a N × I matrix, just like the empirical data.
cor.type	A character string indicating which correlation coefficient (or covariance) is to be computed. One of "pearson" (default), "kendall", or "spearman". @seealso cor.
use	an optional character string giving a method for computing covariances in the presence of missing values. This must be one of the strings "everything", "all.obs", "complete.obs", "na.or.complete", or "pairwise.complete.obs" (default). @seealso cor.
vis	A Boolean variable that will print the factor retention results when set to TRUE, and will not print when set to FALSE. (default = TRUE)
plot	A Boolean variable that will print the CDF plot when set to TRUE, and will not print it when set to FALSE. @seealso plot.CDF. (Default = TRUE)

Details

The Comparison Data Forest (CDF; Goretzko & Ruscio, 2019) Approach is a combination of random forest with the Comparison Data (CD) approach. Its basic steps involve using the method of Ruscio & Roche (2012) to simulate data with different factor counts, then extracting features from this data to train a random forest model. Once the model is trained, it can be used to predict the number of factors in empirical data. The algorithm consists of the following steps:

1. **Simulation Data:**

(1) For each value of $nfact$ in the range from 1 to $nfact_{max}$, generate a population data using the GenData function.

(2) Each population ($N_{pop}I$) is based on $nfact$ factors and consists of N_{pop} observations.

(3) For each generated population, repeat the following for N_{sam} times, For the j-th: a. Draw a sample population NI from the population that matches the size of the empirical data; b. Compute a feature set $\mathbf{fea}_{nfact,j}$.

(4) Combine all the generated feature sets $\mathbf{fea}_{nfact,j}$ into a data frame as $\mathbf{data}_{train,nfact}$.

(5) Combine all $\mathbf{data}_{train,nfact}$ into a final data frame as the training datasets $\mathbf{data}_{train}$.

2. **Training RF:**

Train a Random Forest model RF using the combined $\mathbf{data}_{train}$.

3. **Prediction the Empirical Data:**

(1) Calculate the feature set $\mathbf{fea}_{emp}$for the empirical data.

(2) Use the trained Random Forest model RF to predict the number of factors $nfact_{emp}$ for the empirical data:

$$nfact_{emp} = RF(\mathbf{fea}_{emp})$$

According to Goretzko & Ruscio (2024) and Breiman (2001), the number of trees in the Random Forest `num.trees` is recommended to be 500. The Random Forest in CDF performs a classification task, so the recommended maximum depth for each tree `mtry` is $\sqrt{q}$ (where q is the number of features), which results in $m_{try} = \sqrt{181} = 13$.

Since the CDF approach requires extensive data simulation and computation, which is much more time consuming than the `CD` Approach, C++ code is used to speed up the process.

Value

An object of class `CDF` is a `list` containing the following components:

`nfact`	The number of factors to be retained.
`RF`	the trained Random Forest model
`probability`	A matrix containing the probabilities for factor numbers ranging from 1 to nfact.max (1xnfact.max), where the number in the f-th column represents the probability that the number of factors for the response is f.
`features`	A matrix (1×181) containing all the features for determining the number of factors. @seealso `extractor.feature.FF`

Author(s)

Haijiang Qin <Haijiang133@outlook.com>

References

Breiman, L. (2001). Random Forests. Machine Learning, 45(1), 5-32. https://doi.org/10.1023/A:1010933404324

Goretzko, D., & Ruscio, J. (2024). The comparison data forest: A new comparison data approach to determine the number of factors in exploratory factor analysis. Behavior Research Methods, 56(3), 1838-1851. https://doi.org/10.3758/s13428-023-02122-4

Ruscio, J., & Roche, B. (2012). Determining the number of factors to retain in an exploratory factor analysis using comparison data of known factorial structure. Psychological Assessment, 24, 282–292. http://dx.doi.org/10.1037/a0025697.

See Also

`GenData`

check_python_libraries *Check and Install Python Libraries (numpy and onnxruntime)*

Description

This function checks whether the Python (suggested >= 3.11) libraries ‘numpy‘ and ‘onnxruntime‘ are installed. If not, it will prompt the user to decide whether to install them. If the user chooses ’y’, the required library will be installed using the ‘reticulate‘ package. If the user chooses ’n’, the installation will be skipped. @seealso NN

Usage

```
check_python_libraries()
```

Value

A list indicating whether ‘numpy‘ and ‘onnxruntime‘ are installed. The list contains the following logical elements:

`numpy_installed`
: TRUE if ‘numpy‘ is installed, FALSE otherwise.

`onnxruntime_installed`
: TRUE if ‘onnxruntime‘ is installed, FALSE otherwise.

data.bfi *25 Personality Items Representing 5 Factors*

Description

This dataset includes 25 self-report personality items sourced from the International Personality Item Pool (ipip.ori.org) as part of the Synthetic Aperture Personality Assessment (SAPA) web-based personality assessment project. The dataset contains responses from 2,800 examinees. Additionally, three demographic variables (sex, education, and age) are included.

Format

A data frame with 2,800 observations on 28 variables. The variables include:

- `A1` - Am indifferent to the feelings of others. (q_146)
- `A2` - Inquire about others’ well-being. (q_1162)
- `A3` - Know how to comfort others. (g_1206)
- `A4` - Love children. (g_1364)
- `A5` - Make people feel at ease. (q_1419)

- C1 - Am exacting in my work. (q_124)
- C2 - Continue until everything is perfect. (q_530)
- C3 - Do things according to a plan. (q_619)
- C4 - Do things in a half-way manner. (g_626)
- C5 - Waste my time. (g_1949)
- E1 - Don't talk a lot. (q_712)
- E2 - Find it difficult to approach others. (q_901)
- E3 - Know how to captivate people. (q_1205)
- E4 - Make friends easily. (q_1410)
- E5 - Take charge. (g_1768)
- N1 - Get angry easily. (q_952)
- N2 - Get irritated easily. (q_974)
- N3 - Have frequent mood swings. (q_1099)
- N4 - Often feel blue. (g_1479)
- N5 - Panic easily. (q_1505)
- O1 - Am full of ideas. (q_128)
- O2 - Avoid difficult reading material. (g_316)
- O3 - Carry the conversation to a higher level. (q_492)
- O4 - Spend time reflecting on things. (g_1738)
- O5 - Will not probe deeply into a subject. (q_1964)
- gender - Gender: Males = 1, Females = 2
- education - Education level: 1 = High School, 2 = Finished High School, 3 = Some College, 4 = College Graduate, 5 = Graduate Degree
- age - Age in years

Details

The 25 items are organized by five factors: Agreeableness, Conscientiousness, Extraversion, Neuroticism, and Openness. The scoring key is created using `make.keys`, and scores are calculated using `score.items`. These factors are useful for IRT-based latent factor analysis of the polychoric correlation matrix. Endorsement plots and item information functions reveal variations in item quality. Responses were collected on a 6-point scale: 1 = Very Inaccurate, 2 = Moderately Inaccurate, 3 = Slightly Inaccurate, 4 = Slightly Accurate, 5 = Moderately Accurate, 6 = Very Accurate, as part of the Synthetic Aperture Personality Assessment (SAPA) project (https://www.sapa-project.org/). For examples of data collection techniques, visit https://www.sapa-project.org/ or the International Cognitive Ability Resource at https://icar-project.org. The items were sampled from the International Personality Item Pool of Lewis Goldberg using SAPA sampling techniques. This dataset is a sample from the larger SAPA data bank.

Note

The data.bfi data set and items should not be confused with the BFI (Big Five Inventory) of Oliver Johnand colleagues (John, O. P, Donahue, E. M., & Kentle, R.L. (1991). The Big Five Inventory Versions 4a and 54. Berkeley, CA: University of California, Berkeley, Institute of Personality and Social Research.)

Source

The items are from the ipip (Goldberg, 1999). The data are from the SAPA project (Revelle, Wiltand Rosenthal, 2010), collected Spring, 2010(https://www.sapa-project.org/).

References

Goldberg, L.R. (1999). A broad-bandwidth, public domain, personality inventory measuring the lower-level facets of several five-factor models. In Mervielde, I., Deary, I., De Fruyt, F., & Ostendorf, F. (Eds.), Personality psychology in Europe (Vol. 7, pp. 7-28). Tilburg University Press.

Revelle, W., Wilt, J., & Rosenthal, A. (2010). Individual Differences in Cognition: New Methods for Examining the Personality-Cognition Link. In Gruszka, A., Matthews, G., & Szymura, B. (Eds.), Handbook of Individual Differences in Cognition: Attention, Memory and Executive Control (pp. 117-144). Springer.

Revelle, W., Condon, D., Wilt, J., French, J.A., Brown, A., & Elleman, L.G. (2016). Web and phone-based data collection using planned missing designs. In Fielding, N.G., Lee, R.M., & Blank, G. (Eds.), SAGE Handbook of Online Research Methods (2nd ed., pp. 100-116). Sage Publications.

Examples

```
data(data.bfi)
head(data.bfi)
```

`data.DAPCS` *20-item Dependency-Oriented and Achievement-Oriented Psychological Control Scale (DAPCS)*

Description

This dataset contains responses to a 20-item Dependency-Oriented and Achievement-Oriented Psychological Control Scale (DAPCS), measuring four distinct factors of psychological control perceived by adolescents from their parents.

Details

The data were collected in 2022 from a sample of 987 general high school students in China. Among the participants, 406 were male and 581 were female, with a mean age of 15.823 years (SD = 0.793).

The DAPCS scale was developed by Soenens and Vansteenkiste (2010). It consists of 20 items that are grouped into four distinct dimensions, each with demonstrated internal consistency:

- **Autonomy – Negative Reaction**: Measures the extent of negative parental responses to adolescents' autonomy. *Reliability:* Cronbach's α = 0.857
- **Dependence – Positive Reaction**: Measures the extent of positive parental responses to adolescents' dependence. *Reliability:* Cronbach's α = 0.817

- **Low Achievement – Negative Reaction**: Measures the extent of negative parental responses to adolescents' low academic achievement. *Reliability:* Cronbach's α = 0.885
- **High Achievement – Positive Reaction**: Measures the extent of positive parental responses to adolescents' high academic achievement. *Reliability:* Cronbach's α = 0.889

The scale contains 20 items rated on a 5-point Likert scale, ranging from 1 = strongly disagree to 5 = strongly agree. In the dataset in this `EFAfactors` package, the total scale demonstrated a Cronbach's α of 0.923, and the four subscales showed Cronbach's α ranging from 0.817 to 0.889, indicating good reliability.

References

Soenens, B., & Vansteenkiste, M. (2010). A theoretical upgrade of the concept of parental psychological control: Proposing new insights on the basis of self-determination theory. Developmental Review, 30(1), 74–99.

Examples

```
data(data.DAPCS)
head(data.DAPCS)
```

`data.datasets.DNN` *Subset Dataset for Training the Deep Neural Network (DNN)*

Description

This dataset is a subset of the full datasets, consisting of 1,000 samples from the original 10,000,000-sample datasets.

Format

A 1,000×55 matrix, where the first 54 columns represent feature values and the last column represents the labels, which correspond to the number of factors associated with the features.

Note

Methods for generating and extracting features from the dataset can be found in `NN`.

See Also

`NN`, `load.scaler`, `data.scaler.DNN`, `normalizor`

Examples

```
data(data.datasets.DNN)
head(data.datasets.DNN)
```

data.datasets.LSTM *Subset Dataset for Training the Long Short Term Memory (LSTM) Network*

Description

This dataset is a subset of the full datasets, consisting of 1,000 samples from the original 1,000,000-sample datasets.

Format

A 1,000×21 matrix, where the firt column represents the labels and the last 20 columns represent feature values, which correspond to the number of factors associated with the features.

Note

Methods for generating and extracting features from the dataset can be found in NN.

See Also

NN, load.scaler, data.scaler.LSTM, normalizor

Examples

```
data(data.datasets.LSTM)
head(data.datasets.LSTM)
```

data.scaler.DNN *the Scaler for the pre-trained Deep Neural Network (DNN)*

Description

This dataset contains the means and standard deviations of the 10,000,000 datasets for training the Deep Neural Network (DNN), which can be used to determine the number of factors.

Format

A list containing two vectors, each of length 54:

means A numeric vector representing the means of the 54 features extracted from the 10,000,000 datasets.

sds A numeric vector representing the standard deviations of the 54 features extracted from the 10,000,000 datasets.

See Also

NN, load.scaler, data.datasets.DNN, normalizor

Examples

```
data(data.scaler.DNN)
print(data.scaler.DNN)

data.scaler <- load.scaler(model="DNN")
print(data.scaler)
```

---

data.scaler.LSTM *the Scaler for the pre-trained Long Short Term Memory (LSTM) Network*

Description

This dataset contains the means and standard deviations of the 1,000,000 datasets for training the Long Short Term Memory (LSTM) Network, which can be used to determine the number of factors.

Format

A `list` containing two `vectors`, each of length 20:

means A numeric vector representing the means of the 20 features extracted from the 1,000,000 datasets.

sds A numeric vector representing the standard deviations of the 20 features extracted from the 1,000,000 datasets.

See Also

NN, load.scaler, data.datasets.LSTM, normalizor

Examples

```
data(data.scaler.LSTM)
print(data.scaler.LSTM)

data.scaler <- load.scaler(model="LSTM")
print(data.scaler)
```

EFAhclust *Hierarchical Clustering for EFA*

Description

A function performs clustering on items by calling hclust. Hierarchical cluster analysis on a set of dissimilarities and methods for analyzing it. The items will be continuously clustered in pairs until all items are grouped into a single cluster, at which point the process will stop.

Usage

```
EFAhclust(
  response,
  dissimilarity.type = "R",
  method = "ward.D",
  nfact.max = 10,
  cor.type = "pearson",
  use = "pairwise.complete.obs",
  vis = TRUE,
  plot = TRUE
)
```

Arguments

response	A required N × I matrix or data.frame consisting of the responses of N individuals to I items.
dissimilarity.type	A character indicating which kind of dissimilarity is to be computed. One of "R" or "E" (default) for the correlation coefficient or Euclidean distance.
method	the agglomeration method to be used. This should be (an unambiguous abbreviation of) one of "ward.D", "ward.D2", "single", "complete", "average" (= UPGMA), "mcquitty" (= WPGMA), "median" (= WPGMC) or "centroid" (= UPGMC). (default = "ward.D") @seealso hclust
nfact.max	The maximum number of factors discussed. (default = 10)
cor.type	A character string indicating which correlation coefficient (or covariance) is to be computed. One of "pearson" (default), "kendall", or "spearman". @seealso cor.
use	an optional character string giving a method for computing covariances in the presence of missing values. This must be one of the strings "everything", "all.obs", "complete.obs", "na.or.complete", or "pairwise.complete.obs" (default). @seealso cor.
vis	A Boolean variable that will print the factor retention results when set to TRUE, and will not print when set to FALSE. (default = TRUE)
plot	A Boolean variable that will print the EFAhclust plot when set to TRUE, and will not print it when set to FALSE. @seealso plot.EFAhclust. (Default = TRUE)

Details

Hierarchical cluster analysis always merges the two nodes with the smallest dissimilarity, forming a new node in the process. This continues until all nodes are merged into one large node, at which point the algorithm terminates. This method undoubtedly creates a hierarchical structure by the end of the process, which encompasses the relationships between all items: items with high correlation have short connecting lines between them, while items with low correlation have longer lines. This hierarchical structure is well-suited to be represented as a binary tree. In this representation, the dissimilarity between two nodes can be indicated by the height of the tree nodes; the greater the difference between nodes, the higher the height of the tree nodes connecting them (the longer the line). Researchers can decide whether two nodes belong to the same cluster based on the height differences between nodes, which, in exploratory factor analysis, represents whether these two nodes belong to the same latent factor.

The Second-Order Difference (SOD) approach is a commonly used method for finding the "elbow" (the point of greatest slope change). According to the principles of exploratory factor analysis, items belonging to different latent factors have lower correlations, while items under the same factor are more highly correlated. In hierarchical clustering, this is reflected in the height of the nodes in the dendrogram, with differences in node heights representing the relationships between items. By sorting all node heights in descending order and applying the SOD method to locate the elbow, the number of factors can be determined. @seealso `EFAkmeans`

Value

An object of class `EFAhclust` is a `list` containing the following components:

`hc`	An object of class `hclust` that describes the tree produced by the clustering process. @seealso `hclust`
`cor.response`	A matrix of dimension `I × I` containing all the correlation coefficients of items.
`clusters`	A list containing all the clusters.
`heights`	A vector containing all the heights of the cluster tree. The heights are arranged in descending order.
`nfact.SOD`	The number of factors to be retained by the Second-Order Difference (SOD) approach.

References

Batagelj, V. (1988). Generalized Ward and Related Clustering Problems. In H. H. Bock, Classification and Related Methods of Data Analysis the First Conference of the International Federation of Classification Societies (IFCS), Amsterdam.

Murtagh, F., & Legendre, P. (2014). Ward's Hierarchical Agglomerative Clustering Method: Which Algorithms Implement Ward's Criterion? Journal of Classification, 31(3), 274-295. https://doi.org/10.1007/s00357-014-9161-z.

Examples

```
library(EFAfactors)
set.seed(123)
```

```
##Take the data.bfi dataset as an example.
data(data.bfi)

response <- as.matrix(data.bfi[, 1:25]) ## loading data
response <- na.omit(response) ## Remove samples with NA/missing values

## Transform the scores of reverse-scored items to normal scoring
response[, c(1, 9, 10, 11, 12, 22, 25)] <- 6 - response[, c(1, 9, 10, 11, 12, 22, 25)] + 1


## Run EFAhclust function with default parameters.

EFAhclust.obj <- EFAhclust(response)

plot(EFAhclust.obj)

## Get the heights.
heights <- EFAhclust.obj$heights
print(heights)

## Get the nfact retained by SOD
nfact.SOD <- EFAhclust.obj$nfact.SOD
print(nfact.SOD)
```

`EFAindex` *Various Indeces in EFA*

Description

A function performs clustering on items by calling `VSS` and `fa`. Apply the Very Simple Structure (VSS), Comparative Fit Index (CFI), MAP, and other criteria to determine the appropriate number of factors.

Usage

```
EFAindex(
  response,
  nfact.max = 10,
  cor.type = "cor",
  use = "pairwise.complete.obs"
)
```

Arguments

`response`	A required `N` × `I` matrix or data.frame consisting of the responses of `N` individuals to `I` items.

nfact.max	The maximum number of factors discussed by CD approach. (default = 10)
cor.type	How to find the correlations: "cor" is Pearson, "cov" is covariance, "tet" is tetrachoric, "poly" is polychoric, "mixed" uses mixed cor for a mixture of tetrachorics, polychorics, Pearsons, biserials, and polyserials, "Yuleb" is Yulebonett, "Yuleq" and "YuleY" are the obvious Yule coefficients as appropriate.
use	an optional character string giving a method for computing covariances in the presence of missing values. This must be one of the strings "everything", "all.obs", "complete.obs", "na.or.complete", or "pairwise.complete.obs" (default). @seealso cor.

Value

A matrix with the following components:

CFI the Comparative Fit Index

RMSEA Root Mean Square Error of Approximation (RMSEA) for each number of factors.

SRMR Standardized Root Mean Square Residual.

MAP Velicer's MAP values (lower values are better).

BIC Bayesian Information Criterion (BIC) for each number of factors.

SABIC Sample-size Adjusted Bayesian Information Criterion (SABIC) for each number of factors.

chisq Chi-square statistic from the factor analysis output.

df Degrees of freedom.

prob Probability that the residual matrix is greater than 0.

eChiSq Empirically found chi-square statistic.

eCRMS Empirically found mean residual corrected for degrees of freedom.

eBIC Empirically found BIC based on the empirically found chi-square statistic.

vss VSS fit with complexity 1.

sqresid Squared residual correlations.

fit Factor fit of the complete model.

Examples

```
library(EFAfactors)
set.seed(123)

##Take the data.bfi dataset as an example.
data(data.bfi)

response <- as.matrix(data.bfi[, 1:25]) ## loading data
response <- na.omit(response) ## Remove samples with NA/missing values

## Transform the scores of reverse-scored items to normal scoring
response[, c(1, 9, 10, 11, 12, 22, 25)] <- 6 - response[, c(1, 9, 10, 11, 12, 22, 25)] + 1
```

```
## Run EFAindex function with default parameters.

EFAindex.matrix <- EFAindex(response)

print(EFAindex.matrix)
```

`EFAkmeans` *K-means for EFA*

Description

A function performs K-means algorithm on items by calling kmeans.

Usage

```
EFAkmeans(response, nfact.max = 10, plot = TRUE)
```

Arguments

`response`	A required `N` × `I` matrix or data.frame consisting of the responses of `N` individuals to `I` items.
`nfact.max`	The maximum number of factors discussed by EFAkmeans approach. (default = 10)
`plot`	A Boolean variable that will print the EFAkmeans plot when set to TRUE, and will not print it when set to FALSE. @seealso `plot.EFAkmeans`. (Default = TRUE)

Details

K-means is a well-established and widely used classical clustering algorithm. It is an unsupervised machine learning algorithm that requires the number of clusters K to be specified in advance. After K-means terminates, the total within-cluster sum of squares (WSS) can be calculated to represent the goodness of fit of the clustering:

$$WSS = \sum_{\mathbf{C}_k \in \mathbf{C}} \sum_{i \in \mathbf{C}_k} \|i - \mu_k\|^2$$

where $\mathbf{C}$ is the set of all clusters. $\mathbf{C}_k$ is the k-th cluster. i represents each item in the cluster $\mathbf{C}_k$. μ_k is the centroid of cluster $\mathbf{C}_k$.

Similar to the scree plot where eigenvalues decrease as the number of factors increases, WSS also decreases as K increases. A "significant reduction" in WSS at a particular K may suggest that K is the most appropriate number of clusters, which in exploratory factor analysis implies that the number of factors is K. The "significant reduction" can be identified using the Second-Order Difference (SOD) approach. @seealso `EFAkmeans`

Value

An object of class `EFAkmeans` is a `list` containing the following components:

`wss`	A vector containing all within-cluster sum of squares (WSS).
`nfact.SOD`	The number of factors to be retained by the Second-Order Difference (SOD) approach.

Examples

```
library(EFAfactors)
set.seed(123)

##Take the data.bfi dataset as an example.
data(data.bfi)

response <- as.matrix(data.bfi[, 1:25]) ## loading data
response <- na.omit(response) ## Remove samples with NA/missing values

## Transform the scores of reverse-scored items to normal scoring
response[, c(1, 9, 10, 11, 12, 22, 25)] <- 6 - response[, c(1, 9, 10, 11, 12, 22, 25)] + 1


## Run EFAkmeans function with default parameters.

EFAkmeans.obj <- EFAkmeans(response)

plot(EFAkmeans.obj)

## Get the heights.
wss <- EFAkmeans.obj$wss
print(wss)

## Get the nfact retained by SOD
nfact.SOD <- EFAkmeans.obj$nfact.SOD
print(nfact.SOD)
```

EFAscreet *Scree Plot*

Description

This function generates a scree plot to display the eigenvalues of the correlation matrix computed from the given response data. The scree plot helps in determining the number of factors to retain in exploratory factor analysis by examining the point at which the eigenvalues start to level off, indicating less variance explained by additional factors.

Usage

```
EFAscreet(
  response,
  fa = "pc",
  nfact = 1,
  cor.type = "pearson",
  use = "pairwise.complete.obs"
)
```

Arguments

response	A required N × I matrix or data.frame consisting of the responses of N individuals to I items.
fa	A string that determines the method used to obtain eigenvalues. If 'pc', it represents Principal Component Analysis (PCA); if 'fa', it represents Principal Axis Factoring (a widely used Factor Analysis method; @seealso factor.analysis; Auerswald & Moshagen, 2019). (Default = 'pc')
nfact	A numeric value that specifies the number of factors to extract, only effective when fa = 'fa'. (Default = 1)
cor.type	A character string indicating which correlation coefficient (or covariance) is to be computed. One of "pearson" (default), "kendall", or "spearman". @seealso cor.
use	an optional character string giving a method for computing covariances in the presence of missing values. This must be one of the strings "everything", "all.obs", "complete.obs", "na.or.complete", or "pairwise.complete.obs" (default). @seealso cor.

Value

An object of class EFAscreet is a list containing the following components:

eigen.value	A vector containing the empirical eigenvalues

See Also

plot.EFAscreet

Examples

```
library(EFAfactors)
set.seed(123)

##Take the data.bfi dataset as an example.
data(data.bfi)

response <- as.matrix(data.bfi[, 1:25]) ## loading data
response <- na.omit(response) ## Remove samples with NA/missing values

## Transform the scores of reverse-scored items to normal scoring
```

```
response[, c(1, 9, 10, 11, 12, 22, 25)] <- 6 - response[, c(1, 9, 10, 11, 12, 22, 25)] + 1


## Run EFAscreet function with default parameters.

EFAscreet.obj <- EFAscreet(response)

plot(EFAscreet.obj)
```

`EFAsim.data` *Simulate Data that Conforms to the theory of Exploratory Factor Analysis.*

Description

This function is used to simulate data that conforms to the theory of exploratory factor analysis, with a high degree of customization for the variables involved.

Usage

```
EFAsim.data(
  nfact,
  vpf,
  N = 500,
  distri = "normal",
  fc = "R",
  pl = "R",
  cl = "R",
  low.vpf = 5,
  up.vpf = 15,
  a = NULL,
  b = NULL,
  vis = TRUE
)
```

Arguments

`nfact`	A numeric value specifying the number of factors to simulate.
`vpf`	A numeric or character value specifying the number of items under each factor. If a numeric value is provided, the numeric must be larger than 2, and the number of items under each factor will be fixed to this value. If a character value is provided, it must be one of 'S', 'M', 'L', or 'R'. These represent random selection of items under each factor from $U(5, 10)$, $U(5, 15)$, $U(5, 20)$, or $U(low.vpf up.vpf)$, respectively.

`N` A numeric value specifying the number of examinees to simulate.

`distri` A character, either 'normal' or 'beta', indicating whether the simulated data will follow a standard multivariate normal distribution or a multivariate beta distribution.

`fc` A numeric or character value specifying the degree of correlation between factors. If a numeric value is provided, it must be within the range of 0 to 0.75, and the correlation between all factors will be fixed at this value. If a character value is provided, it must be 'R', and the correlations between factors will be randomly selected from $U(0.0, 0.5)$.

`pl` A numeric or character value specifying the size of the primary factor loadings. If a numeric value is provided, it must be within the range of 0 to 1, and all primary factor loadings in the loading matrix will be fixed at this value. If a character value is provided, it must be one of 'L', 'M', 'H', or 'R', representing $pl\ U(0.35, 0.50)$, $pl\ U(0.50, 0.65)$, $pl\ U(0.65, 0.80)$, or $pl\ U(0.35, 0.80)$, respectively, consistent with the settings in Goretzko & Buhner (2020).

`cl` A numeric or character value specifying the size of cross-loadings. If a numeric value is provided, it must be within the range of 0 to 0.5, and all cross-loadings in the loading matrix will be fixed at this value. If a character value is provided, it must be one of 'L', 'H', 'None', or 'R', representing $cl\ U(-0.1, 0.1)$, $cl\ U(-0.2, -0.1) \cup U(0.1, 0.2)$, $cl = 0$, or $cl\ U(-0.2, 0.2)$, respectively, consistent with the settings in Auerswald & Moshagen (2019).

`low.vpf` A numeric value specifying the minimum number of items per factor, must be larger than 2, effective only when `vpf` is 'R'. (default = 5)

`up.vpf` A numeric value specifying the maximum number of items per factor, effective only when `vpf` is 'R'. (default = 15)

`a` A numeric or NULL specifying the 'a' parameter of the beta distribution, effective only when `distri = 'beta'`. If a numeric value is provided, it will be used as the 'a' parameter of the beta distribution. If NULL, a random integer between 1 and 10 will be used. (default = NULL)

`b` A numeric or NULL specifying the 'b' parameter of the beta distribution, effective only when `distri = 'beta'`. If a numeric value is provided, it will be used as the 'b' parameter of the beta distribution. If NULL, a random integer between 1 and 10 will be used. (default = NULL)

`vis` A logical value indicating whether to print process information. (default = `TRUE`)

Details

A population correlation matrix was created for each data set based on the following decomposition:

$$\mathbf{\Sigma} = \mathbf{\Lambda}\mathbf{\Phi}\mathbf{\Lambda}^T + \mathbf{\Delta}$$

where $\mathbf{\Lambda}$ is the loading matrix, $\mathbf{\Phi}$ is the factor correlation matrix, and $\mathbf{\Delta}$ is a diagonal matrix, with $\mathbf{\Delta} = 1 - \text{diag}(\mathbf{\Lambda}\mathbf{\Phi}\mathbf{\Lambda}^T)$. The purpose of $\mathbf{\Delta}$ is to ensure that the diagonal elements of $\mathbf{\Sigma}$ are 1.

The response data for each subject was simulated using the following formula:

$$X_i = L_i + \epsilon_i, \quad 1 \leq i \leq I$$

where L_i follows a a standard normal distribution (`distri = 'normal'`) or a beta distribution (`distri = 'beta'`), representing the contribution of latent factors. And ϵ_i is the residual term following a standard normal distribution (`distri = 'normal'`) or a beta distribution (`distri = 'beta'`) . L_i and ϵ_i are uncorrelated, and ϵ_i and ϵ_j are also uncorrelated.

Value

An object of class `EFAdata` is a `list` containing the following components:

`loadings`	A simulated loading matrix.
`items`	A `list` containing all factors and the item indices under each factor.
`cor.factors`	A simulated factor correlation matrix.
`cor.items`	A simulated item correlation matrix.
`response`	A simulated response data matrix.

References

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. https://doi.org/10.1037/met0000262.

Auerswald, M., & Moshagen, M. (2019). How to determine the number of factors to retain in exploratory factor analysis: A comparison of extraction methods under realistic conditions. Psychological methods, 24(4), 468-491. https://doi.org/https://doi.org/10.1037/met0000200

Examples

```
library(EFAfactors)

## Run EFAsim.data function with default parameters.
data.obj <- EFAsim.data(nfact = 3, vpf = 5, N=500, distri="normal", fc="R", pl="R", cl="R",
                        low.vpf = 5, up.vpf = 15, a = NULL, b = NULL, vis = TRUE)

head(data.obj$loadings)
```

`EFAvote` *Voting Method for Number of Factors in EFA*

Description

This function implements a voting method to determine the most appropriate number of factors in exploratory factor analysis (EFA). The function accepts a vector of votes, where each value represents the number of factors suggested by different EFA approaches. If there is a clear winner (a single number of factors with the most votes), that number is returned. In case of a tie, the function returns the first value among the tied results and outputs a message. The result is returned as an object of class `vote`, which can be printed and plotted.

Usage

```
EFAvote(votes, vis = TRUE, plot = TRUE)
```

Arguments

votes	A vector of integers, where each element corresponds to the number of factors suggested by an EFA method.
vis	Logical, whether to print the results of the voting. Defaults to TRUE.
plot	Logical, whether to display a pie chart of the voting results. Defaults to TRUE.

Value

An object of class EFAvote, which is a list containing:

nfact	The number of factors with the most votes. If there is a tie, the first one in the order is returned.
votes	The original vector of votes.

See Also

plot.EFAvote

Examples

```
library(EFAfactors)

nfacts <- c(5, 5, 5, 6, 6, 4)
names(nfacts) <- c("Hull", "CD", "PA", "EKC", "FF", "DNN")

EFAvote.obj <- EFAvote(votes = nfacts)

# Visualize the voting results
plot(EFAvote.obj)
```

EKC *Empirical Kaiser Criterion*

Description

This function will apply the Empirical Kaiser Criterion (Braeken & van Assen, 2017) method to determine the number of factors. The method assumes that the distribution of eigenvalues asymptotically follows a Marcenko-Pastur distribution (Marcenko & Pastur, 1967). It calculates the reference eigenvalues based on this distribution and determines whether to retain a factor by comparing the size of the empirical eigenvalues to the reference eigenvalues.

Usage

```
EKC(
  response,
  cor.type = "pearson",
  use = "pairwise.complete.obs",
  vis = TRUE,
  plot = TRUE
)
```

Arguments

response	A required N × I matrix or data.frame consisting of the responses of N individuals to I items.
cor.type	A character string indicating which correlation coefficient (or covariance) is to be computed. One of "pearson" (default), "kendall", or "spearman". @seealso cor.
use	an optional character string giving a method for computing covariances in the presence of missing values. This must be one of the strings "everything", "all.obs", "complete.obs", "na.or.complete", or "pairwise.complete.obs" (default). @seealso cor.
vis	A Boolean variable that will print the factor retention results when set to TRUE, and will not print when set to FALSE. (default = TRUE)
plot	A Boolean variable that will print the EKC plot when set to TRUE, and will not print it when set to FALSE. @seealso plot.EKC. (Default = TRUE)

Details

The Empirical Kaiser Criterion (EKC; Auerswald & Moshagen, 2019; Braeken & van Assen, 2017) refines Kaiser-Guttman Criterion by accounting for random sample variations in eigenvalues. At the population level, the EKC is equivalent to the original Kaiser-Guttman Criterion, extracting all factors whose eigenvalues from the correlation matrix are greater than one. However, at the sample level, it adjusts for the distribution of eigenvalues in normally distributed data. Under the null model, the eigenvalue distribution follows the Marčenko-Pastur distribution (Marčenko & Pastur, 1967) asymptotically. The upper bound of this distribution serves as the reference eigenvalue for the first eigenvalue λ, so

$$\lambda_{1,ref} = \left(1 + \sqrt{\frac{I}{N}}\right)^2$$

, which is determined by N individuals and I items. For subsequent eigenvalues, adjustments are made based on the variance explained by previous factors. The j-th reference eigenvalue is:

$$\lambda_{j,ref} = \max\left[\frac{I - \sum_{i=0}^{j-1}\lambda_i}{I - j + 1}\left(1 + \sqrt{\frac{I}{N}}\right)^2, 1\right]$$

The j-th reference eigenvalue is reduced according to the magnitude of earlier eigenvalues since higher previous values mean less unexplained variance remains. As in the original Kaiser-Guttman Criterion, the reference eigenvalue cannot drop below one.

$$F = \sum_{i=1}^{I} I(\lambda_i > \lambda_{i,ref})$$

Here, \(F \) represents the number of factors determined by the EKC, and $I(\cdot)$ is the indicator function, which equals 1 when the condition is true, and 0 otherwise.

Value

An object of class `EKC` is a `list` containing the following components:

`nfact`	The number of factors to be retained.
`eigen.value`	A vector containing the empirical eigenvalues
`eigen.ref`	A vector containing the reference eigenvalues

Author(s)

Haijiang Qin <Haijiang133@outlook.com>

References

Auerswald, M., & Moshagen, M. (2019). How to determine the number of factors to retain in exploratory factor analysis: A comparison of extraction methods under realistic conditions. Psychological methods, 24(4), 468-491. https://doi.org/10.1037/met0000200.

Braeken, J., & van Assen, M. A. L. M. (2017). An empirical Kaiser criterion. Psychological methods, 22(3), 450-466. https://doi.org/10.1037/met0000074.

Marc ˇenko, V. A., & Pastur, L. A. (1967). Distribution of eigenvalues for some sets of random matrices. Mathematics of the USSR-Sbornik, 1, 457–483. http://dx.doi.org/10.1070/SM1967v001n04ABEH001994

Examples

```
library(EFAfactors)
set.seed(123)

##Take the data.bfi dataset as an example.
data(data.bfi)

response <- as.matrix(data.bfi[, 1:25]) ## loading data
response <- na.omit(response) ## Remove samples with NA/missing values

## Transform the scores of reverse-scored items to normal scoring
response[, c(1, 9, 10, 11, 12, 22, 25)] <- 6 - response[, c(1, 9, 10, 11, 12, 22, 25)] + 1


## Run EKC function with default parameters.
```

```
EKC.obj <- EKC(response)

print(EKC.obj)

plot(EKC.obj)

## Get the eigen.value, eigen.ref and  nfact results.
eigen.value <- EKC.obj$eigen.value
eigen.ref <- EKC.obj$eigen.ref
nfact <- EKC.obj$nfact

print(eigen.value)
print(eigen.ref)
print(nfact)
```

`extractor.feature.FF` *Extracting features According to Goretzko & Buhner (2020)*

Description

This function will extract 181 features from the data according to the method by Goretzko & Buhner (2020).

Usage

```
extractor.feature.FF(
  response,
  cor.type = "pearson",
  use = "pairwise.complete.obs"
)
```

Arguments

`response`	A required `N` × `I` matrix or data.frame consisting of the responses of `N` individuals to `I` items.
`cor.type`	A character string indicating which correlation coefficient (or covariance) is to be computed. One of `"pearson"` (default), `"kendall"`, or `"spearman"`. @seealso `cor`.
`use`	an optional character string giving a method for computing covariances in the presence of missing values. This must be one of the strings `"everything"`, `"all.obs"`, `"complete.obs"`, `"na.or.complete"`, or `"pairwise.complete.obs"` (default). @seealso `cor`.

Details

The code for the `extractor.feature.FF` function is implemented based on the publicly available code by Goretzko & Buhner (2020) (https://osf.io/mvrau/). The extracted features are completely consistent with the 181 features described in the original text by Goretzko & Buhner (2020). These features include:

- `1.` - Number of examinees
- `2.` - Number of items
- `3.` - Number of eigenvalues greater than 1
- `4.` - Proportion of variance explained by the 1st eigenvalue
- `5.` - Proportion of variance explained by the 2nd eigenvalue
- `6.` - Proportion of variance explained by the 3rd eigenvalue
- `7.` - Number of eigenvalues greater than 0.7
- `8.` - Standard deviation of the eigenvalues
- `9.` - Number of eigenvalues accounting for 50
- `10.` - Number of eigenvalues accounting for 75
- `11.` - L1-norm of the correlation matrix
- `12.` - Frobenius-norm of the correlation matrix
- `13.` - Maximum-norm of the correlation matrix
- `14.` - Average of the off-diagonal correlations
- `15.` - Spectral-norm of the correlation matrix
- `16.` - Number of correlations smaller or equal to 0.1
- `17.` - Average of the initial communality estimates
- `18.` - Determinant of the correlation matrix
- `19.` - Measure of sampling adequacy (MSA after Kaiser, 1970)
- `20.` - Gini coefficient (Gini, 1921) of the correlation matrix
- `21.` - Kolm measure of inequality (Kolm, 1999) of the correlation matrix
- `22-101.` - Eigenvalues from Principal Component Analysis (PCA), padded with -1000 if insufficient
- `102-181.` - Eigenvalues from Factor Analysis (FA), fixed at 1 factor, padded with -1000 if insufficient

Value

A matrix (1×181) containing all the 181 features (Goretzko & Buhner, 2020).

References

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. Psychol Methods, 25(6), 776-786. https://doi.org/10.1037/met0000262.

`extractor.feature.NN` *Extracting features for the pre-trained Neural Networks for Determining the Number of Factors*

Description

This function is used to extract the features required by the pre-trained Neural Networks (DNN or LSTM) for Determining the Number of Factors. @seealso NN

Usage

```
extractor.feature.NN(
  response,
  model = "DNN",
  cor.type = "pearson",
  use = "pairwise.complete.obs"
)
```

Arguments

`response`	A required `N` × `I` matrix or data.frame consisting of the responses of `N` individuals to `I` items.
`model`	A character string indicating the model type. Possible values are `"DNN"` (default) or `"LSTM"`.
`cor.type`	A character string indicating which correlation coefficient (or covariance) is to be computed. One of `"pearson"` (default), `"kendall"`, or `"spearman"`. @seealso cor.
`use`	an optional character string giving a method for computing covariances in the presence of missing values. This must be one of the strings `"everything"`, `"all.obs"`, `"complete.obs"`, `"na.or.complete"`, or `"pairwise.complete.obs"` (default). @seealso cor.

Details

For `"DNN"`, a total of two types of features (6 kinds, making up 54 features in total) will be extracted, and they are as follows: 1. Clustering-Based Features

(1) Hierarchical clustering is performed with correlation coefficients as dissimilarity. The top 9 tree node heights are calculated, and all heights are divided by the maximum height. The heights from the 2nd to 9th nodes are used as features. @seealso EFAhclust

(2) Hierarchical clustering with Euclidean distance as dissimilarity is performed. The top 9 tree node heights are calculated, and all heights are divided by the maximum height. The heights from the 2nd to 9th nodes are used as features. @seealso EFAhclust

(3) K-means clustering is applied with the number of clusters ranging from 1 to 9. The within-cluster sum of squares (WSS) for clusters 2 to 9 are divided by the WSS for a single cluster. @seealso EFAkmeans

These three features are based on clustering algorithms. The purpose of division is to normalize the data. These clustering metrics often contain information unrelated to the number of factors, such as the number of items and the number of respondents, which can be avoided by normalization. The reason for using the 2nd to 9th data is that only the top F-1 data are needed to determine the number of factors F. The first data point is fixed at 1 after the division operations, so it is excluded. This approach helps in model simplification.

2. Traditional Exploratory Factor Analysis Features (Eigenvalues)

(4) The top 10 largest eigenvalues.

(5) The ratio of the top 10 largest eigenvalues to the corresponding reference eigenvalues from Empirical Kaiser Criterion (EKC; Braeken & van Assen, 2017). @seealso EKC

(6) The cumulative variance proportion of the top 10 largest eigenvalues.

Only the top 10 elements are used to simplify the model.

For "LSTM", a total of 2 types of features. These features are as follows:

(1) The top 10 largest eigenvalues.

(2) The difference of the top 10 largest eigenvalues to the corresponding reference eigenvalues from arallel Analysis (PA). @seealso PA

Value

A matrix (1×54 or 1×20) containing all the features for the DNN or LSTM.

Author(s)

Haijiang Qin <Haijiang133@outlook.com>

See Also

NN

Examples

```
library(EFAfactors)
set.seed(123)

##Take the data.bfi dataset as an example.
data(data.bfi)

response <- as.matrix(data.bfi[, 1:25]) ## loading data
response <- na.omit(response) ## Remove samples with NA/missing values

## Transform the scores of reverse-scored items to normal scoring
response[, c(1, 9, 10, 11, 12, 22, 25)] <- 6 - response[, c(1, 9, 10, 11, 12, 22, 25)] + 1


## Run extractor.feature.NN function.
features <- extractor.feature.NN(response, model="DNN")
```

```
print(features)

features <- extractor.feature.NN(response, model="LSTM")

print(features)
```

`factor.analysis` *Factor Analysis by Principal Axis Factoring*

Description

This function performs factor analysis using the Principal Axis Factoring (PAF) method. The process involves extracting factors from an initial correlation matrix and iteratively refining the factor estimates until convergence is achieved.

Usage

```
factor.analysis(
  data,
  nfact = 1,
  iter.max = 1000,
  criterion = 0.001,
  cor.type = "pearson",
  use = "pairwise.complete.obs"
)
```

Arguments

`data`	A data.frame or matrix of response If the matrix is square, it is assumed to be a correlation matrix. Otherwise, correlations (with pairwise deletion) will be computed.
`nfact`	The number of factors to extract. (default = 1)
`iter.max`	The maximum number of iterations for the factor extraction process. Default is 1000.
`criterion`	The convergence criterion for the iterative process. The extraction process will stop when the change in communalities is less than this value. Default is 0.001
`cor.type`	A character string indicating which correlation coefficient (or covariance) is to be computed. One of `"pearson"` (default), `"kendall"`, or `"spearman"`. @seealso `cor`.
`use`	an optional character string giving a method for computing covariances in the presence of missing values. This must be one of the strings `"everything"`, `"all.obs"`, `"complete.obs"`, `"na.or.complete"`, or `"pairwise.complete.obs"` (default). @seealso `cor`.

Details

The Principal Axis Factoring (PAF) method involves the following steps:

Step 1. **Basic Principle**: The core principle of factor analysis using Principal Axis Factoring (PAF) is expressed as:

$$\mathbf{R} = \mathbf{\Lambda}\mathbf{\Lambda}^T + \mathbf{\Phi}$$

$$R_{ii} = H_i^2 + \Phi_{ii}$$

where $\mathbf{\Lambda}$ is the matrix of factor loadings, and $\mathbf{\Phi}$ is the diagonal matrix of unique variances. Here, H_i^2 represents the portion of the i-th item's variance explained by the factor model. $\mathbf{H}^2$ reflects the amount of total variance in the variable accounted for by the factors in the model, indicating the explanatory power of the factor model for that variable.

Step 2. **Factor Extraction by Iteratoin**:

- Initial Communalities: Compute the initial communalities as the squared multiple correlations:

$$H_{i(t)}^2 = R_{ii(t)}$$

where $H_{i(t)}^2$ is the communality of i-th item in the t-th iteration, and $R_{ii(t)}$ is the i-th diagonal element of the correlation matrix in the t-th iteration.

- Extract Factors and Update Communalities:

$$\Lambda_{ij} = \sqrt{\lambda_j} \times v_{ij}$$

$$H_{i(t+1)}^2 = \sum_j \Lambda_{ij}^2$$

$$R_{ii(t+1)} = H_{i(t+1)}^2$$

where Λ_{ij} represents the j-th factor loading for the i-th item, λ_j is the j-th eigenvalue, $H_{i(t+1)}^2$ is the communality of i-th item in the $t+1$-th iteration, and v_{ij} is the j-th value of the i-th item in the eigen vector matrix $\mathbf{v}$.

Step 3. **Iterative Refinement**:

- Calculate the Change between $\mathbf{H}_t^2$ and $\mathbf{H}_{t+1}^2$:

$$\Delta H_i^2 = |H_{i(t+1)}^2 - H_{i(t)}^2|$$

where ΔH_i^2 represents the change in communalities between iterations t and $t+1$.

- Convergence Criterion: Continue iterating until the change in communalities is less than the specified criterion $criterion$:

$$\sum_i \Delta H_i^2 < criterion$$

The iterative process is implemented using C++ code to ensure computational speed.

Value

A list containing:

`loadings`	The extracted factor loadings.
`eigen.value`	The eigenvalues of the correlation matrix.
`H2`	A vector that contains the explanatory power of the factor model for all items.

Author(s)

Haijiang Qin <Haijiang133@outlook.com>

Examples

```
library(EFAfactors)
set.seed(123)

##Take the data.bfi dataset as an example.
data(data.bfi)

response <- as.matrix(data.bfi[, 1:25]) ## loading data
response <- na.omit(response) ## Remove samples with NA/missing values

## Transform the scores of reverse-scored items to normal scoring
response[, c(1, 9, 10, 11, 12, 22, 25)] <- 6 - response[, c(1, 9, 10, 11, 12, 22, 25)] + 1


## Run factor.analysis function to extract 5 factors

PAF.obj <- factor.analysis(response, nfact = 5)


## Get the loadings, eigen.value and  H2 results.
loadings <- PAF.obj$loadings
eigen.value <- PAF.obj$eigen.value
H2 <- PAF.obj$H2

print(loadings)
print(eigen.value)
print(H2)
```

FF *Factor Forest (FF) Powered by An Tuned XGBoost Model for Determining the Number of Factors*

Description

This function will invoke a tuned XGBoost model (Goretzko & Buhner, 2020; Goretzko, 2022; Goretzko & Ruscio, 2024) that can reliably perform the task of determining the number of factors. The maximum number of factors that the network can discuss is 8.

Usage

```
FF(
  response,
  cor.type = "pearson",
  use = "pairwise.complete.obs",
  vis = TRUE,
  plot = TRUE
)
```

Arguments

response	A required N × I matrix or data.frame consisting of the responses of N individuals to I items.
cor.type	A character string indicating which correlation coefficient (or covariance) is to be computed. One of "pearson" (default), "kendall", or "spearman". @seealso cor.
use	an optional character string giving a method for computing covariances in the presence of missing values. This must be one of the strings "everything", "all.obs", "complete.obs", "na.or.complete", or "pairwise.complete.obs" (default). @seealso cor.
vis	A Boolean variable that will print the factor retention results when set to TRUE, and will not print when set to FALSE. (default = TRUE)
plot	A Boolean variable that will print the FF plot when set to TRUE, and will not print it when set to FALSE. @seealso plot.FF. (Default = TRUE)

Details

A total of 500,000 datasets were simulated to extract features for training the tuned XGBoost model (Goretzko & Buhner, 2020; Goretzko, 2022). Each dataset was generated according to the following specifications:

- Factor number: *F* ~ U[1,8]
- Sample size: *N* ~ U[200,1000]
- Number of variables per factor: *vpf* ~ U[3,10]
- Factor correlation: *fc* ~ U[0.0,0.4]
- Primary loadings: *pl* ~ U[0.35,0.80]
- Cross-loadings: *cl* ~ U[0.0,0.2]

A population correlation matrix was created for each data set based on the following decomposition:

$$\mathbf{\Sigma} = \mathbf{\Lambda}\mathbf{\Phi}\mathbf{\Lambda}^T + \mathbf{\Delta}$$

where $\mathbf{\Lambda}$ is the loading matrix, $\mathbf{\Phi}$ is the factor correlation matrix, and $\mathbf{\Delta}$ is a diagonal matrix, with $\mathbf{\Delta} = 1 - \text{diag}(\mathbf{\Lambda}\mathbf{\Phi}\mathbf{\Lambda}^T)$. The purpose of $\mathbf{\Delta}$ is to ensure that the diagonal elements of $\mathbf{\Sigma}$ are 1.

The response data for each subject were simulated using the following formula:

$$X_i = L_i + \epsilon_i, \quad 1 \leq i \leq I$$

where L_i follows a normal distribution $N(0, \sigma)$, representing the contribution of latent factors, and ϵ_i is the residual term following a standard normal distribution. L_i and ϵ_i are uncorrelated, and ϵ_i and ϵ_j are also uncorrelated.

For each simulated dataset, a total of 184 features are extracted and compiled into a feature vector. These features include:

- `1.` - Number of examinees
- `2.` - Number of items
- `3.` - Number of eigenvalues greater than 1
- `4.` - Proportion of variance explained by the 1st eigenvalue
- `5.` - Proportion of variance explained by the 2nd eigenvalue
- `6.` - Proportion of variance explained by the 3rd eigenvalue
- `7.` - Number of eigenvalues greater than 0.7
- `8.` - Standard deviation of the eigenvalues
- `9.` - Number of eigenvalues accounting for 50
- `10.` - Number of eigenvalues accounting for 75
- `11.` - L1-norm of the correlation matrix
- `12.` - Frobenius-norm of the correlation matrix
- `13.` - Maximum-norm of the correlation matrix
- `14.` - Average of the off-diagonal correlations
- `15.` - Spectral-norm of the correlation matrix
- `16.` - Number of correlations smaller or equal to 0.1
- `17.` - Average of the initial communality estimates
- `18.` - Determinant of the correlation matrix
- `19.` - Measure of sampling adequacy (MSA after Kaiser, 1970)
- `20.` - Gini coefficient (Gini, 1921) of the correlation matrix
- `21.` - Kolm measure of inequality (Kolm, 1999) of the correlation matrix
- `21.` - Number of factors retained by the PA method @seealso PA
- `23.` - Number of factors retained by the EKC method @seealso EKC
- `24.` - Number of factors retained by the CD method @seealso CD
- `25-104.` - Eigenvalues from Principal Component Analysis (PCA), padded with -1000 if insufficient
- `105-184.` - Eigenvalues from Factor Analysis (FA), fixed at 1 factor, padded with -1000 if insufficient

The code for the `FF` function is implemented based on the publicly available code by Goretzko & Buhner (2020) (https://osf.io/mvrau/). The Tuned XGBoost Model is also obtained from this site. However, to meet the requirements for a streamlined R package, we can only save the core components of the Tuned XGBoost Model. Although these non-core parts do not affect performance, they include a lot of information about the model itself, such as the number of features, subsets of samples, and data from the training process, among others. For the complete Tuned XGBoost Model, please download it from https://osf.io/mvrau/.

Value

An object of class FF is a list containing the following components:

nfact	The number of factors to be retained.
probability	A matrix containing the probabilities for factor numbers ranging from 1 to 8 (1x8), where the number in the f-th column represents the probability that the number of factors for the response is f.
features	A matrix (1×184) containing all the features for determining the number of factors by the tuned XGBoost Model.

References

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. Psychol Methods, 25(6), 776-786. https://doi.org/10.1037/met0000262.

Goretzko, D. (2022). Factor Retention in Exploratory Factor Analysis With Missing Data. Educ Psychol Meas, 82(3), 444-464. https://doi.org/10.1177/00131644211022031.

Examples

```
library(EFAfactors)
set.seed(123)

##Take the data.bfi dataset as an example.
data(data.bfi)

response <- as.matrix(data.bfi[, 1:25]) ## loading data
response <- na.omit(response) ## Remove samples with NA/missing values

## Transform the scores of reverse-scored items to normal scoring
response[, c(1, 9, 10, 11, 12, 22, 25)] <- 6 - response[, c(1, 9, 10, 11, 12, 22, 25)] + 1


## Run FF function with default parameters.
## Not run:
FF.obj <- FF(response)

print(FF.obj)

plot(FF.obj)

## Get the probability and nfact results.
probability <- FF.obj$probability
nfact <- FF.obj$nfact

print(probability)
print(nfact)


## End(Not run)
```

GenData *Simulating Data Following John Ruscio's RGenData*

Description

This function simulates data with $nfact$ factors based on empirical data. It represents the simulation data part of the CD function and the CDF function. This function improves upon GenDataPopulation in R package RGenData by utilizing C++ code to achieve faster data simulation.

Usage

```
GenData(
  response,
  nfact = 1,
  N.pop = 10000,
  Max.Trials = 5,
  lr = 1,
  cor.type = "pearson",
  use = "pairwise.complete.obs",
  isSort = FALSE
)
```

Arguments

response	A required N × I matrix or data.frame consisting of the responses of N individuals to I items.
nfact	The number of factors to extract in factor analysis. (default = 1)
N.pop	Size of finite populations for simulating. (default = 10,000)
Max.Trials	The maximum number of consecutive trials without obtaining a lower RMSR. (default = 5)
lr	The learning rate for updating the correlation matrix during iteration. (default = 1)
cor.type	A character string indicating which correlation coefficient (or covariance) is to be computed. One of "pearson" (default), "kendall", or "spearman". @seealso cor.
use	an optional character string giving a method for computing covariances in the presence of missing values. This must be one of the strings "everything", "all.obs", "complete.obs", "na.or.complete", or "pairwise.complete.obs" (default). @seealso cor.
isSort	Logical, determines whether the simulated data needs to be sorted in descending order. (default = FALSE)

Details

The core idea of `GenData` is to start with the empirical data's correlation matrix and iteratively approach data with `nfact` factors. Any value in the simulated data must come from the empirical data. The specific steps of `GenData` are as follows:

(1) Use the empirical data ($\mathbf{Y}_{emp}$) correlation matrix as the target, $\mathbf{R}_{targ}$.

(2) Simulate scores for $N.pop$ examinees on $nfact$ factors using a multivariate standard normal distribution:

$$\mathbf{S}_{(N.pop \times nfact)} \sim \mathcal{N}(0, 1)$$

Simulate noise for $N.pop$ examinees on I items:

$$\mathbf{U}_{(N.pop \times I)} \sim \mathcal{N}(0, 1)$$

(3) Initialize $\mathbf{R}_{temp} = \mathbf{R}_{targ}$, and set the minimum Root Mean Square Residual $RMSR_{min} =$ Inf. Start the iteration process.

(4) Extract `nfact` factors from $\mathbf{R}_{temp}$, and obtain the factor loadings matrix $\mathbf{L}_{shar}$. Ensure that the first element of $\mathbf{L}_{share}$ is positive to standardize the direction.

(5) Calculate the unique factor matrix $\mathbf{L}_{uniq,(I \times 1)}$:

$$L_{uniq,i} = \sqrt{1 - \sum_{j=1}^{nfact} L_{share,i,j}^2}$$

(6) Calculate the simulated data $\mathbf{Y}_{sim}$:

$$Y_{sim,i,j} = \mathbf{S}_i \mathbf{L}_{shar,j}^T + U_{i,j} L_{uniq,j}$$

(7) Compute the correlation matrix of the simulated data, $\mathbf{R}_{simu}$.

(8) Calculate the residual correlation matrix $\mathbf{R}_{resi}$ between the target matrix $\mathbf{R}_{targ}$ and the simulated data's correlation matrix $\mathbf{R}_{simu}$:

$$\mathbf{R}_{resi} = \mathbf{R}_{targ} - \mathbf{R}_{simu}$$

(9) Calculate the current RMSR:

$$RMSR_{cur} = \sqrt{\frac{\sum_{i<j} \mathbf{R}_{resi,i,j}^2}{0.5 \times (I^2 - I)}}$$

(10) If $RMSR_{cur} < RMSR_{min}$, update $\mathbf{R}_{temp} = \mathbf{R}_{temp} + lr \times \mathbf{R}_{resi}$, $RMSR_{min} = RMSR_{cur}$, set $\mathbf{R}_{min,resi} = \mathbf{R}_{resi}$, and reset the count of consecutive trials without improvement $cou = 0$. If $RMSR_{cur} \geq RMSR_{min}$, update $\mathbf{R}_{temp} = \mathbf{R}_{temp} + 0.5 \times cou \times lr \times \mathbf{R}_{min,resi}$ and increment $cou = cou + 1$.

(11) Repeat steps (4) through (10) until $cou \geq Max.Trials$.

Of course C++ code is used to speed up.

Value

A `N.pop` * `I` matrix containing the simulated data.

References

Ruscio, J., & Roche, B. (2012). Determining the number of factors to retain in an exploratory factor analysis using comparison data of known factorial structure. Psychological Assessment, 24, 282–292. http://dx.doi.org/10.1037/a0025697.

`Hull` *the Hull Approach*

Description

The Hull method is a heuristic approach used to determine the optimal number of common factors in factor analysis. It evaluates models with increasing numbers of factors and uses goodness-of-fit indices relative to the model degrees of freedom to select the best-fitting model. The method is known for its effectiveness and reliability compared to other methods like the scree plot.

Usage

```
Hull(
  response,
  fa = "pc",
  nfact = 1,
  cor.type = "pearson",
  use = "pairwise.complete.obs",
  vis = TRUE,
  plot = TRUE
)
```

Arguments

`response`	A required `N` × `I` matrix or data.frame consisting of the responses of `N` individuals to `I` items.
`fa`	A string that determines the method used to obtain eigenvalues in PA. If `"pc"`, it represents Principal Component Analysis (PCA); if `"fa"`, it represents Principal Axis Factoring (a widely used Factor Analysis method; @seealso `factor.analysis`; Auerswald & Moshagen, 2019). (Default = `"pc"`)
`nfact`	A numeric value that specifies the number of factors to extract, only effective when `fa = 'fa'`. (Default = 1)
`cor.type`	A character string indicating which correlation coefficient (or covariance) is to be computed. One of `"pearson"` (default), `"kendall"`, or `"spearman"`. @seealso `cor`.
`use`	an optional character string giving a method for computing covariances in the presence of missing values. This must be one of the strings `"everything"`, `"all.obs"`, `"complete.obs"`, `"na.or.complete"`, or `"pairwise.complete.obs"` (default). @seealso `cor`.

`vis` A Boolean variable that will print the factor retention results when set to `TRUE`, and will not print when set to `FALSE`. (default = `TRUE`)

`plot` A Boolean variable that will print the Hull plot when set to `TRUE`, and will not print it when set to `FALSE`. @seealso `plot.Hull`. (Default = `TRUE`)

Details

The Hull method (Lorenzo-Seva & Timmerman, 2011) is a heuristic approach used to determ ine the number of common factors in factor analysis. This method is similar to non-graphical variants of Cattell's scree plot but relies on goodness-of-fit indices relative to the model degrees of freedom. The Hull method finds the optimal number of factors by following these steps:

1. Calculate the goodness-of-fit index (CFI) and model degrees of freedom (df; Lorenzo-Seva & Timmerman, 2011; $df = IF - 0.5F * (F - 1)$, I is the number of items, and F is the number of factors) for models with an increasing number of factors, up to a prespecified maximum, which is equal to the
2. nfact of `PA` method. the GOF will always be Comparative Fit Index (CFI), for it performs best under various conditions than other GOF (Auerswald & Moshagen, 2019; Lorenzo-Seva & Timmerman, 2011), such as RMSEA and SRMR. @seealso `EFAindex`
3. Identify and exclude solutions that are less complex (with fewer factors) but have a higher fit index.
4. Further exclude solutions if their fit indices fall below the line connecting adjacent viable solutions.
5. Determine the number of factors where the ratio of the difference in goodness-of-fit indices to the difference in degrees of freedom is maximized.

Value

A list with the following components:

`nfact` The optimal number of factors according to the Hull method.

`CFI` A numeric vector of CFI values for each number of factors considered.

`df` A numeric vector of model degrees of freedom for each number of factors considered.

`Hull.CFI` A numeric vector of CFI values with points below the convex Hull curve removed.

`Hull.df` A numeric vector of model degrees of freedom with points below the convex Hull curve removed.

Author(s)

Haijiang Qin <Haijiang133@outlook.com>

References

Auerswald, M., & Moshagen, M. (2019). How to determine the number of factors to retain in exploratory factor analysis: A comparison of extraction methods under realistic conditions. Psychological methods, 24(4), 468-491. https://doi.org/https://doi.org/10.1037/met0000200.

Lorenzo-Seva, U., Timmerman, M. E., & Kiers, H. A. L. (2011). The Hull Method for Selecting the Number of Common Factors. Multivariate Behavioral Research, 46(2), 340-364. https://doi.org/10.1080/00273171.2011.564

Examples

```
library(EFAfactors)
set.seed(123)

##Take the data.bfi dataset as an example.
data(data.bfi)

response <- as.matrix(data.bfi[, 1:25]) ## loading data
response <- na.omit(response) ## Remove samples with NA/missing values

## Transform the scores of reverse-scored items to normal scoring
response[, c(1, 9, 10, 11, 12, 22, 25)] <- 6 - response[, c(1, 9, 10, 11, 12, 22, 25)] + 1


## Run EKC function with default parameters.

Hull.obj <- Hull(response)

print(Hull.obj)

plot(Hull.obj)

## Get the CFI, df and  nfact results.
CFI <- Hull.obj$CFI
df <- Hull.obj$df
nfact <- Hull.obj$nfact

print(CFI)
print(df)
print(nfact)
```

KGC *Kaiser-Guttman Criterion*

Description

This function implements the Kaiser-Guttman criterion (Guttman, 1954; Kaiser, 1960) for determining the number of factors to retain in factor analysis. It is based on the eigenvalues of the correlation matrix of the responses. According to the criterion, factors are retained if their corresponding eigenvalues are greater than 1.

Usage

```
KGC(
  response,
  fa = "pc",
  nfact = 1,
  cor.type = "pearson",
  use = "pairwise.complete.obs",
  vis = TRUE,
  plot = TRUE
)
```

Arguments

response	A required N × I matrix or data.frame consisting of the responses of N individuals to I items.
fa	A string that determines the method used to obtain eigenvalues. If 'pc', it represents Principal Component Analysis (PCA); if 'fa', it represents Principal Axis Factoring (a widely used Factor Analysis method; @seealso factor.analysis; Auerswald & Moshagen, 2019). (Default = 'pc')
nfact	A numeric value that specifies the number of factors to extract, only effective when fa = 'fa'. (Default = 1)
cor.type	A character string indicating which correlation coefficient (or covariance) is to be computed. One of "pearson" (default), "kendall", or "spearman". @seealso cor.
use	an optional character string giving a method for computing covariances in the presence of missing values. This must be one of the strings "everything", "all.obs", "complete.obs", "na.or.complete", or "pairwise.complete.obs" (default). @seealso cor.
vis	A Boolean variable that will print the factor retention results when set to TRUE, and will not print when set to FALSE. (default = TRUE)
plot	A Boolean variable that will print the KGC plot when set to TRUE, and will not print it when set to FALSE. @seealso plot.KGC. (Default = TRUE)

Value

An object of class KGC is a list containing the following components:

nfact	The number of factors to be retained.
eigen.value	A vector containing the empirical eigenvalues

References

Guttman, L. (1954). Some necessary conditions for common-factor analysis. Psychometrika, 19, 149–161. http://dx.doi.org/10.1007/BF02289162.

Kaiser, H. F. (1960). The application of electronic computers to factor analysis. Educational and Psychological Measurement, 20, 141–151. http://dx.doi.org/10.1177/001316446002000116.

Examples

```
library(EFAfactors)
set.seed(123)

##Take the data.bfi dataset as an example.
data(data.bfi)

response <- as.matrix(data.bfi[, 1:25]) ## loading data
response <- na.omit(response) ## Remove samples with NA/missing values

## Transform the scores of reverse-scored items to normal scoring
response[, c(1, 9, 10, 11, 12, 22, 25)] <- 6 - response[, c(1, 9, 10, 11, 12, 22, 25)] + 1


## Run KGC function with default parameters.

KGC.obj <- KGC(response)

print(KGC.obj)

plot(KGC.obj)

## Get the eigen.value, eigen.ref and  nfact results.
eigen.value <- KGC.obj$eigen.value
nfact <- KGC.obj$nfact

print(eigen.value)
print(nfact)
```

`load.NN` *Load the the pre-trained Neural Networks for Determining the Number of Factors*

Description

Loads the pre-trained Deep Neural Network (DNN) from the `DNN.onnx` or Long Short Term Memory (LSTM) Network form `LSTM.onnx`. The function uses the `reticulate` package to import the `onnxruntime` Python library and create an inference session for the model.

Usage

```
load.NN(model = "DNN")
```

Arguments

model	A character string indicating the model type. Possible values are "DNN" (default) or "LSTM".

Value

An ONNX runtime inference session object for the DNN or LSTM model.

Note

Note that Python (suggested >= 3.11) and the libraries numpy and onnxruntime are required.

First, please ensure that Python is installed on your computer and that Python is included in the system's PATH environment variable. If not, please download and install it from the official website (https://www.python.org/).

If you encounter an error when running this function stating that the numpy and onnxruntime modules are missing:

```
Error in py_module_import(module, convert = convert) :
```

```
ModuleNotFoundError: No module named 'numpy'
```

or

```
Error in py_module_import(module, convert = convert) :
```

```
ModuleNotFoundError: No module named 'onnxruntime'
```

this means that the numpy or onnxruntime library is missing from your Python environment. If you are using Windows or macOS, please run the command `pip install numpy` or `pip install onnxruntime` in Command Prompt or Windows PowerShell (Windows), or Terminal (macOS). If you are using Linux, please ensure that `pip` is installed and use the command `pip install numpy` or `pip install onnxruntime` to install the missing libraries.

See Also

NN

---

load.scaler *Load the Scaler for the pre-trained Neural Networks for Determining the Number of Factors*

Description

Loads the scaler object within the `EFAfactors` package. This object is a `list` containing a mean vector and a standard deviation vector, which were computed from the 10,000,000 datasets `data.datasets.DNN` training the Deep Neural Network (DNN) or the 1,000,000 datasets `data.datasets.LSTM` training the Long Short Term Memory (LSTM) Network. It serves as a tool for normalizing features in `NN`.

Usage

```
load.scaler(model = "DNN")
```

Arguments

model	A character string indicating the model type. Possible values are "DNN" (default) or "LSTM".

Value

scaler objective.

See Also

NN, normalizor, data.scaler.DNN, data.scaler.LSTM

Examples

```
library(EFAfactors)

scaler <- load.scaler()
print(scaler)
```

---

load.xgb *Load the Tuned XGBoost Model*

Description

Loads the tuned XGBoost model object within the EFAfactors package into the global environment and retrieves it for use. Only the core model is retained to reduce the size.

Usage

```
load.xgb()
```

Value

The tuned XGBoost model object

Examples

```
library(EFAfactors)

xgb.model <- load.xgb()
print(xgb.model)
```

MAP *Minimum Average Partial (MAP) Test*

Description

The MAP test by Velicer (1976)was originally designed for determining the number of components in PCA but is used in EFA as well. At it score, the averaged partial correlations after excluding the variance that can be explained by the previous components are compared among different solutions and the one with the smallest value is chosen (therefore, MAP test).

Usage

```
MAP(
  response,
  fa = "pc",
  nfact.max = 10,
  cor.type = "pearson",
  use = "pairwise.complete.obs",
  vis = TRUE,
  plot = TRUE
)
```

Arguments

response	A required N × I matrix or data.frame consisting of the responses of N individuals to I items.
fa	A string that determines the method used to obtain factors loadings. If "pc", it represents Principal Component Analysis (PCA); if "fa", it represents Maximun Likelihhod Estimation (a widely used Factor Analysis method; @seealso fa; Auerswald & Moshagen, 2019). (Default = "pc")
nfact.max	The maximum number of factors discussed by MAP. (default = 10)
cor.type	A character string indicating which correlation coefficient (or covariance) is to be computed. One of "pearson" (default), "kendall", or "spearman". @seealso cor.
use	an optional character string giving a method for computing covariances in the presence of missing values. This must be one of the strings "everything", "all.obs", "complete.obs", "na.or.complete", or "pairwise.complete.obs" (default). @seealso cor.
vis	A Boolean variable that will print the factor retention results when set to TRUE, and will not print when set to FALSE. (default = TRUE)
plot	A Boolean variable that will print the MAP plot when set to TRUE, and will not print it when set to FALSE. @seealso plot.MAP. (Default = TRUE)

Value

An object of class MAP, which is a list containing the following components:

nfact	The number of factors to retain by both Optimal Coordinate and PA.
MAP.values	the averaged squared partial correlations for 1 to nfact.max factors.

References

Velicer, W. F. (1976). Determining the number of components from the matrix of partial correlations. Psychometrika, 41(3), 321–327. https://doi.org/10.1007/BF02293557

Goretzko, D. (2025). How many factors to retain in exploratory factor analysis? A critical overview of factor retention methods. Psychological methods, Advance online publication. https://doi.org/10.1037/met0000733

See Also

MAP

Examples

```
library(EFAfactors)
set.seed(123)

##Take the data.bfi dataset as an example.
data(data.bfi)

response <- as.matrix(data.bfi[, 1:25]) ## loading data
response <- na.omit(response) ## Remove samples with NA/missing values

## Transform the scores of reverse-scored items to normal scoring
response[, c(1, 9, 10, 11, 12, 22, 25)] <- 6 - response[, c(1, 9, 10, 11, 12, 22, 25)] + 1


 MAP.obj <- MAP(response, plot=FALSE)

 ## MAP plot
 plot(MAP.obj)
```

---

model.xgb *the Tuned XGBoost Model for Determining the Number of Facotrs*

Description

the Tuned XGBoost Model for Determining the Number of Facotrs

Format

An object of class TuneModel is the Tuned XGBoost Model for Determining the Number of Facotrs

See Also

FF, load.xgb

Examples

```
data(model.xgb)
print(model.xgb)

model.xgb <- load.xgb()
print(model.xgb)
```

NN *the pre-trained Neural Networks for Determining the Number of Factors*

Description

This function will invoke a pre-trained Neural Networks (DNN or LSTM) that can reliably perform the task of determining the number of factors. The maximum number of factors that the network can discuss is 10. The DNN model is implemented in Python and trained on PyTorch (https://pytorch.org/) with CUDA 11.8 for acceleration. The LSTM model is implemented in Python and trained on PyTorch (https://pytorch.org/) with CUDA 12.6 for acceleration. After training, the DNN and LSTM were saved as DNN.onnx and LSTM.onnx file. The NN function performs inference by loading the DNN.onnx or LSTM.onnx file in both Python and R environments. Therefore, please note that Python (suggested >= 3.11) and the libraries numpy and onnxruntime are required. @seealso check_python_libraries

To run this function, Python (suggested >= 3.11) is required, along with the installation of numpy and onnxruntime. See more in Details and Note.

Usage

```
NN(
  response,
  model = "DNN",
  cor.type = "pearson",
  use = "pairwise.complete.obs",
  vis = TRUE,
  plot = TRUE
)
```

Arguments

response	A required N × I matrix or data.frame consisting of the responses of N individuals to I items.
model	A character string indicating the model type. Possible values are "DNN" (default) or "LSTM".
cor.type	A character string indicating which correlation coefficient (or covariance) is to be computed. One of "pearson" (default), "kendall", or "spearman". @seealso cor.
use	an optional character string giving a method for computing covariances in the presence of missing values. This must be one of the strings "everything", "all.obs", "complete.obs", "na.or.complete", or "pairwise.complete.obs" (default). @seealso cor.
vis	A Boolean variable that will print the factor retention results when set to TRUE, and will not print when set to FALSE. (default = TRUE)
plot	A Boolean variable that will print the NN plot when set to TRUE, and will not print it when set to FALSE. @seealso plot.NN. (Default = TRUE)

Details

Due to the improved performance of DNN with larger datasets (Chen et al., 2017), a total of 10,000,000 datasets (data.datasets.DNN) were simulated to extract features for training deep learning neural networks. Each dataset was generated following the methods described by Auerswald & Moshagen (2019) and Goretzko & Buhner (2020), with the following specifications:

- Factor number: *F* ~ U[1,10]
- Sample size: *N* ~ U[100,1000]
- Number of variables per factor: *vpf* ~ [3,20]
- Factor correlation: *fc* ~ U[0.0,0.4]
- Primary loadings: *pl* ~ U[0.35,0.80]
- Cross-loadings: *cl* ~ U[-0.2,0.2]

A population correlation matrix was created for each data set based on the following decomposition:

$$\mathbf{\Sigma} = \mathbf{\Lambda}\mathbf{\Phi}\mathbf{\Lambda}^T + \mathbf{\Delta}$$

where $\mathbf{\Lambda}$ is the loading matrix, $\mathbf{\Phi}$ is the factor correlation matrix, and $\mathbf{\Delta}$ is a diagonal matrix, with $\mathbf{\Delta} = 1 - \text{diag}(\mathbf{\Lambda}\mathbf{\Phi}\mathbf{\Lambda}^T)$. The purpose of $\mathbf{\Delta}$ is to ensure that the diagonal elements of $\mathbf{\Sigma}$ are 1.

The response data for each subject was simulated using the following formula:

$$X_i = L_i + \epsilon_i, \quad 1 \leq i \leq I$$

where L_i follows a normal distribution $N(0, \sigma)$, representing the contribution of latent factors, and ϵ_i is the residual term following a standard normal distribution. L_i and ϵ_i are uncorrelated, and ϵ_i and ϵ_j are also uncorrelated.

For each simulated dataset, a total of 6 types of features (which can be classified into 2 types; @seealso extractor.feature.NN) are extracted and compiled into a feature vector, consisting of 54 features: 8 + 8 + 8 + 10 + 10 + 10. These features are as follows:

1. Clustering-Based Features

(1) Hierarchical clustering is performed with correlation coefficients as dissimilarity. The top 9 tree node heights are calculated, and all heights are divided by the maximum height. The heights from the 2nd to 9th nodes are used as features. @seealso EFAhclust

(2) Hierarchical clustering with Euclidean distance as dissimilarity is performed. The top 9 tree node heights are calculated, and all heights are divided by the maximum height. The heights from the 2nd to 9th nodes are used as features. @seealso EFAhclust

(3) K-means clustering is applied with the number of clusters ranging from 1 to 9. The within-cluster sum of squares (WSS) for clusters 2 to 9 are divided by the WSS for a single cluster. @seealso EFAkmeans

These three features are based on clustering algorithms. The purpose of division is to normalize the data. These clustering metrics often contain information unrelated to the number of factors, such as the number of items and the number of respondents, which can be avoided by normalization. The reason for using the 2nd to 9th data is that only the top F-1 data are needed to determine the number of factors F. The first data point is fixed at 1 after the division operations, so it is excluded. This approach helps in model simplification.

2. Traditional Exploratory Factor Analysis Features (Eigenvalues)

(4) The top 10 largest eigenvalues.

(5) The ratio of the top 10 largest eigenvalues to the corresponding reference eigenvalues from Empirical Kaiser Criterion (EKC; Braeken & van Assen, 2017). @seealso EKC

(6) The cumulative variance proportion of the top 10 largest eigenvalues.

Only the top 10 elements are used to simplify the model.

The DNN model is implemented in Python and trained on PyTorch (https://download.pytorch.org/whl/cu118) with CUDA 11.8 for acceleration. After training, the DNN was saved as a `DNN.onnx` file. The `NN` function performs inference by loading the `DNN.onnx` file in both Python and R environments.

And a total of 1,000,000 datasets (data.datasets.LSTM) were simulated to extract features for training LSTM. Each dataset was generated by:

- Factor number: F ~ U[1,10]
- Sample size: N ~ U[100,1000]
- Number of variables per factor: vpf ~ [3,10]
- Factor correlation: fc ~ U[0.0,0.5]
- Primary loadings: pl ~ U[0.35,0.80]
- Cross-loadings: cl ~ U[-0.2,0.2]

For each simulated dataset, a total of 2 types of features (@seealso `extractor.feature.NN`). These features are as follows:

(1) The top 10 largest eigenvalues.

(2) The difference of the top 10 largest eigenvalues to the corresponding reference eigenvalues from arallel Analysis (PA). @seealso `PA`

The LSTM model is implemented in Python and trained on PyTorch (https://download.pytorch.org/whl/cu126) with CUDA 12.6 for acceleration. After training, the LSTM was saved as a `LSTM.onnx` file. The `NN` function performs inference by loading the `LSTM.onnx` file in both Python and R environments.

Value

An object of class NN is a list containing the following components:

nfact	The number of factors to be retained.
features	A matrix (1×54 or 1×20) containing all the features for determining the number of factors by the DNN or LSTM.
probability	A matrix containing the probabilities for factor numbers ranging from 1 to 10 (1x10), where the number in the f-th column represents the probability that the number of factors for the response is f.

Note

Note that Python (suggested >= 3.11) and the libraries numpy and onnxruntime are required.

First, please ensure that Python is installed on your computer and that Python is included in the system's PATH environment variable. If not, please download and install it from the official website (https://www.python.org/).

If you encounter an error when running this function stating that the numpy and onnxruntime modules are missing:

```
Error in py_module_import(module, convert = convert) :
ModuleNotFoundError: No module named 'numpy'
```

or

```
Error in py_module_import(module, convert = convert) :
ModuleNotFoundError: No module named 'onnxruntime'
```

this means that the numpy or onnxruntime library is missing from your Python environment. The check_python_libraries function can help you install these two dependency libraries.

Of course, you can also choose not to use the check_python_libraries function. You can directly install the numpy or onnxruntime library using the appropriate commands. If you are using Windows or macOS, please run the command `pip install numpy` or `pip install onnxruntime` in Command Prompt or Windows PowerShell (Windows), or Terminal (macOS). If you are using Linux, please ensure that pip is installed and use the command `pip install numpy` or `pip install onnxruntime` to install the missing libraries.

Author(s)

Haijiang Qin <Haijiang133@outlook.com>

References

Auerswald, M., & Moshagen, M. (2019). How to determine the number of factors to retain in exploratory factor analysis: A comparison of extraction methods under realistic conditions. Psychological methods, 24(4), 468-491. https://doi.org/10.1037/met0000200.

Braeken, J., & van Assen, M. A. L. M. (2017). An empirical Kaiser criterion. Psychological methods, 22(3), 450-466. https://doi.org/10.1037/met0000074.

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. Psychol Methods, 25(6), 776-786. https://doi.org/10.1037/met0000262.

`normalizor` *Feature Normalization for the pre-trained Neural Networks for Determining the Number of Factors*

Description

This function normalizes a matrix of features using precomputed means and standard deviations. The function automatically runs load.scaler to read the standard deviations and means of the features, which are organized into a `list` object named `data.scaler.DNN` or `data.scaler.LSTM`. These means and standard deviations are computed from the 10,000,000 datasets `data.datasets.DNN` for training the pre-trained Deep Neural Network (DNN) or the 1,000,000 datasets `data.datasets.LSTM` for training the pre-trained Long Short Term Memory (LSTM) Network.

Usage

```
normalizor(features, model = "DNN")
```

Arguments

`features`	A numeric matrix where each row represents an observation and each column represents a feature.
`model`	A character string indicating the model type. Possible values are `"DNN"` (default) or `"LSTM"`. It determines which precomputed scaler (means and standard deviations) will be used. The scaler is loaded via `load.scaler` and should match the model used in subsequent analysis or prediction.

Details

The function applies z-score normalization to each element in the `features` matrix. It uses the `scaler` object, which is expected to contain precomputed means and standard deviations for each feature. The normalized value for each element is computed as:

$$z = \frac{x - \mu}{\sigma}$$

where x is the original value, μ is the mean, and σ is the standard deviation.

Value

A matrix of the same dimensions as `features`, where each feature has been normalized.

See Also

NN, `load.scaler`, `data.datasets.DNN`, `data.scaler.DNN`, `data.datasets.LSTM`, `data.scaler.LSTM`

PA *Parallel Analysis*

Description

This function performs Parallel Analysis (PA), which is a method used to determine the number of factors to retain in exploratory factor analysis. It compares the empirical eigenvalues with those obtained from simulated random data to identify the point where the observed eigenvalues are larger than those expected by chance. The number of empirical eigenvalues that are greater than the corresponding reference eigenvalues is the number of factors recommended to be retained by the PA method.

Usage

```
PA(
  response,
  fa = "pc",
  n.iter = 100,
  type = "quant",
  nfact = 1,
  quant = 0.95,
  cor.type = "pearson",
  use = "pairwise.complete.obs",
  vis = TRUE,
  plot = TRUE
)
```

Arguments

response	A required N × I matrix or data.frame consisting of the responses of N individuals to I items.
fa	A string that determines the method used to obtain eigenvalues in PA. If "pc", it represents Principal Component Analysis (PCA); if "fa", it represents Principal Axis Factoring (a widely used Factor Analysis method; @seealso factor.analysis; Auerswald & Moshagen, 2019). (Default = "pc")
n.iter	A numeric value that determines the number of simulations for the random data. (Default = 100)
type	A string that determines the method used to calculate the reference eigenvalues from the simulated data. If 'mean', the reference eigenvalue (eigen.ref) is the mean of the simulated eigenvalues (eigen.sim); if 'quant', the reference eigenvalue is the quant percentile of eigen.sim. (Default = 'quant')
nfact	A numeric value that specifies the number of factors to extract, only effective when fa = 'fa'. (Default = 1)
quant	A numeric value between 0 and 1, representing the quantile to be used for the reference eigenvalues calculation when type = 'quant'. (Default = 0.95)

cor.type	A character string indicating which correlation coefficient (or covariance) is to be computed. One of "pearson" (default), "kendall", or "spearman". @seealso cor.
use	an optional character string giving a method for computing covariances in the presence of missing values. This must be one of the strings "everything", "all.obs", "complete.obs", "na.or.complete", or "pairwise.complete.obs" (default). @seealso cor.
vis	A Boolean variable that will print the factor retention results when set to TRUE, and will not print when set to FALSE. (default = TRUE)
plot	A Boolean variable that will print the PA plot when set to TRUE, and will not print it when set to FALSE. @seealso plot.PA. (Default = TRUE)

Details

This function performs Parallel Analysis (PA; Horn, 1965; Auerswald & Moshagen, 2019) to determine the number of factors to retain. PA is a widely used method and is considered the "gold standard" for factor retention due to its high accuracy and stability, although it may underperform compared to methods like CD or EKC under certain conditions. The core idea of PA is to simulate random data multiple times, for example, 100 times, and compute the eigenvalues from each simulation. These simulated eigenvalues are then processed using either the mean or a quantile method to obtain the reference eigenvalues, such as the i-th reference eigenvalue $\lambda_{i,ref}$. The relationship between the i-th empirical eigenvalue λ_i and $\lambda_{i,ref}$ indicates whether the i-th factor should be retained. If $\lambda_i > \lambda_{i,ref}$, it suggests that the explanatory power of the i-th factor from the original data is stronger than that of the i-th factor from the random data, and therefore the factor should be retained. Conversely, if $\lambda_i <= \lambda_{i,ref}$, it indicates that the explanatory power of the i-th factor from the original data is weaker or equal to that of the random data, making it indistinguishable from noise, and thus the factor should not be retained. So,

$$F = \sum_{i=1}^{I} I(\lambda_i > \lambda_{i,ref})$$

Here, \(F \) represents the number of factors determined by the EKC, and $I(\cdot)$ is the indicator function, which equals 1 when the condition is true, and 0 otherwise.

Auerswald & Moshagen (2019) found that the most accurate results for PA were obtained when using PCA to extract eigenvalues and using the 95th percentile of the simulated eigenvalues to calculate the reference eigenvalues. Therefore, the recommended settings for this function are fa = 'pc', type = 'quant', and quant = 0.95.

Value

An object of class PA, which is a list containing the following components:

nfact	The number of factors to retain.
fa	Indicates the method used to obtain eigenvalues in PA. 'pc' represents Principal Component Analysis, and 'fa' represents Principal Axis Factoring.
type	Indicates the method used to calculate eigen.ref. If 'mean', eigen.ref is the mean of eigen.sim; if 'quant', eigen.ref is the quant percentile of eigen.sim.

eigen.value	A vector containing the empirical eigenvalues.
eigen.ref	A vector containing the reference eigenvalues, which depend on type.
eigen.sim	A matrix containing the simulated eigenvalues for all iterations.

Author(s)

Haijiang Qin <Haijiang133@outlook.com>

References

Auerswald, M., & Moshagen, M. (2019). How to determine the number of factors to retain in exploratory factor analysis: A comparison of extraction methods under realistic conditions. Psychological methods, 24(4), 468-491. https://doi.org/10.1037/met0000200.

Horn, J. L. (1965). A rationale and test for the number of factors in factor analysis. Psychometrika, 30, 179–185. http://dx.doi.org/10.1007/BF02289447.

Examples

```
library(EFAfactors)
set.seed(123)

##Take the data.bfi dataset as an example.
data(data.bfi)

response <- as.matrix(data.bfi[, 1:25]) ## loading data
response <- na.omit(response) ## Remove samples with NA/missing values

## Transform the scores of reverse-scored items to normal scoring
response[, c(1, 9, 10, 11, 12, 22, 25)] <- 6 - response[, c(1, 9, 10, 11, 12, 22, 25)] + 1


## Run PA function with default parameters.

PA.obj <- PA(response)

print(PA.obj)

plot(PA.obj)

## Get the eigen.value, eigen.ref and  nfact results.
eigen.value <- PA.obj$eigen.value
eigen.ref <- PA.obj$eigen.ref
nfact <- PA.obj$nfact

print(eigen.value)
print(eigen.ref)
print(nfact)
```

plot *Plot Methods*

Description

These 'plot' methods for different results, including:

- Hull plot for `Hull` results.
- Comparison Data plot for `CD` results.
- Parallel Analysis plot for `PA` results.
- Empirical Kaiser Criterion for `EKC` results.
- Kaiser-Guttman Criterion for `KGC` results.
- K-means plot for `EFAkmeans` results.
- Hierarchical Clustering plot for `EFAhclust` results.
- pre-trained Neural Networks plot for `NN` results.
- Factor Forest for `FF` results.
- Comparison Data Forest plot for `CDF` results.
- Voting Method plot for `EFAvote` results.
- Scree Plot for `EFAscreet` results.
- Minimum Average Partial Test plot for `MAP` results.
- Scree Test Optimal Coordinate plot for `STOC` results.

Usage

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

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

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

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

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

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

## S3 method for class 'EFAhclust'
```

```
plot(x, ...)

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

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

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

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

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

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

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

Arguments

x	An object of class Hull, CD, PA, EKC, KGC, EFAkmeans, EFAhclust, NN, FF, CDF, EFAvote, EFAscreet, MAP, or STOC.
...	Additional arguments passed to the plotting functions.

Value

None. Plots are produced as side effects.

Methods (by class)

- plot(Hull): Plot method for Hull objects
- plot(CD): Plot method for CD objects
- plot(PA): Plot method for PA objects
- plot(EKC): Plot method for EKC objects
- plot(KGC): Plot method for KGC objects
- plot(EFAkmeans): Plot method for EFAkmeans objects
- plot(EFAhclust): Plot method for EFAhclust objects
- plot(NN): Plot method for NN objects
- plot(FF): Plot method for FF objects
- plot(CDF): Plot method for CDF objects
- plot(EFAvote): Plot method for EFAvote objects

- `plot(EFAscreet)`: Plot method for EFAscreet objects
- `plot(MAP)`: Plot method for MAP objects
- `plot(STOC)`: Plot method for STOC objects

See Also

Hull, CD, PA, EKC, KGC, EFAkmeans, EFAhclust, NN, FF, CDF, EFAvote, EFAscreet, MAP, STOC

`predictLearner.classif.xgboost.earlystop` *Prediction Function for the Tuned XGBoost Model with Early Stopping*

Description

This function performs predictions using a trained XGBoost model with early stopping. The function itself does not have any specific purpose; its existence is solely to ensure the proper operation of FF.

Usage

```
## S3 method for class 'classif.xgboost.earlystop'
predictLearner(.learner, .model, .newdata, ...)
```

Arguments

`.learner`	An object representing the learner.
`.model`	The trained XGBoost model used to make predictions.
`.newdata`	A data frame or matrix containing new observations for which predictions are to be made.
`...`	Additional parameters passed to the `predict` function in XGBoost.

Value

A vector of predicted class labels or a matrix of predicted probabilities.

print *Print Methods*

Description

These 'print' methods for different results, including:

- Hull print for `Hull` results.
- Comparison Data print for `CD` results.
- Parallel Analysis print for `PA` results.
- Empirical Kaiser Criterion for `EKC` results.
- Kaiser-Guttman Criterion for `KGC` results.
- K-means print for `EFAkmeans` results.
- Hierarchical Clustering print for `EFAhclust` results.
- pre-trained Neural Networks print for `NN` results.
- Factor Forest for `FF` results.
- Comparison Data Forest print for `CDF` results.
- Voting Method print for `EFAvote` results.
- Scree print for `EFAscreet` results.
- Minimum Average Partial Test print for `MAP` results.
- Scree Test Optimal Coordinate print for `STOC` results.

Usage

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

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

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

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

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

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

## S3 method for class 'NN'
```

```
print(x, ...)

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

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

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

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

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

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

Arguments

x	An object of class Hull, CD, PA, EKC, KGC, EFAkmeans, EFAhclust, NN, FF, CDF, EFAvote, EFAscreet, MAP, or STOC.
...	Additional arguments passed to the printing functions.

Value

None. Prints are produced as side effects.

Methods (by class)

- print(Hull): Print method for Hull objects
- print(CD): Print method for CD objects
- print(PA): Print method for PA objects
- print(EKC): Print method for EKC objects
- print(KGC): Print method for KGC objects
- print(EFAhclust): Print method for EFAhclust objects
- print(NN): Print method for NN objects
- print(FF): Print method for FF objects
- print(CDF): Print method for CDF objects
- print(EFAvote): Print method for EFAvote objects
- print(EFAdata): Print method for EFAdata objects
- print(EFAscreet): Print method for EFAscreet objects
- print(MAP): Print method for MAP objects

See Also

Hull, CD, PA, EKC, KGC, EFAkmeans, EFAhclust, NN, FF, CDF, EFAvote, EFAscreet, MAP, STOC

STOC *Scree Test Optimal Coordinate (STOC)*

Description

The STOC (Raiche et al., 2013) approach is based on fitting $I - 2$ (I is the number of observed variables) two point regression models using the (i+1)th and I-th (the last one) eigenvalue to obtain a prediction for the i-th eigenvalue that is referred to as optimal coordinate (with p being the number of observed variables and therefore also the number of eigenvalues). If the observed empirical eigenvalue is larger than the predicted optimal coordinate. Raiche et al. (2013) also argue for a restricted version that retains only factors with an eigenvalue greater than the reference eigenvalue by PA (holds PCA and quant=0.95 in this package).

Usage

```
STOC(
  response,
  fa = "pc",
  nfact = 1,
  cor.type = "pearson",
  use = "pairwise.complete.obs",
  vis = TRUE,
  plot = TRUE
)
```

Arguments

response	A required N × I matrix or data.frame consisting of the responses of N individuals to I items.
fa	A string that determines the method used to obtain eigenvalues in PA. If "pc", it represents Principal Component Analysis (PCA); if "fa", it represents Principal Axis Factoring (a widely used Factor Analysis method; @seealso factor.analysis; Auerswald & Moshagen, 2019). (Default = "pc")
nfact	A numeric value that specifies the number of factors to extract, only effective when fa = 'fa'. (Default = 1)
cor.type	A character string indicating which correlation coefficient (or covariance) is to be computed. One of "pearson" (default), "kendall", or "spearman". @seealso cor.
use	an optional character string giving a method for computing covariances in the presence of missing values. This must be one of the strings "everything", "all.obs", "complete.obs", "na.or.complete", or "pairwise.complete.obs" (default). @seealso cor.

vis	A Boolean variable that will print the factor retention results when set to TRUE, and will not print when set to FALSE. (default = TRUE)
plot	A Boolean variable that will print the STOC plot when set to TRUE, and will not print it when set to FALSE. @seealso plot.STOC. (Default = TRUE)

Value

An object of class STOC, which is a list containing the following components:

nfact	The number of factors to retain by both Optimal Coordinate and PA.
nfact.STOC	The number of factors to retain only by Optimal Coordinate.
nfact.PA	The number of factors to retain only by PA.
fa	Indicates the method used to obtain eigenvalues. 'pc' represents Principal Component Analysis, and 'fa' represents Principal Axis Factoring.
eigen.value	A vector containing the empirical eigenvalues.
eigen.ref	A vector containing the reference eigenvalues by both Optimal Coordinate and PA.

References

Raiche, G., Walls, T. A., Magis, D., Riopel, M., & Blais, J.-G. (2013). Non-graphical solutions for Cattell's scree test. Methodology, 9(1), 23-29. https://doi.org/10.1027/1614-2241/a000051

Goretzko, D. (2025). How many factors to retain in exploratory factor analysis? A critical overview of factor retention methods. Psychological methods, Advance online publication. https://doi.org/10.1037/met0000733

See Also

STOC

Examples

```
library(EFAfactors)
set.seed(123)

##Take the data.bfi dataset as an example.
data(data.bfi)

response <- as.matrix(data.bfi[, 1:25]) ## loading data
response <- na.omit(response) ## Remove samples with NA/missing values

## Transform the scores of reverse-scored items to normal scoring
response[, c(1, 9, 10, 11, 12, 22, 25)] <- 6 - response[, c(1, 9, 10, 11, 12, 22, 25)] + 1


 STOC.obj <- STOC(response, plot=FALSE)

 ## STOC plot
 plot(STOC.obj)
```

Index

af.softmax, 3

CD, 4, *8*, *36*, *38*, *57*, *59, 60*, *62*
CDF, 6, *38*, *57*, *59, 60*, *62*
check_python_libraries, 9, *49*, *52*
cor, *4*, *7*, *15*, *18*, *21*, *26*, *28*, *30*, *32*, *35*, *38*, *40*, *43*, *47*, *50*, *55*, *62*

data.bfi, 9
data.DAPCS, 11
data.datasets.DNN, 12, *14*, *45*, *50*, *53*
data.datasets.LSTM, 13, *14*, *45*, *51*, *53*
data.scaler.DNN, *12*, 13, *46*, *53*
data.scaler.LSTM, *13*, 14, *46*, *53*

EFAhclust, 15, *30*, *51*, *57*, *59, 60*, *62*
EFAindex, 17, *41*
EFAkmeans, *16*, 19, *19*, *30*, *51*, *57*, *59, 60*, *62*
EFAscreet, 20, *57*, *59, 60*, *62*
EFAsim.data, 22
EFAvote, 24, *57*, *59, 60*, *62*
EKC, 25, *31*, *36*, *51*, *57*, *59, 60*, *62*
extractor.feature.FF, *8*, 28
extractor.feature.NN, 30, *50, 51*

fa, *17*, *47*
factor.analysis, *21*, 32, *40*, *43*, *54*, *62*
FF, 34, *49*, *57*, *59, 60*, *62*

GenData, *6–8*, 38
GenDataPopulation, *38*

hclust, *15, 16*
Hull, 40, *57*, *59, 60*, *62*

KGC, 42, *57*, *59, 60*, *62*
kmeans, *19*

load.NN, 44
load.scaler, *12–14*, 45, *53*
load.xgb, 46, *49*

MAP, 47, *48*, *57*, *59, 60*, *62*
model.xgb, 48

NN, *3*, *9*, *12–14*, *30, 31*, *45, 46*, 49, *53*, *57*, *59, 60*, *62*
normalizor, *12–14*, *46*, 53

PA, *31*, *36*, *41*, *51*, 54, *57*, *59, 60*, *62*
plot, 57
plot.CD, *4*
plot.CDF, *7*
plot.EFAhclust, *15*
plot.EFAkmeans, *19*
plot.EFAscreet, *21*
plot.EFAvote, *25*
plot.EKC, *26*
plot.FF, *35*
plot.Hull, *41*
plot.KGC, *43*
plot.MAP, *47*
plot.NN, *50*
plot.PA, *55*
plot.STOC, *63*
predictLearner.classif.xgboost.earlystop, *59*
print, 60

round, *7*

STOC, *57*, *59, 60*, 62, *62, 63*

VSS, *17*