Package ‘EFAtools’

November 4, 2020

Title Fast and Flexible Implementations of Exploratory Factor Analysis Tools

Version 0.3.0

Description Provides functions to perform exploratory factor analysis (EFA) procedures and compare their solutions. The goal is to provide state-of-the-art factor retention methods and a high degree of flexibility in the EFA procedures. This way, for example, implementations from R ‘psych’ and ‘SPSS’ can be compared. Moreover, functions for Schmid-Leiman transformation and the computation of omegas are provided. To speed up the analyses, some of the iterative procedures, like principal axis factoring (PAF), are implemented in C++.

Depends R (>= 3.6.0)

License GPL-3

Encoding UTF-8

LazyData true

URL https://github.com/mdsteiner/EFAtools

BugReports https://github.com/mdsteiner/EFAtools/issues

RoxygenNote 7.1.1

Imports lavaan, psych, crayon, stringr, stats, ggplot2, tibble, magrittr, dplyr, cli, Rcpp, viridisLite, graphics, future.apply, future, GPArotation, checkmate, tidyr, progressr, progress, rlang

LinkingTo Rcpp, RcppArmadillo

Suggests testthat, knitr, rmarkdown, microbenchmark

VignetteBuilder knitr

NeedsCompilation yes

Author Markus Steiner [aut, cre], Silvia Grieder [aut], William Revelle [ctb], Max Auerswald [ctb], Morten Moshagen [ctb], John Ruscio [ctb], Brendan Roche [ctb],
Urbano Lorenzo-Seva [ctb],
David Navarro-Gonzalez [ctb]

Maintainer Markus Steiner <markus.d.steiner@gmail.com>
Repository CRAN
Date/Publication 2020-11-04 18:00:02 UTC

R topics documented:

- .compute_vars .............................................. 3
- .factor_corres ........................................... 4
- .numformat .................................................. 4
- .paf_iter ................................................... 5
- .parallel_sim .............................................. 5
- BARTLETT .................................................... 6
- CD ............................................................ 7
- COMPARE ..................................................... 9
- DOSPERT ...................................................... 12
- DOSPERT_raw .............................................. 12
- EFA .......................................................... 13
- EFA_AVERAGE ................................................ 19
- EKC .......................................................... 24
- FACTOR_SCORES ............................................ 26
- GRiPS_raw ................................................... 27
- HULL .......................................................... 28
- IDS2_R ......................................................... 31
- KGC ........................................................... 32
- KMO ........................................................... 34
- N_FACTORS .................................................... 36
- OMEGA ........................................................ 39
- PARALLEL ...................................................... 43
- plot.CD ....................................................... 46
- plot.EFA_AVERAGE .......................................... 47
- plot.EKC ...................................................... 47
- plot.HULL ..................................................... 48
- plot.KGC ...................................................... 48
- plot.PARALLEL ............................................... 49
- plot.SCREE ................................................... 49
- population_models .......................................... 50
- print.BARTLETT ............................................. 52
- print.CD ...................................................... 53
- print.COMPARE ............................................ 53
- print.EFA ..................................................... 54
- print.EFA_AVERAGE ......................................... 54
- print.EKC ..................................................... 55
- print.HULL ................................................... 56
- print.KGC ..................................................... 56
- print.KMO ..................................................... 57
.compute_vars

Compute explained variances from loadings

Description

From unrotated loadings compute the communalities and uniquenesses for total variance. Compute explained variances per factor from rotated loadings (and factor intercorrelations Phi if oblique rotation was used).

Usage

.compute_vars(L_unrot, L_rot, Phi = NULL)

Arguments

L_unrot matrix. Unrotated factor loadings.
L_rot matrix. Rotated factor loadings.
Phi matrix. Factor intercorrelations. Provide only if oblique rotation is used.

Value

A matrix with sum of squared loadings, proportion explained variance from total variance per factor, same as previous but cumulative. Proportion of explained variance from total explained variance, and same as previous but cumulative.
**.factor_corres**

*Compute number of non-matching indicator-to-factor correspondences*

**Description**

Compute number of non-matching indicator-to-factor correspondences

**Usage**

`.factor_corres(x, y, thresh = 0.3)`

**Arguments**

- **x** numeric matrix. A matrix of pattern coefficients.
- **y** numeric matrix. A second matrix of coefficients.
- **thresh** numeric. The threshold to classify a pattern coefficient as substantial.

**.numformat**

*Format numbers for print method*

**Description**

Helper function used in the print method for class LOADINGS and SLLOADINGS. Strips the 0 in front of the decimal point of a number if number < 1, only keeps the first `digits` number of digits, and adds an empty space in front of the number if the number is positive. This way all returned strings (except for those > 1, which are exceptions in LOADINGS) have the same number of characters.

**Usage**

`.numformat(x, digits = 2, print_zero = FALSE)`

**Arguments**

- **x** numeric. Number to be formatted.
- **digits** numeric. Number of digits after the comma to keep.
- **print_zero** logical. Whether, if a number is between [-1, 1], the zero should be omitted or printed (default is FALSE, i.e. omit zeros).

**Value**

A formatted number
.paf_iter

Perform the iterative PAF procedure

Description

Function called from within PAF so usually no call to this is needed by the user. Provides a C++ implementation of the PAF procedure.

Usage

.paf_iter(h2, criterion, R, n_fac, abs_eig, crit_type, max_iter)

Arguments

- **h2**: numeric. The initial communality estimates.
- **criterion**: double. The convergence criterion to use.
- **R**: matrix. The correlation matrix with the initial communality estimates in the diagonal.
- **n_fac**: numeric. The number of factors to extract.
- **abs_eig**: logical. Whether absolute eigenvalues should be used to compute the loadings.
- **crit_type**: numeric. Whether maximum absolute differences (crit_type = 1), or sum of differences (crit_type = 2) should be used
- **max_iter**: numeric. The number of iterations after which to end the procedure if no convergence has been reached by then.

.parallel_sim

Parallel analysis on simulated data.

Description

Function called from within PARALLEL so usually no call to this is needed by the user. Provides a C++ implementation of the PARALLEL simulation procedure.

Usage

.parallel_sim(n_datasets, n_vars, N, eigen_type, maxit = 10000L)
Arguments

- **n_datasets**: numeric. Number of datasets with dimensions (N, n_vars) to simulate.
- **n_vars**: numeric. Number of variables / indicators in dataset.
- **N**: numeric. Number of cases / observations in dataset.
- **eigen_type**: numeric. Whether PCA (eigen_type = 1; i.e., leaving diagonal of correlation matrix at 1) or PAF (eigen_type = 2; i.e., setting diagonal of correlation matrix to SMCs).
- **maxit**: numeric. Maximum iterations to perform after which to abort.

BARTLETT

Bartlett’s test of sphericity

Description

This function tests whether a correlation matrix is significantly different from an identity matrix (Bartlett, 1951). If the Bartlett’s test is not significant, the correlation matrix is not suitable for factor analysis because the variables show too little covariance.

Usage

```r
BARTLETT(
  x,
  N = NA,
  use = c("pairwise.complete.obs", "all.obs", "complete.obs", "everything", "na.or.complete"),
  cor_method = c("pearson", "spearman", "kendall")
)
```

Arguments

- **x**: data.frame or matrix. Dataframe or matrix of raw data or matrix with correlations.
- **N**: numeric. The number of observations. Needs only be specified if a correlation matrix is used.
- **use**: character. Passed to `stats::cor` if raw data is given as input. Default is "pairwise.complete.obs".
- **cor_method**: character. Passed to `stats::cor`. Default is "pearson".

Details

Bartlett (1951) proposed this statistic to determine a correlation matrix’ suitability for factor analysis. The statistic is approximately chi square distributed with \[ df = \frac{p(p-1)}{2} \] and is given by

\[
\text{chi}^2 = \frac{-\log(\text{det}(R))}{N - 1 - (2 \cdot p + 5)/6}
\]
where \( \det(R) \) is the determinant of the correlation matrix, \( N \) is the sample size, and \( p \) is the number of variables.

This test requires multivariate normality. If this condition is not met, the Kaiser-Meyer-Olkin criterion \( \text{(KMO)} \) can still be used.

This function was heavily influenced by the \texttt{psych::cortest.bartlett} function from the psych package.

The \texttt{BARTLETT} function can also be called together with the \( \text{(KMO)} \) function and with factor retention criteria in the \texttt{N_FACTORS} function.

**Value**

A list containing

- \textit{chisq} The chi square statistic.
- \textit{p_value} The p value of the chi square statistic.
- \textit{df} The degrees of freedom for the chi square statistic.
- \textit{settings} A list of the settings used.

**Source**


**See Also**

- \texttt{KMO} for another measure to determine suitability for factor analysis.
- \texttt{N_FACTORS} as a wrapper function for this function, \texttt{KMO} and several factor retention criteria.

**Examples**

```
BARTLETT(test_models$baseline$cormat, N = 500)
```
Usage

```r
cD(
  x,
  n_factors_max = NA,
  N_pop = 10000,
  N_samples = 500,
  alpha = 0.3,
  use = c("pairwise.complete.obs", "all.obs", "complete.obs", "everything",
         "na.or.complete"),
  cor_method = c("pearson", "spearman", "kendall"),
  max_iter = 50
)
```

Arguments

- **x**: data.frame or matrix. Dataframe or matrix of raw data.
- **n_factors_max**: numeric. The maximum number of factors to test against. Larger numbers will increase the duration the procedure takes, but test more possible solutions. If left NA (default) the maximum number of factors for which the model is still over-identified (df > 0) is used.
- **N_pop**: numeric. Size of finite populations of comparison data. Default is 10000.
- **N_samples**: numeric. Number of samples drawn from each population. Default is 500.
- **alpha**: numeric. The alpha level used to test the significance of the improvement added by an additional factor. Default is .30.
- **use**: character. Passed to `stats::cor`. Default is "pairwise.complete.obs".
- **cor_method**: character. Passed to `stats::cor`. Default is "pearson".
- **max_iter**: numeric. The maximum number of iterations to perform after which the iterative PAF procedure is halted. Default is 50.

Details

"Parallel analysis (PA) is an effective stopping rule that compares the eigenvalues of randomly generated data with those for the actual data. PA takes into account sampling error, and at present it is widely considered the best available method. We introduce a variant of PA that goes even further by reproducing the observed correlation matrix rather than generating random data. Comparison data (CD) with known factorial structure are first generated using 1 factor, and then the number of factors is increased until the reproduction of the observed eigenvalues fails to improve significantly" (Ruscio & Roche, 2012, p. 282).

The CD implementation here is based on the code by Ruscio and Roche (2012), but is slightly adapted to increase speed by performing the principal axis factoring using a C++ based function.

The CD function can also be called together with other factor retention criteria in the `N_FACTORS` function.
Value

A list of class CD containing

- **n_factors**  The number of factors to retain according to comparison data results.
- **eigenvalues**  A vector containing the eigenvalues of the entered data.
- **RMSE_eigenvalues**  A matrix containing the RMSEs between the eigenvalues of the generated data and those of the entered data.
- **settings**  A list of the settings used.

Source


See Also

Other factor retention criteria: **EKC, HULL, KGC, PARALLEL, SMT**

**N_FACTORS** as a wrapper function for this and all the above-mentioned factor retention criteria.

Examples

```r
# determine n factors of the GRiPS
CD(GRiPS_raw)

# determine n factors of the DOSPERT risk subscale
CD(DOSPERT_raw)
```

**COMPARE**  
*Compare two vectors or matrices (communalities or loadings)*

Description

The function takes two objects of the same dimensions containing numeric information (loadings or communalities) and returns a list of class COMPARE containing summary information of the differences of the objects.
Usage

COMPARE(
  x,
  y,
  reorder = c("congruence", "names", "none"),
  corres = TRUE,
  thresh = 0.3,
  digits = 4,
  m_red = 0.001,
  range_red = 0.001,
  round_red = 3,
  print_diff = TRUE,
  na.rm = FALSE,
  x_labels = c("x", "y"),
  plot = TRUE,
  plot_red = 0.01
)

Arguments

x  matrix, or vector. Loadings or communalities of a factor analysis output.

y  matrix, or vector. Loadings or communalities of another factor analysis output to compare to x.

reorder  character. Whether and how elements / columns should be reordered. If "congruence" (default), reordering is done according to Tuckers correspondence coefficient, if "names", objects according to their names, if "none", no reordering is done.

corres  logical. Whether factor correspondences should be compared if a matrix is entered.

thresh  numeric. The threshold to classify a pattern coefficient as substantial. Default is .3.

digits  numeric. Number of decimals to print in the output. Default is 4.

m_red  numeric. Number above which the mean and median should be printed in red (i.e., if .001 is used, the mean will be in red if it is larger than .001, otherwise it will be displayed in green.) Default is .001.

range_red  numeric. Number above which the min and max should be printed in red (i.e., if .001 is used, min and max will be in red if the max is larger than .001, otherwise it will be displayed in green. Default is .001). Note that the color of min also depends on max, that is min will be displayed in the same color as max.

round_red  numeric. Number above which the max decimals to round to where all corresponding elements of x and y are still equal are displayed in red (i.e., if 3 is used, the number will be in red if it is smaller than 3, otherwise it will be displayed in green). Default is 3.

print_diff  logical. Whether the difference vector or matrix should be printed or not. Default is TRUE.
na.rm logical. Whether NAs should be removed in the mean, median, min, and max functions. Default is FALSE.

x_labels character. A vector of length two containing identifying labels for the two objects x and y that will be compared. These will be used as labels on the x-axis of the plot. Default is "x" and "y".

plot logical. If TRUE (default), a plot illustrating the differences will be shown.

plot_red numeric. Threshold above which to plot the absolute differences in red. Default is .001.

Value

A list of class COMPARE containing summary statistics on the differences of x and y.

diff The vector or matrix containing the differences between x and y.

mean_abs_diff The mean absolute difference between x and y.

median_abs_diff The median absolute difference between x and y.

min_abs_diff The minimum absolute difference between x and y.

max_abs_diff The maximum absolute difference between x and y.

max_dec The maximum number of decimals to which a comparison makes sense. For example, if x contains only values up to the third decimals, and y is a normal double, max_dec will be three.

are_equal The maximal number of decimals to which all elements of x and y are equal.

diff_corres The number of differing variable-to-factor correspondences between x and y, when only the highest loading is considered.

diff_corres_cross The number of differing variable-to-factor correspondences between x and y when all loadings >= thresh are considered.

g The root mean squared distance (RMSE) between x and y.

settings List of the settings used.

Examples

# A type SPSS EFA to mimick the SPSS implementation
EFA_SPSS_6 <- EFA(test_models$case_11b$cormat, n_factors = 6, type = "SPSS")

# A type psych EFA to mimick the psych::fa() implementation
EFA_psych_6 <- EFA(test_models$case_11b$cormat, n_factors = 6, type = "psych")

# compare the two
COMPARE(EFA_SPSS_6$unrot_loadings, EFA_psych_6$unrot_loadings, x_labels = c("SPSS", "psych"))
**DOSPERT**

**Description**

A list containing the bivariate correlations (cormat) of the 40 items of the Domain Specific Risk Taking Scale (DOSPERT; Weber, Blais, & Betz, 2002) and the sample size (N) based on the publicly available dataset at (https://osf.io/rce7g) of the Basel-Berlin Risk Study (Frey et al., 2017). The items measure risk-taking propensity on six different domains: social, recreational, gambling, health/safety, investment, and ethical.

**Usage**

DOSPERT

**Format**

An object of class list of length 2.

**Source**


https://osf.io/rce7g

---

**DOSPERT_raw**

**Description**


**Usage**

DOSPERT_raw

**Format**

An object of class data.frame with 3123 rows and 30 columns.
**Source**


---

**EFA**  
*Exploratory factor analysis (EFA)*

**Description**

This function does an EFA with either PAF, ML, or ULS with or without subsequent rotation. All arguments with default value NA can be left to default if type is set to one of “EFAtools”, “SPSS”, or “psych”. The respective specifications are then handled according to the specified type (see details). For all rotations except varimax and promax, the GPArotation package is needed.

**Usage**

```r
EFA(
  x,
  n_factors,
  N = NA,
  method = c("PAF", "ML", "ULS"),
  rotation = c("none", "varimax", "equamax", "quartimax", "geominT", "bentlerT",
               "bifactorT", "promax", "oblimin", "quadimin", "simplimax", "bentlerQ", "geominQ",
               "bifactorQ"),
  type = c("EFAtools", "psych", "SPSS", "none"),
  max_iter = NA,
  init_comm = NA,
  criterion = NA,
  criterion_type = NA,
  abs_eigen = NA,
  use = c("pairwise.complete.obs", "all.obs", "complete.obs", "everything",
          "na.or.complete"),
  varimax_type = NA,
  k = NA,
  normalize = TRUE,
  P_type = NA,
  precision = 1e-05,
  order_type = NA,
  start_method = "psych",
  cor_method = c("pearson", "spearman", "kendall"),
  ...
)
```

Arguments

- **x**: data.frame or matrix. Dataframe or matrix of raw data or matrix with correlations. If raw data is entered, the correlation matrix is found from the data.

- **n_factors**: numeric. Number of factors to extract.

- **N**: numeric. The number of observations. Needs only be specified if a correlation matrix is used. If input is a correlation matrix and $N = \text{NA}$ (default), not all fit indices can be computed.

- **method**: character. One of "PAF", "ML", or "ULS" to use principal axis factoring, maximum likelihood, or unweighted least squares (also called minres), respectively, to fit the EFA.

- **rotation**: character. Either perform no rotation ("none"; default), an orthogonal rotation ("varimax", "equamax", "quartimax", "geominT", "bentlerT", or "bifactorT"), or an oblique rotation ("promax", "oblimin", "quartimin", "simplicimax", "bentlerQ", "geominQ", or "bifactorQ").

- **type**: character. If one of "EFAtools" (default), "psych", or "SPSS" is used, and the following arguments with default NA are left with NA, these implementations are executed according to the respective program ("psych" and "SPSS") or according to the best solution found in Grieder & Steiner (2020; "EFAtools"). Individual properties can be adapted using one of the three types and specifying some of the following arguments. If set to "none" additional arguments must be specified depending on the method and rotation used (see details).

- **max_iter**: numeric. The maximum number of iterations to perform after which the iterative PAF procedure is halted with a warning. If type is one of "EFAtools", "SPSS", or "psych", this is automatically specified if max_iter is left to be $\text{NA}$, but can be overridden by entering a number. Default is NA.

- **init_comm**: character. The method to estimate the initial communalities in PAF. "smc" will use squared multiple correlations, "mac" will use maximum absolute correlations, "unity" will use 1s (see details). Default is NA.

- **criterion**: numeric. The convergence criterion used for PAF. If the change in communalities from one iteration to the next is smaller than this criterion the solution is accepted and the procedure ends. Default is NA.

- **criterion_type**: character. Type of convergence criterion used for PAF. "max_individual" selects the maximum change in any of the communalities from one iteration to the next and tests it against the specified criterion. This is also used by SPSS. "sum" takes the difference of the sum of all communalities in one iteration and the sum of all communalities in the next iteration and tests this against the criterion. This procedure is used by the psych::fa function. Default is NA.

- **abs_eigen**: logical. Which algorithm to use in the PAF iterations. If FALSE, the loadings are computed from the eigenvalues. This is also used by the psych::fa function. If TRUE the loadings are computed with the absolute eigenvalues as done by SPSS. Default is NA.

- **use**: character. Passed to stats::cor if raw data is given as input. Default is "pairwise.complete.obs".
varimax_type character. The type of the varimax rotation performed. If "svd", singular value decomposition is used, as stats::varimax does. If "kaiser", the varimax procedure performed in SPSS is used. This is the original procedure from Kaiser (1958), but with slight alterations in the varimax criterion (see details, and Grieder & Steiner, 2020). Default is NA.

k numeric. Either the power used for computing the target matrix P in the promax rotation or the number of 'close to zero loadings' for the simplimax rotation (see GPArotation::GPFoblq). If left to NA (default), the value for promax depends on the specified type. For simplimax, nrow(L), where L is the matrix of unrotated loadings, is used by default.

normalize logical. If TRUE, a kaiser normalization is performed before the specified rotation. Default is TRUE.

P_type character. This specifies how the target matrix P is computed in promax rotation. If "unnorm" it will use the unnormalized target matrix as originally done in Hendrickson and White (1964). This is also used in the psych and stats packages. If "norm" it will use the normalized target matrix as used in SPSS. Default is NA.

precision numeric. The tolerance for stopping in the rotation procedure. Default is 10^-5 for all rotation methods.

order_type character. How to order the factors. "eigen" will reorder the factors according to the largest to lowest eigenvalues of the matrix of rotated loadings. "ss_factors" will reorder the factors according to descending sum of squared factor loadings per factor. Default is NA.

start_method character. How to specify the starting values for the optimization procedure for ML. Default is "psych" which takes the starting values specified in psych::fa. "factanal" takes the starting values specified in the stats::factanal function. Solutions are very similar.

cor_method character. Passed to stats::cor. Default is "pearson".

... Additional arguments passed to rotation functions from the GPArotation package (e.g., maxit for maximum number of iterations).

Details

There are two main ways to use this function. The easiest way is to use it with a specified type (see above), which sets most of the other arguments accordingly. Another way is to use it more flexibly by explicitly specifying all arguments used and set type to "none" (see examples). A mix of the two can also be done by specifying a type as well as additional arguments. However, this will throw warnings to avoid unintentional deviations from the implementations according to the specified type.

The type argument is evaluated for PAF and for all rotations (mainly important for the varimax and promax rotations). The type-specific settings for these functions are detailed below.

For PAF, the values of init_comm, criterion, criterion_type, and abs_eigen depend on the type argument.

type = "EFAtools" will use the following argument specification: init_comm = "smc", criterion = .001, criterion_type = "sum", abs_eigen = TRUE.
type = "psych" will use the following argument specification: init_comm = "smc",criterion = .001,criterion_type = "sum",abs_eigen = FALSE.

type = "SPSS" will use the following argument specification: init_comm = "smc",criterion = .001,criterion_type = "max_individual",abs_eigen = TRUE.

If SMCs fail, SPSS takes "mac". However, as SPSS takes absolute eigenvalues, this is hardly ever the case. Psych, on the other hand, takes "unity" if SMCs fail. The EFAtools type setting combination was the best in terms of accuracy and number of Heywood cases compared to all the other setting combinations tested in simulation studies in Grieder & Steiner (2020), which is why this type is used as a default here.

For varimax, the values of varimax_type and order_type depend on the type argument.

- type = "EFAtools" will use the following argument specification: varimax_type = "svd",order_type = "eigen".
- type = "psych" will use the following argument specification: varimax_type = "svd",order_type = "eigen".
- type = "SPSS" will use the following argument specification: varimax_type = "kaiser",order_type = "ss_factors".

For promax, the values of P_type, order_type, and k depend on the type argument.

- type = "EFAtools" will use the following argument specification: P_type = "norm",order_type = "eigen",k = 4.
- type = "psych" will use the following argument specification: P_type = "unnorm",order_type = "eigen",k = 4.
- type = "SPSS" will use the following argument specification: P_type = "norm",order_type = "ss_factors",k = 4.

The P_type argument can take two values, "unnorm" and "norm". It controls which formula is used to compute the target matrix P in the promax rotation. "unnorm" uses the formula from Hendrickson and White (1964), specifically: $P = \frac{\text{abs}(A^{k+1})}{A}$, where A is the unnormalized matrix containing varimax rotated loadings. "SPSS" uses the normalized varimax rotated loadings. Specifically it used the following formula, which can be found in the SPSS 23 Algorithms manual: $P = \frac{\text{abs}(A / \sqrt{\text{rowSums}(A^2)})^{k+1} \times (\sqrt{\text{rowSums}(A^2)})}{A}$. As for PAF, the EFA-tools type setting combination for promax was the best compared to the other setting combinations tested in simulation studies in Grieder & Steiner (2020).

The varimax_type argument can take two values, "svd", and "kaiser". "svd" uses singular value decomposition, by calling stats::varimax. "kaiser" performs the varimax procedure as described in the SPSS 23 Algorithms manual and as described by Kaiser (1958). However, there is a slight alteration in computing the varimax criterion, which we found to better align with the results obtain from SPSS. Specifically, the original varimax criterion as described in the SPSS 23 Algorithms manual is $\sum(n \times \text{colSums}((\lambda \times 4) - \text{colSums}((\lambda \times 2) \times 2) / n^2)$, where n is the number of indicators, and lambda is the rotated loadings matrix. However, we found the following to produce results more similar to those of SPSS: $\text{sum(n \times \text{colSums(abs(lambda)) - colSums(lambda^4) \times 2) / n^2}$. For all other rotations except varimax and promax, the type argument only controls the order_type argument with the same values as stated above for the varimax and promax rotations. For these other rotations, the GPArotation package is needed. Additional arguments can also be specified and will be passed to the respective GPArotation function (e.g., maxit to change the maximum number of iterations for the rotation procedure).
The `type` argument has no effect on ULS and ML. For ULS, no additional arguments are needed. For ML, an additional argument `start_method` is needed to determine the starting values for the optimization procedure. Default for this argument is "factanal" which takes the starting values specified in the `stats::factanal` function.

**Value**

A list of class EFA containing (a subset of) the following:

- `orig_R` Original correlation matrix.
- `h2_init` Initial communality estimates from PAF.
- `h2` Final communality estimates from the unrotated solution.
- `orig_eigen` Eigen values of the original correlation matrix.
- `init_eigen` Initial eigenvalues, obtained from the correlation matrix with the initial communality estimates as diagonal in PAF.
- `final_eigen` Eigenvalues obtained from the correlation matrix with the final communality estimates as diagonal.
- `iter` The number of iterations needed for convergence.
- `convergence` Integer code for convergence as returned by `stats::optim` (only for ML and ULS). 0 indicates successful completion.
- `unrot_loadings` Loading matrix containing the final unrotated loadings.
- `vars_accounted` Matrix of explained variances and sums of squared loadings. Based on the unrotated loadings.
- `fit_indices` For ML and ULS: Fit indices derived from the unrotated factor loadings: Chi Square, including significance level, degrees of freedom (df), Comparative Fit Index (CFI), Root Mean Square Error of Approximation (RMSEA), including its 90% confidence interval, Akaike Information Criterion (AIC), Bayesian Information Criterion (BIC), and the common part accounted for (CAF) index as proposed by Lorenzo-Seva, Timmerman, & Kiers (2011). For PAF, only the CAF and dfs are returned.
- `rot_loadings` Loading matrix containing the final rotated loadings (pattern matrix).
- `Phi` The factor intercorrelations (only for oblique rotations).
- `Structure` The structure matrix (only for oblique rotations).
- `rotmat` The rotation matrix.
- `vars_accounted_rot` Matrix of explained variances and sums of squared loadings. Based on rotated loadings and, for oblique rotations, the factor intercorrelations.
- `settings` A list of the settings used.

**Source**


Examples

# A type EFAtools (as presented in Steiner and Grieder, 2020) EFA
EFAtools_PAF <- EFA(test_models$baseline$cormat, n_factors = 3, N = 500,
    type = "EFAtools", method = "PAF", rotation = "none")

# A type SPSS EFA to mimick the SPSS implementation (this will throw a warning,
# see below)
SPSS_PAF <- EFA(test_models$baseline$cormat, n_factors = 3, N = 500,
    type = "SPSS", method = "PAF", rotation = "none")

# A type psych EFA to mimick the psych::fa() implementation
psych_PAF <- EFA(test_models$baseline$cormat, n_factors = 3, N = 500,
    type = "psych", method = "PAF", rotation = "none")

# Use ML instead of PAF with type EFAtools
EFAtools_ML <- EFA(test_models$baseline$cormat, n_factors = 3, N = 500,
    type = "EFAtools", method = "ML", rotation = "none")

# Use oblimin rotation instead of no rotation with type EFAtools
EFAtools_oblim <- EFA(test_models$baseline$cormat, n_factors = 3, N = 500,
    type = "EFAtools", method = "PAF", rotation = "oblimin")

# Do a PAF without rotation without specifying a type, so the arguments
# can be flexibly specified (this is only recommended if you know what your
# doing)
PAF_none <- EFA(test_models$baseline$cormat, n_factors = 3, N = 500,
    type = "none", method = "PAF", rotation = "none",
    max_iter = 500, init_comm = "mac", criterion = 1e-4,
    criterion_type = "sum", abs_eigen = FALSE)

# Add a promax rotation
PAF_pro <- EFA(test_models$baseline$cormat, n_factors = 3, N = 500,
    type = "none", method = "PAF", rotation = "promax",
    max_iter = 500, init_comm = "mac", criterion = 1e-4,
    criterion_type = "sum", abs_eigen = FALSE, k = 3,
    P_type = "unnorm", precision = 1e-5, order_type = "eigen",
    varimax_type = "svd"
Model averaging across different EFA methods and types

Description

Not all EFA procedures always arrive at the same solution. This function allows you perform a number of EFAs from different methods (e.g., Maximum Likelihood and Principal Axis Factoring), with different implementations (e.g., the SPSS and psych implementations of Principal Axis Factoring), and across different rotations of the same type (e.g., multiple oblique rotations, like promax and oblimin). EFA_AVERAGE will then run all these EFAs (using the EFA function) and provide a summary across the different solutions.

Usage

EFA_AVERAGE(
  x,  
  n_factors,  
  N = NA,  
  method = "PAF",  
  rotation = "promax",  
  type = "none",  
  averaging = c("mean", "median"),  
  trim = 0,  
  salience_threshold = 0.3,  
  max_iter = 10000,  
  init_comm = c("smc", "mac", "unity"),  
  criterion = c(0.001),  
  criterion_type = c("sum", "max_individual"),  
  abs_eigen = c(TRUE),  
  varimax_type = c("svd", "kaiser"),  
  normalize = TRUE,  
  k_promax = 2:4,  
  k_simplimax = ncol(x),  
  P_type = c("norm", "unnorm"),  
  precision = 1e-05,  
  start_method = c("psych", "factanal"),  
  use = c("pairwise.complete.obs", "all.obs", "complete.obs", "everything", "na.or.complete"),  
  cor_method = c("pearson", "spearman", "kendall"),  
  show_progress = TRUE
)

Arguments

x data.frame or matrix. Dataframe or matrix of raw data or matrix with correlations. If raw data is entered, the correlation matrix is found from the data.
n_factors numeric. Number of factors to extract.
**N** numeric. The number of observations. Needs only be specified if a correlation matrix is used. If input is a correlation matrix and \( N = \text{NA} \) (default), not all fit indices can be computed.

**method** character vector. Any combination of "PAF", "ML", and "ULS", to use principal axis factoring, maximum likelihood, or unweighted least squares (also called minres), respectively, to fit the EFAs. Default is "PAF".

**rotation** character vector. Either perform no rotation ("none"), any combination of orthogonal rotations ("varimax", "equamax", "quartimax", "geominT", "bentlerT", and "bifactorT"; using "orthogonal” runs all of these), or of oblique rotations ("promax", "oblimin", "quartimin", "simplimax", "bentlerQ", "geominQ", and "bifactorQ"; using "oblique" runs all of these). Rotation types (no rotation, orthogonal rotations, and oblique rotations) cannot be mixed. Default is "promax".

**type** character vector. Any combination of "none" (default), "EFAtools", "psych", and "SPSS" can be entered. "none" allows the specification of various combinations of the arguments controlling both factor extraction methods and the rotations. The others ("EFAtools", "psych", and "SPSS"), control the execution of the respective factor extraction method and rotation to be in line with how it is executed in this package (i.e., the respective default procedure), in the psych package, and in SPSS. A specific psych implementation exists for PAF, ML, varimax, and promax. The SPSS implementation exists for PAF, varimax, and promax. For details, see EFA.

**averaging** character. One of "mean" (default), and "median". Controls whether the different results should be averaged using the (trimmed) mean, or the median.

**trim** numeric. If averaging is set to "mean", this argument controls the trimming of extremes (for details see base::mean). By default no trimming is done (i.e., trim = 0).

**salience_threshold** numeric. The threshold to use to classify a pattern coefficient or loading as salient (i.e., substantial enough to assign it to a factor). Default is 0.3. Indicator-to-factor correspondences will be inferred based on this threshold. Note that this may not be meaningful if rotation = "none" and n_factors > 1 are used, as no simple structure is present there.

**max_iter** numeric. The maximum number of iterations to perform after which the iterative PAF procedure is halted with a warning. Default is 10,000. Note that non-converged procedures are excluded from the averaging procedure.

**init_comm** character vector. Any combination of "smc", "mac", and "unity". Controls the methods to estimate the initial communalities in PAF if "none" is among the specified types. "smc" will use squared multiple correlations, "mac" will use maximum absolute correlations, "unity" will use 1s (for details see EFA). Default is c("smc", "mac", "unity").

**criterion** numeric vector. The convergence criterion used for PAF if "none" is among the specified types. If the change in communalities from one iteration to the next is smaller than this criterion the solution is accepted and the procedure ends. Default is 0.001.

**criterion_type** character vector. Any combination of "max_individual" and "sum". Type of convergence criterion used for PAF if "none" is among the specified types.
"max_individual" selects the maximum change in any of the communalities from one iteration to the next and tests it against the specified criterion. "sum" takes the difference of the sum of all communalities in one iteration and the sum of all communalities in the next iteration and tests this against the criterion (for details see EFA). Default is c("sum", "max_individual").

abs.eigen logical vector. Any combination of TRUE and FALSE. Which algorithm to use in the PAF iterations if "none" is among the specified types. If FALSE, the loadings are computed from the eigenvalues. This is also used by the psych::fa function. If TRUE the loadings are computed with the absolute eigenvalues as done by SPSS (for details see EFA). Default is TRUE.

varimax.type character vector. Any combination of "svd" and "kaiser". The type of the varimax rotation performed if "none" is among the specified types and "varimax", "promax", "orthogonal", or "oblique" is among the specified rotations. "svd" uses singular value decomposition, as stats::varimax does, and "kaiser" uses the varimax procedure performed in SPSS. This is the original procedure from Kaiser (1958), but with slight alterations in the varimax criterion (for details, see EFA and Grieder & Steiner, 2020). Default is c("svd", "kaiser").

normalize logical vector. Any combination of TRUE and FALSE. TRUE performs a kaiser normalization before the specified rotation(s). Default is TRUE.

k.promax numeric vector. The power used for computing the target matrix P in the promax rotation if "none" is among the specified types and "promax" or "oblique" is among the specified rotations. Default is 2:4.

k.simplimax numeric. The number of ‘close to zero loadings’ for the simplimax rotation (see GPArotation::GPFoblq) if "simplimax" or "oblique" is among the specified rotations. Default is ncol(x), where x is the entered data.

P.type character vector. Any combination of "norm" and "unnorm". This specifies how the target matrix P is computed in promax rotation if "none" is among the specified types and "promax" or "oblique" is among the specified rotations. "unnorm" will use the unnormalized target matrix as originally done in Hendrickson and White (1964). "norm" will use a normalized target matrix (for details see EFA). Default is c("norm", "unnorm").

precision numeric vector. The tolerance for stopping in the rotation procedure(s). Default is 10^-5.

start.method character vector. Any combination of "psych" and "factanal". How to specify the starting values for the optimization procedure for ML. "psych" takes the starting values specified in psych::fa. "factanal" takes the starting values specified in the stats::factanal function. Default is c("psych", "factanal").

use character. Passed to stats::cor if raw data is given as input. Default is "pairwise.complete.obs".

cor.method character. Passed to stats::cor. Default is "pearson".

show.progress logical. Whether a progress bar should be shown in the console. Default is TRUE.

Details

As a first step in this function, a grid is produced containing the setting combinations for the to-be-performed EFAs. These settings are then entered as arguments to the EFA function and the EFAs are
run in a second step. After all EFAs are run, the factor solutions are averaged and their variability
determined in a third step.

The grid containing the setting combinations is produced based on the entries to the respective
arguments. To this end, all possible combinations resulting in unique EFA models are considered.
That is, if, for example, the type argument was set to c("none","SPSS") and one combination
of the specific settings entered was identical to the SPSS combination, this combination would be
included in the grid and run only once. We include here a list of arguments that are only evaluated
under specific conditions:

The arguments init_comm, criterion, criterion_type, abs_eigen are only evaluated if "PAF"
is included in method and "none" is included in type.

The argument varimax_type is only evaluated if "varimax", "promax", "oblique", or "orthogonal"
is included in rotation and "none" is included in type.

The argument normalize is only evaluated if rotation is not set to "none" and "none" is included
in type.

The argument k_simplimax is only evaluated if "simplimax" or "oblique" is included in rotation.

The arguments k_promax and P_type are only evaluated if "promax" or "oblique" is included in
rotation and "none" is included in type.

The argument start_method is only evaluated if "ML" is included in method.

To avoid a bias in the averaged factor solutions from problematic solutions, these are excluded
prior to averaging. A solution is deemed problematic if at least one of the following is true: an
error occurred, the model did not converge, or there is at least one Heywood case (defined as a
loading or communality of >= .998). Information on errors, convergence, and Heywood cases are
returned in the implementations_grid and a summary of these is given when printing the output. In
addition to these, information on the admissibility of the factor solutions is also included. A solution
was deemed admissible if (1) no error occurred, (2) the model converged, (3) no Heywood cases
are present, and (4) there are at least two salient loadings (i.e., loadings exceeding the specified
salience_threshold) for each factor. So, solutions failing one of the first three of these criteria
admissibility are also deemed problematic and therefore excluded from averaging. However,
solutions failing only the fourth criterion of admissibility are still included for averaging. Finally, if
all solutions are problematic (e.g., all solutions contain Heywood cases), no averaging is performed
and the respective outputs are NA. In this case, the implementations_grid should be inspected to see
if there are any error messages, and the separate EFA solutions that are also included in the output
can be inspected as well, for example, to see where Heywood cases occurred.

A core output of this function includes the average, minimum, and maximum loadings derived from
all non-problematic (see above) factor solutions. Please note that these are not entire solutions,
but the matrices include the average, minimum, or maximum value for each cell (i.e., each loading
separately). This means that, for example, the matrix with the minimum loadings will contain
the minimum value in any of the factor solutions for each specific loading, and therefore most
likely contains loadings from different factor solutions. The matrices containing the minimum and
maximum factor solutions can therefore not be interpreted as whole factor solutions.

The output also includes information on the average, minimum, maximum, and variability of the
fit indices across the non-problematic factor solutions. It is important to note that not all fit indices
are computed for all fit methods: For ML and ULS, all fit indices can be computed, while for PAF,
only the common part accounted for (CAF) index (Lorenzo-Seva, Timmerman, & Kiers, 2011) can
be computed. As a consequence, if only "PAF" is included in the method argument, averaging can
only be performed for the CAF, and the other fit indices are NA. If a combination of "PAF" and "ML" and/or "ULS" are included in the method argument, the CAF is averaged across all non-problematic factor solutions, while all other fit indices are only averaged across the ML and ULS solutions. The user should therefore keep in mind that the number of EFAs across which the fit indices are averaged can diverge for the CAF compared to all other fit indices.

Value

A list of class EFA_AVERAGE containing

- **orig_R** Original correlation matrix.
- **h2** A list with the average, standard deviation, minimum, maximum, and range of the final communality estimates across the factor solutions.
- **loadings** A list with the average, standard deviation, minimum, maximum, and range of the final loadings across the factor solutions. If rotation was "none", the unrotated loadings, otherwise the rotated loadings (pattern coefficients).
- **Phi** A list with the average, standard deviation, minimum, maximum, and range of the factor intercorrelations across factor solutions obtained with oblique rotations.
- **ind_fac_corres** A matrix with each cell containing the proportion of the factor solutions in which the respective indicator-to-factor correspondence occurred, i.e., in which the loading exceeded the specified salience threshold. Note: Rowsums can exceed 1 due to cross-loadings.
- **vars_accounted** A list with the average, standard deviation, minimum, maximum, and range of explained variances and sums of squared loadings across the factor solutions. Based on the unrotated loadings.
- **fit_indices** A matrix containing the average, standard deviation, minimum, maximum, and range for all applicable fit indices across the respective factor solutions, and the degrees of freedom (df). If the method argument contains ML or ULS: Fit indices derived from the unrotated factor loadings: Chi Square (chisq), including significance level, Comparative Fit Index (CFI), Root Mean Square Error of Approximation (RMSEA), Akaike Information Criterion (AIC), Bayesian Information Criterion (BIC) and the common part accounted for (CAF) index as proposed by Lorenzo-Seva, Timmerman, & Kiers (2011). For PAF, only the CAF can be calculated (see details).
- **implementations_grid** A matrix containing, for each performed EFA, the setting combination, if an error occurred (logical), the error message (character), an integer code for convergence as returned by `stats::optim` (0 indicates successful completion.), if heywood cases occurred (logical, see details for definition), if the solution was admissible (logical, see details for definition), and the fit indices.
- **efa_list** A list containing the outputs of all performed EFAs. The names correspond to the rownames from the implementations_grid.
- **settings** A list of the settings used.
Source

Examples
# Averaging across different implementations of PAF and promax rotation (72 EFAs)
Aver_PAF <- EFA_AVERAGE(test_models$baseline$cormat, n_factors = 3, N = 500)

# Use median instead of mean for averaging (72 EFAs)
Aver_PAF_md <- EFA_AVERAGE(test_models$baseline$cormat, n_factors = 3, N = 500, averaging = "median")

# Averaging across different implementations of PAF and promax rotation,
# and across ULS and different versions of ML (108 EFAs)
Aver_meth_ext <- EFA_AVERAGE(test_models$baseline$cormat, n_factors = 3, N = 500, method = c("PAF", "ULS", "ML"))

# Averaging across one implementation each of PAF (EFAtools type), ULS, and
# ML with one implementation of promax (EFAtools type) (3 EFAs)
Aver_meth <- EFA_AVERAGE(test_models$baseline$cormat, n_factors = 3, N = 500, method = c("PAF", "ULS", "ML"), type = "EFAtools", start_method = "psych")

# Averaging across different oblique rotation methods, using one implementation
# of ML and one implementation of promax (EFAtools type) (7 EFAs)
Aver_rot <- EFA_AVERAGE(test_models$baseline$cormat, n_factors = 3, N = 500, method = "ML", rotation = "oblique", type = "EFAtools", start_method = "psych")

Empirical Kaiser Criterion

Description
The empirical Kaiser criterion incorporates random sampling variations of the eigenvalues from the Kaiser-Guttman criterion (KGC; see Auerswald & Moshagen, 2019; Braeken & van Assen, 2017). The code is based on Auerswald and Moshagen (2019).
Usage

EKC(
  x,
  N = NA,
  use = c("pairwise.complete.obs", "all.obs", "complete.obs", "everything",
         "na.or.complete"),
  cor_method = c("pearson", "spearman", "kendall")
)

Arguments

x  data.frame or matrix. data.frame or matrix of raw data or matrix with correlations.
N  numeric. The number of observations. Only needed if x is a correlation matrix.
use character. Passed to stats::cor if raw data is given as input. Default is "pairwise.complete.obs".
cor_method character. Passed to stats::cor. Default is "pearson".

Details

The Kaiser-Guttman criterion was defined with the intend that a factor should only be extracted if it explains at least as much variance as a single factor (see KGC). However, this only applies to population-level correlation matrices. Due to sampling variation, the KGC strongly overestimates the number of factors to retrieve (e.g., Zwick & Velicer, 1986). To account for this and to introduce a factor retention method that performs well with small number of indicators and correlated factors (cases where the performance of parallel analysis, see PARALLEL, is known to deteriorate) Braeken and van Assen (2017) introduced the empirical Kaiser criterion in which a series of reference eigenvalues is created as a function of the variables-to-sample-size ratio and the observed eigenvalues.

Braeken and van Assen (2017) showed that "(a) EKC performs about as well as parallel analysis for data arising from the null, 1-factor, or orthogonal factors model; and (b) clearly outperforms parallel analysis for the specific case of oblique factors, particularly whenever factor intercorrelation is moderate to high and the number of variables per factor is small, which is characteristic of many applications these days" (p.463-464).

The EKC function can also be called together with other factor retention criteria in the N_FACTORS function.

Value

A list of class EKC containing

eigenvalues  A vector containing the eigenvalues found on the correlation matrix of the entered data.
n_factors  The number of factors to retain according to the empirical Kaiser criterion.
references  The reference eigenvalues.
settings  A list with the settings used.
FACTOR_SCORES

Source


See Also

Other factor retention criteria: CD, HULL, KGC, PARALLEL, SMT

N_FACTORS as a wrapper function for this and all the above-mentioned factor retention criteria.

Examples

EKC(test_models$baseline$cormat, N = 500)

FACTOR_SCORES

Estimate factor scores for an EFA model

Description

This is a wrapper function for psych::factor.scores to be used directly with an output from EFA or by manually specifying the factor loadings and intercorrelations. Calculates factor scores according to the specified methods if raw data are provided, and only factor weights if a correlation matrix is provided.

Usage

FACTOR_SCORES(
  x,
  f,
  Phi = NULL,
  impute = c("none", "means", "median")
)

Arguments

x data.frame or matrix. Dataframe or matrix of raw data (needed to get factor scores) or matrix with correlations.

f object of class EFA or matrix.
**Phi**

A matrix of factor intercorrelations. Only needs to be specified if a factor loadings matrix is entered directly into \( f \). Default is NULL, in which case all intercorrelations are assumed to be zero.

**method**

Character. The method used to calculate factor scores. One of "Thurstone" (regression-based; default), "tenBerge", "Anderson", "Bartlett", "Harman", or "components". See `psych::factor.scores` for details.

**impute**

Character. Whether and how missing values in \( x \) should be imputed. One of "none" (default, only complete cases are scored), "median", or "mean".

**Value**

A list of class FACTOR_SCORES containing the following:

- **scores**
  The factor scores (only if raw data are provided.)

- **weights**
  The factor weights.

- **r.scores**
  The correlations of the factor score estimates.

- **missing**
  A vector of the number of missing observations per subject (only if raw data are provided).

- **R2**
  Multiple R2 of the scores with the factors.

- **settings**
  A list of the settings used.

**Examples**

```r
# Example with raw data with method "Bartlett" and no imputation
EFA_raw <- EFA(DOSPERT_raw, n_factors = 10, type = "EFAtools", method = "PAF",
               rotation = "oblimin")
fac_scores_raw <- FACTOR_SCORES(DOSPERT_raw, f = EFA_raw, method = "Bartlett",
                                impute = "none")

# Example with a correlation matrix (does not return factor scores)
EFA_cor <- EFA(test_models$baseline$cormat, n_factors = 3, N = 500,
               type = "EFAtools", method = "PAF", rotation = "oblimin")
fac_scores_cor <- FACTOR_SCORES(test_models$baseline$cormat, f = EFA_cor)
```

---

**GRiPS_raw**

**Description**

A data.frame containing responses to the General Risk Propensity Scale (GRiPS, Zhang, Highhouse & Nye, 2018) of 810 participants of Study 1 of Steiner and Frey (2020). The original data can be accessed via [https://osf.io/kxp8t/](https://osf.io/kxp8t/).

**Usage**

GRiPS_raw
Format

An object of class data.frame with 810 rows and 8 columns.

Source


---

**HULL**

_Hull method for determining the number of factors to retain_

Description


Usage

```r
HULL(
  x,
  N = NA,
  n_fac_theor = NA,
  method = c("PAF", "ULS", "ML"),
  gof = c("CAF", "CFI", "RMSEA"),
  eigen_type = c("SMC", "PCA", "EFA"),
  use = c("pairwise.complete.obs", "all.obs", "complete.obs", "everything", "na.or.complete"),
  cor_method = c("pearson", "spearman", "kendall"),
  n_datasets = 1000,
  percent = 95,
  decision_rule = c("means", "percentile", "crawford"),
  n_factors = 1,
  ...
)
```

Arguments

- **x** matrix or data.frame. Dataframe or matrix of raw data or matrix with correlations.
- **N** numeric. Number of cases in the data. This is passed to PARALLEL. Only has to be specified if x is a correlation matrix, otherwise it is determined based on the dimensions of x.
n_fac_theor numeric. Theoretical number of factors to retain. The maximum of this number and the number of factors suggested by PARALLEL plus one will be used in the Hull method.

method character. The estimation method to use. One of "PAF", "ULS", or "ML", for principal axis factoring, unweighted least squares, and maximum likelihood, respectively.

gof character. The goodness of fit index to use. Either "CAF", "CFI", or "RMSEA", or any combination of them. If method = "PAF" is used, only the CAF can be used as goodness of fit index. For details on the CAF, see Lorenzo-Seva, Timmerman, and Kiers (2011).

eigen_type character. On what the eigenvalues should be found in the parallel analysis. Can be one of "SMC", "PCA", or "EFA". If using "SMC" (default), the diagonal of the correlation matrices is replaced by the squared multiple correlations (SMCs) of the indicators. If using "PCA", the diagonal values of the correlation matrices are left to be 1. If using "EFA", eigenvalues are found on the correlation matrices with the final communalities of an EFA solution as diagonal. This is passed to PARALLEL.

use character. Passed to stats::cor if raw data is given as input. Default is "pairwise.complete.obs".

cor_method character. Passed to stats::cor. Default is "pearson".

n_datasets numeric. The number of datasets to simulate. Default is 1000. This is passed to PARALLEL.

percent numeric. A vector of percentiles to take the simulated eigenvalues from. Default is 95. This is passed to PARALLEL.

decision_rule character. Which rule to use to determine the number of factors to retain. Default is "means", which will use the average simulated eigenvalues. "percentile", uses the percentiles specified in percent. "crawford" uses the 95th percentile for the first factor and the mean afterwards (based on Crawford et al, 2010). This is passed to PARALLEL.

n_factors numeric. Number of factors to extract if "EFA" is included in eigen_type. Default is 1. This is passed to PARALLEL.

Details

The Hull method aims to find a model with an optimal balance between model fit and number of parameters. That is, it aims to retrieve only major factors (Lorenzo-Seva, Timmerman, & Kiers, 2011). To this end, it performs the following steps (Lorenzo-Seva, Timmerman, & Kiers, 2011, p.351):

1. It performs parallel analysis and adds one to the identified number of factors (this number is denoted \( J \)). \( J \) is taken as an upper bound of the number of factors to retain in the hull method. Alternatively, a theoretical number of factors can be entered. In this case \( J \) will be set to whichever of these two numbers (from parallel analysis or based on theory) is higher.

2. For all 0 to \( J \) factors, the goodness-of-fit (one of CAF, RMSEA, or CFI) and the degrees of freedom (\( df \)) are computed.
3. The solutions are ordered according to their \( df \).

4. Solutions that are not on the boundary of the convex hull are eliminated (see Lorenzo-Seva, Timmerman, & Kiers, 2011, for details).

5. All the triplets of adjacent solutions are considered consecutively. The middle solution is excluded if its point is below or on the line connecting its neighbors in a plot of the goodness-of-fit versus the degrees of freedom.

6. Step 5 is repeated until no solution can be excluded.

7. The \( st \) values of the “hull” solutions are determined.

8. The solution with the highest \( st \) value is selected.

The \texttt{PARALLEL} function and the principal axis factoring of the different number of factors can be parallelized using the future framework, by calling the \texttt{future::plan} function. The examples provide example code on how to enable parallel processing.

Note that if \( \text{gof} = \text{"RMSEA"} \) is used, \( 1 - \text{RMSEA} \) is actually used to compare the different solutions. Thus, the threshold of .05 is then .95. This is necessary due to how the heuristic to locate the elbow of the hull works.

The ML estimation method uses the \texttt{stats::factanal} starting values. See also the \texttt{EFA} documentation.

The \texttt{HULL} function can also be called together with other factor retention criteria in the \texttt{N_FACTORS} function.

\textbf{Value}

A list of class \texttt{HULL} containing the following objects

- \texttt{n_fac_CAF} The number of factors to retain according to the Hull method with the CAF.
- \texttt{n_fac_CFI} The number of factors to retain according to the Hull method with the CFI.
- \texttt{n_fac_RMSEA} The number of factors to retain according to the Hull method with the RMSEA.
- \texttt{solutions_CAF} A matrix containing the CAFs, degrees of freedom, and for the factors lying on the hull, the \( st \) values of the hull solution (see Lorenzo-Seva, Timmerman, and Kiers 2011 for details).
- \texttt{solutions_CFI} A matrix containing the CFIs, degrees of freedom, and for the factors lying on the hull, the \( st \) values of the hull solution (see Lorenzo-Seva, Timmerman, and Kiers 2011 for details).
- \texttt{solutions_RMSEA} A matrix containing the RMSEAs, degrees of freedom, and for the factors lying on the hull, the \( st \) values of the hull solution (see Lorenzo-Seva, Timmerman, and Kiers 2011 for details).
- \texttt{n_fac_max} The upper bound \( J \) of the number of factors to extract (see details).
- \texttt{settings} A list of the settings used.

\textbf{Source}

### IDS2_R

**Intelligence subtests from the Intelligence and Development Scales–2**

#### Description

A matrix containing the bivariate correlations of the 14 intelligence subtests from the Intelligence and Development Scales–2 (IDS-2; Grob & Hagmann-von Arx, 2018), an intelligence and development test battery for children and adolescents aged 5 to 20 years, for the standardization and validation sample (N = 1,991). Details can be found in Grieder & Grob (2019).

#### Usage

IDS2_R

#### Format

A 14 x 14 matrix of bivariate correlations

- **GS** (numeric) - Geometric shapes.
- **PL** (numeric) - Plates.
- **TC** (numeric) - Two characteristics.
**Kaiser-Guttman Criterion**

*Description*

Probably the most popular factor retention criterion. Kaiser and Guttman suggested to retain as many factors as there are sample eigenvalues greater than 1. This is why the criterion is also known as eigenvalues-greater-than-one rule.

*Usage*

```r
KGC(
  x,
  eigen_type = c("PCA", "SMC", "EFA"),
  use = c("pairwise.complete.obs", "all.obs", "complete.obs", "everything",
          "na.or.complete"),
  cor_method = c("pearson", "spearman", "kendall"),
  n_factors = 1,
  ...
)
```
Arguments

x data.frame or matrix. Dataframe or matrix of raw data or matrix with correlations.

eigen_type character. On what the eigenvalues should be found. Can be either "PCA", "SMC", or "EFA", or some combination of them. If using "PCA", the diagonal values of the correlation matrices are left to be 1. If using "SMC", the diagonal of the correlation matrices is replaced by the squared multiple correlations (SMCs) of the indicators. If using "EFA", eigenvalues are found on the correlation matrices with the final communalities of an exploratory factor analysis solution (default is principal axis factoring extracting 1 factor) as diagonal.

use character. Passed to stats::cor if raw data is given as input. Default is "pairwise.complete.obs".

cor_method character. Passed to stats::cor. Default is "pearson".

n_factors numeric. Number of factors to extract if "EFA" is included in eigen_type. Default is 1.

Additional arguments passed to EFA. For example, to change the extraction method (PAF is default).

Details

Originally, the Kaiser-Guttman criterion was intended for the use with prinicipal components, hence with eigenvalues derived from the original correlation matrix. This can be done here by setting eigen_type to "PCA". However, it is well-known that this criterion is often inaccurate and that it tends to overestimate the number of factors, especially for unidimensional or orthogonal factor structures (e.g., Zwick & Velicer, 1986).

The criterion’s inaccuracy in these cases is somewhat addressed if it is applied on the correlation matrix with communalities in the diagonal, either initial communalities estimated from SMCs (done setting eigen_type to "SMC") or final communality estimates from an EFA (done setting eigen_type to "EFA"; see Auerswald & Moshagen, 2019). However, although this variant of the KGC is more accurate in some cases compared to the traditional KGC, it is at the same time less accurate than the PCA-variant in other cases, and it is still often less accurate than other factor retention methods, for example parallel analysis (PARALLEL), the Hull method HULL, or sequential $\chi^2$ model tests (SMT; see Auerswald & Moshagen, 2019).

The KGC function can also be called together with other factor retention criteria in the N_FACTORS function.

Value

A list of class KGC containing

eigen_PCA A vector containing the eigenvalues found with PCA.

eigen_SMC A vector containing the eigenvalues found with SMCs.

eigen_EFA A vector containing the eigenvalues found with EFA.

The number of factors to retain according to the Kaiser- Guttmann criterion with PCA eigenvalues type.
n_fac_SMC  The number of factors to retain according to the Kaiser- Guttmann criterion with SMC eigenvalues type.

n_fac_EFA  The number of factors to retain according to the Kaiser- Guttmann criterion with EFA eigenvalues type.

settings  A list of the settings used.

Source


See Also

Other factor retention criteria: CD, EKC, HULL, PARALLEL, SMT

N_FACTORS as a wrapper function for this and all the above-mentioned factor retention criteria.

Examples

KGC(test_models$baseline$cormat, eigen_type = c("PCA", "SMC"))
Arguments

x data.frame or matrix. Dataframe or matrix of raw data or matrix with correlations.

use character. Passed to stats::cor if raw data is given as input. Default is "pairwise.complete.obs".

cor_method character. Passed to stats::cor. Default is "pearson".

Details

Kaiser (1970) proposed this index, originally called measure of sampling adequacy (MSA), that indicates how near the inverted correlation matrix $R^{-1}$ is to a diagonal matrix $S$ to determine a given correlation matrix’s ($R$) suitability for factor analysis. The index is

$$KMO = \frac{\sum_{i<j} \sum r_{ij}^2}{\sum_{i<j} \sum r_{ij}^2 + \sum_{i<j} \sum q_{ij}^2}$$

with $Q = SR^{-1}S$ and $S = (\text{diag} R^{-1})^{-1/2}$ where $\sum_{i<j} \sum r_{ij}^2$ is the sum of squares of the upper off-diagonal elements of $R$ and $\sum_{i<j} \sum q_{ij}^2$ is the sum of squares of the upper off-diagonal elements of $Q$ (see also Cureton & D’Augustino, 1983).

So KMO varies between 0 and 1, with larger values indicating higher suitability for factor analysis. Kaiser and Rice (1974) suggest that KMO should at least exceed .50 for a correlation matrix to be suitable for factor analysis.

This function was heavily influenced by the psych::KMO function.

See also BARTLETT for another test of suitability for factor analysis.

The KMO function can also be called together with the BARTLETT function and with factor retention criteria in the N_FACTORS function.

Value

A list containing

- KMO Overall KMO.
- KMO_i KMO for each variable.
- settings A list of the settings used.

Source


**See Also**

*BARTLETT* for another measure to determine suitability for factor analysis.

*N_FACTORS* as a wrapper function for this function, *BARTLETT* and several factor retention criteria.

**Examples**

```r
KMO(test_models$baseline$cormat)
```

---

### Description

Among the most important decisions for an exploratory factor analysis (EFA) is the choice of the number of factors to retain. Several factor retention criteria have been developed for this. With this function, various factor retention criteria can be performed simultaneously. Additionally, the data can be checked for their suitability for factor analysis.

### Usage

```r
N_FACTORS(
  x,
  criteria = c("CD", "EKC", "HULL", "KGC", "PARALLEL", "SCREE", "SMT"),
  suitability = TRUE,
  N = NA,
  use = c("pairwise.complete.obs", "all.obs", "complete.obs", "everything",
         "na.or.complete"),
  cor_method = c("pearson", "spearman", "kendall"),
  n_factors_max = NA,
  N_pop = 10000,
  N_samples = 500,
  alpha = 0.3,
  max_iter_CD = 50,
  n_fac_theor = NA,
  method = c("PAF", "ULS", "ML"),
  gof = c("CAF", "CFI", "RMSEA"),
  eigen_type_HULL = c("SMC", "PCA", "EFA"),
  eigen_type_other = c("PCA", "SMC", "EFA"),
  n_factors = 1,
  n_datasets = 1000,
  percent = 95,
  decision_rule = c("means", "percentile", "crawford"),
  show_progress = TRUE,
  ...
)
```
Arguments

- **x** data.frame or matrix. Dataframe or matrix of raw data or matrix with correlations. If "CD" is included as a criterion, x must be raw data.
- **criteria** character. A vector with the factor retention methods to perform. Possible inputs are: "CD", "EKC", "HULL", "KGC", "PARALLEL", "SCREE", and "SMT" (see details). By default, all factor retention methods are performed.
- **suitability** logical. Whether the data should be checked for suitability for factor analysis using the Bartlett’s test of sphericity and the Kaiser-Guttman criterion (see details). Default is TRUE.
- **N** numeric. The number of observations. Only needed if x is a correlation matrix.
- **use** character. Passed to stats::cor if raw data is given as input. Default is "pairwise.complete.obs".
- **cor_method** character. Passed to stats::cor Default is "pearson".
- **n_factors_max** numeric. Passed to CD. The maximum number of factors to test against. Larger numbers will increase the duration the procedure takes, but test more possible solutions. Maximum possible is number of variables / 2. Default is NA. If not specified, number of variables / 2 is used.
- **N_pop** numeric. Passed to CD. Size of finite populations of comparison data. Default is 10000.
- **N_samples** numeric. Passed to CD. Number of samples drawn from each population. Default is 500.
- **alpha** numeric. Passed to CD. The alpha level used to test the significance of the improvement added by an additional factor. Default is .30.
- **max_iter_CD** numeric. Passed to CD. The maximum number of iterations to perform after which the iterative PAF procedure is halted. Default is 50.
- **n_fac_theor** numeric. Passed to HULL. Theoretical number of factors to retain. The maximum of this number and the number of factors suggested by PARALLEL plus one will be used in the Hull method.
- **method** character. Passed to EFA in HULL, KGC, SCREE, and PARALLEL. The estimation method to use. One of "PAF", "ULS", or "ML", for principal axis factoring, unweighted least squares, and maximum likelihood, respectively.
- **gof** character. Passed to HULL. The goodness of fit index to use. Either "CAF", "CFI", or "RMSEA", or any combination of them. If method = "PAF" is used, only the CAF can be used as goodness of fit index. For details on the CAF, see Lorenzo-Seva, Timmerman, and Kiers (2011).
- **eigen_type_HULL** character. Passed to PARALLEL in HULL. On what the eigenvalues should be found in the parallel analysis. Can be one of "SMC", "PCA", or "EFA". If using "SMC" (default), the diagonal of the correlation matrices is replaced by the squared multiple correlations (SMCs) of the indicators. If using "PCA", the diagonal values of the correlation matrices are left to be 1. If using "EFA", eigenvalues are found on the correlation matrices with the final communalities of an EFA solution as diagonal.


\begin{itemize}
  \item \texttt{eigen\_type\_other} character. Passed to \texttt{KGC}, \texttt{SCREE}, and \texttt{PARALLEL}. The same as \texttt{eigen\_type\_HULL}, but multiple inputs are possible here. Default is to use all inputs, that is, c("PCA", "SMC", "EFA")
  \item \texttt{n\_factors} numeric. Passed to \texttt{PARALLEL} (also within \texttt{HULL}), \texttt{KGC}, and \texttt{SCREE}. Number of factors to extract if "EFA" is included in \texttt{eigen\_type\_HULL} or \texttt{eigen\_type\_other}. Default is 1.
  \item \texttt{n\_datasets} numeric. Passed to \texttt{PARALLEL} (also within \texttt{HULL}). The number of datasets to simulate. Default is 1000.
  \item \texttt{percent} numeric. Passed to \texttt{PARALLEL} (also within \texttt{HULL}). A vector of percentiles to take the simulated eigenvalues from. Default is 95.
  \item \texttt{decision\_rule} character. Passed to \texttt{PARALLEL} (also within \texttt{HULL}). Which rule to use to determine the number of factors to retain. Default is "means", which will use the average simulated eigenvalues. "percentile", uses the percentiles specified in percent. "crawford" uses the 95th percentile for the first factor and the mean afterwards (based on Crawford et al, 2010).
  \item \texttt{show\_progress} logical. Whether a progress bar should be shown in the console. Default is TRUE.
\end{itemize}

\textbf{Details}

By default, the entered data are checked for suitability for factor analysis using the following methods (see respective documentations for details):

- Bartlett’s test of sphericity (see \texttt{BARTLETT})
- Kaiser-Meyer-Olkin criterion (see \texttt{KMO})

The available factor retention criteria are the following (see respective documentations for details):

- Comparison data (see \texttt{CD})
- Empirical Kaiser criterion (see \texttt{EKC})
- Hull method (see \texttt{HULL})
- Kaiser-Guttman criterion (see \texttt{KGC})
- Parallel analysis (see \texttt{PARALLEL})
- Scree plot (see \texttt{SCREE})
- Sequential chi-square model tests, RMSEA lower bound, and AIC (see \texttt{SMT})

\textbf{Value}

A list of class \texttt{N\_FACTORS} containing

\begin{itemize}
  \item \texttt{outputs} A list with the outputs from \texttt{BARTLETT} and \texttt{KMO} and the factor retention criteria.
  \item \texttt{n\_factors} A named vector containing the suggested number of factors from each factor retention criterion.
  \item \texttt{settings} A list of the settings used.
\end{itemize}
Examples

# All criteria, with correlation matrix and fit method "ML" (where needed)
# This will throw a warning for CD, as no raw data were specified
nfac_all <- N_FACTORS(test_models$baseline$cormat, N = 500, method = "ML")

# The same as above, but without "CD"
ncf_wo_CD <- N_FACTORS(test_models$baseline$cormat, criteria = c("EKC", "HULL", "KGC", "PARALLEL", "SCREE", "SMT"), N = 500, method = "ML")

# Use PAF instead of ML (this will take a lot longer). For this, gof has
# to be set to "CAF" for the Hull method.
nfac_PAF <- N_FACTORS(test_models$baseline$cormat, criteria = c("EKC", "HULL", "KGC", "PARALLEL", "SCREE", "SMT"), N = 500, gof = "CAF")

# Do KGC and PARALLEL with only "PCA" type of eigenvalues
nfac_PCA <- N_FACTORS(test_models$baseline$cormat, criteria = c("EKC", "HULL", "KGC", "PARALLEL", "SCREE", "SMT"), N = 500, method = "ML", eigen_type_other = "PCA")

# Use raw data, such that CD can also be performed
nfac_raw <- N_FACTORS(GRiPS_raw, method = "ML")

---

**OMEGA**

**McDonald’s omega**

Description

This function finds omega total, omega hierarchical, and omega subscale from a Schmid-Leiman (SL) solution or lavaan single factor, second-order (see below), or bifactor solution. The SL-based omegas can either be found from a `psych::schmid`, SL, or, in a more flexible way, by leaving `model = NULL` and specifying additional arguments. By setting the type argument, results from `psych::omega` can be reproduced.

Usage

```r
OMEGA(
  model = NULL,
  type = c("EFAtools", "psych"),
  g_name = "g",
  group_names = NULL,
  factor_corres = NULL,
  var_names = NULL,
  fac_names = NULL,
  g_load = NULL,
)```
s_load = NULL,
u2 = NULL,
cormat = NULL,
pattern = NULL,
Phi = NULL,
variance = c("correlation", "sums_load")
)

Arguments

model class SL, class schmid, or class lavaan object. That is, an output object from
SL or psych::schmid, or a lavaan fit object with a single factor, second-order,
or bifactor solution. If of class lavaan, only g_name needs to be specified ad-
ditionally. If of class SL or schmid, only the arguments factor_corres and
cormat need to be specified additionally.

type character. Either "EFAtools" (default) or "psych" (see details)

g_name character. The name of the general factor from the lavaan solution. This needs
only be specified if model is a lavaan second-order or bifactor solution. Default
is "g".

group_names character. An optional vector of group names. The length must correspond to
the number of groups for which the lavaan model was fitted.

factor_corres matrix. A logical matrix or a numeric matrix containing 0's and 1's that indi-
cates which variable corresponds to which group factor. Must have the same
dimensions as the matrix of group factor loadings from the SL solution. Cross-
loadings are allowed here. See examples for use.

var_names character. A vector with subtest names in the order of the rows from the SL
solution. This needs only be specified if model is left NULL.

fac_names character. An optional vector of group factor names in the order of the columns
of the SL solution. If left NULL, names of the group factors from the entered
solution are taken.

g_load numeric. A vector of general factor loadings from an SL solution. This needs
only be specified if model is left NULL.

s_load matrix. A matrix of group factor loadings from an SL solution. This needs only
be specified if model is left NULL.

u2 numeric. A vector of uniquenesses from an SL solution. This needs only be
specified if model is left NULL.

cormat matrix. A correlation matrix to be used when variance = "correlation". If
left NULL and an SL output is entered in model, the correlation matrix is taken
from the output. If left NULL and a psych::schmid output is entered, the cor-
relation matrix will be found based on the pattern matrix and Phi from the
psych::schmid output using psych::factor.model. If left NULL and model is
also left NULL, the correlation matrix is found based on the pattern matrix and
Phi entered. However, if the correlation matrix is available, cormat should be
specified instead of Phi and pattern.
pattern matrix. Pattern coefficients from an oblique factor solution. This needs only be specified if `model` is left NULL, `variance` = "correlation" and `cormat` is also left NULL.

Phi matrix. Factor intercorrelations from an oblique factor solution. This needs only be specified if `model` is left NULL, `variance` = "correlation" and `cormat` is also left NULL.

variance character. If "correlation" (default), then total variances for the whole scale as well as for the subscale composites are calculated based on the correlation matrix. If "sums_load", then total variances are calculated using the squared sums of general factor loadings and group factor loadings and the sum of uniquenesses (see details).

Details

If `model` is a lavaan second-order or bifactor solution, only the name of the general factor from the lavaan model needs to be specified additionally with the `g_name` argument. It is then determined whether this general factor is a second-order factor (second-order model assumed) or a breadth factor (bifactor model assumed). In case of a second-order solution, a Schmid-Leiman transformation is performed on the first- and second-order loadings and omega coefficients are obtained from the transformed (orthogonalized) solution (see SL for more information on Schmid-Leiman transformation). There is also the possibility to enter a lavaan single factor solution. In this case, `g_name` is not needed. Finally, if a solution from a lavaan multiple group analysis is entered, the omegas are computed for each group. The type argument is not evaluated if `model` is of class lavaan.

If `model` is of class SL or psych::schmid only the type and, depending on the type (see below), the factor_corres arguments need to be specified additionally. If `model` is of class psych::schmid and `variance` = "correlation" (default), it is recommended to also provide the original correlation matrix in `cormat` to get more accurate results. Otherwise, the correlation matrix will be found based on the pattern matrix and Phi from the psych::schmid output using the psych::factor.model function.

If `model` = NULL, the arguments type, factor_corres (depending on the type, see below), var_names, g_load, s_load, and u2 and either `cormat` (recommended) or Phi and pattern need to be specified. If Phi and pattern are specified instead of `cormat`, the correlation matrix is found using the psych::factor.model function.

The only difference between `type` = "EFAtools" and `type` = "psych" is the determination of variable-to-factor correspondences. `type` = "psych" reproduces the psych::omega results, where variable-to-factor correspondences are found by taking the highest group factor loading for each variable as the relevant group factor loading. To do this, factor_corres must be left NULL.

The calculation of the total variance (for the whole scale as well as the subscale composites) can also be controlled in this function using the variance argument. For both types—"EFAtools" and "psych" —variance is set to "correlation" by default, which means that total variances are found using the correlation matrix. If `variance` = "sums_load" the total variance is calculated using the squared sums of general loadings and group factor loadings and the sum of uniquenesses. This will only get comparable results to `variance` = "correlation" if no cross-loadings are present and simple structure is well-achieved in general with the SL solution (i.e., the uniquenesses should capture almost all of the variance not explained by the general factor and the variable’s allocated group factor).
Value

If found for an SL or lavaan second-order of bifactor solution without multiple groups: A matrix with omegas for the whole scale and for the subscales.

tot  Omega total.
hier Omega hierarchical.
sub  Omega subscale.

If found for a lavaan single factor solution without multiple groups: A vector with omega total for the single factor.

If found for a lavaan output from a multiple group analysis: A list containing the output described above for each group.

Source


Examples

```r
## Use with lavaan outputs

# Create and fit bifactor model in lavaan (assume all variables have SDs of 1)
mod <- 'F1 =~ V1 + V2 + V3 + V4 + V5 + V6
F2 =~ V7 + V8 + V9 + V10 + V11 + V12
F3 =~ V13 + V14 + V15 + V16 + V17 + V18
g =~ V1 + V2 + V3 + V4 + V5 + V6 + V7 + V8 + V9 + V10 + V11 + V12 + V13 + V14 + V15 + V16 + V17 + V18'
fit_bi <- lavaan::cfa(mod, sample.cov = test_models$baseline$cormat,
                      sample.nobs = 500, estimator = "ml", orthogonal = TRUE)

# Compute omega for bifactor solution
OMEGA(fit_bi, g_name = "g")

# Create and fit second-order model in lavaan (assume all variables have SDs of 1)
mod <- 'F1 =~ V1 + V2 + V3 + V4 + V5 + V6
F2 =~ V7 + V8 + V9 + V10 + V11 + V12
F3 =~ V13 + V14 + V15 + V16 + V17 + V18
g =~ F1 + F2 + F3'
fit_ho <- lavaan::cfa(mod, sample.cov = test_models$baseline$cormat,
                      sample.nobs = 500, estimator = "ml")

# Compute omega for second-order solution
```
## Use with an output from the SL function, with type EFAtools

```r
efa_mod <- EFA(test_models$baseline$cormat, N = 500, n_factors = 3,
    type = "EFAtools", method = "PAF", rotation = "promax")
sl_mod <- SL(efa_mod, type = "EFAtools", method = "PAF")
```

# Two examples how to specify the indicator-to-factor correspondences:

# Based on a specific salience threshold for the loadings (here: .20):

```r
factor_corres_1 <- sl_mod$sl[, c("F1", "F2", "F3")] >= .2
```

# Or more flexibly (could also be TRUE and FALSE instead of 0 and 1):

```r
factor_corres_2 <- matrix(c(rep(0, 12), rep(1, 6), rep(0, 6), rep(1, 6),
    rep(0, 6), rep(1, 6), rep(0, 12)), ncol = 3,
    byrow = FALSE)
```

```r
OMEGA(sl_mod, type = "EFAtools", factor_corres = factor_corres_1)
```

## Use with an output from the psych::schmid function, with type psych for
## OMEGA

```r
schmid_mod <- psych::schmid(test_models$baseline$cormat, nfactors = 3,
    n.obs = 500, fm = "pa", rotate = "Promax")
```

# Find correlation matrix from phi and pattern matrix from psych::schmid output

```r
OMEGA(schmid_mod, type = "psych")
```

# Use specified correlation matrix

```r
OMEGA(schmid_mod, type = "psych", cormat = test_models$baseline$cormat)
```

## Manually specify components (useful if omegas should be computed for a SL
## or bifactor solution found with another program)
## As an example, we extract the elements from an SL output here. This gives
## the same results as in the second example above.

```r
efa_mod <- EFA(test_models$baseline$cormat, N = 500, n_factors = 3,
    type = "EFAtools", method = "PAF", rotation = "promax")
sl_mod <- SL(efa_mod, type = "EFAtools", method = "PAF")
```

```r
factor_corres <- matrix(c(rep(0, 12), rep(1, 6), rep(0, 6), rep(1, 6),
    rep(0, 6), rep(1, 6), rep(0, 12)), ncol = 3,
    byrow = FALSE)
```

```r
OMEGA(model = NULL, type = "EFAtools", var_names = rownames(sl_mod$sl),
    g_load = sl_mod$sl[, "g"], s_load = sl_mod$sl[, c("F1", "F2", "F3")],
    u2 = sl_mod$sl[, "u2"], cormat = test_models$baseline$cormat,
    factor_corres = factor_corres)
```
Description

Various methods for performing parallel analysis. This function uses `future::lapply` for which a parallel processing plan can be selected. To do so, call `library(future)` and, for example, `plan(multisession)`; see examples.

Usage

```r
PARALLEL(
  x = NULL,
  N = NA,
  n_vars = NA,
  n_datasets = 1000,
  percent = 95,
  eigen_type = c("PCA", "SMC", "EFA"),
  use = c("pairwise.complete.obs", "all.obs", "complete.obs", "everything",
          "na.or.complete"),
  cor_method = c("pearson", "spearman", "kendall"),
  decision_rule = c("means", "percentile", "crawford"),
  n_factors = 1,
  ...
)
```

Arguments

- **x**: matrix or data.frame. The real data to compare the simulated eigenvalues against. Must not contain variables of classes other than numeric. Can be a correlation matrix or raw data.
- **N**: numeric. The number of cases / observations to simulate. Only has to be specified if `x` is either a correlation matrix or `NULL`. If `x` contains raw data, `N` is found from the dimensions of `x`.
- **n_vars**: numeric. The number of variables / indicators to simulate. Only has to be specified if `x` is left as `NULL` as otherwise the dimensions are taken from `x`.
- **n_datasets**: numeric. The number of datasets to simulate. Default is 1000.
- **percent**: numeric. The percentile to take from the simulated eigenvalues. Default is 95.
- **eigen_type**: character. On what the eigenvalues should be found. Can be either "SMC", "PCA", or "EFA". If using "SMC", the diagonal of the correlation matrix is replaced by the squared multiple correlations (SMCs) of the indicators. If using "PCA", the diagonal values of the correlation matrices are left to be 1. If using "EFA", eigenvalues are found on the correlation matrices with the final communalities of an EFA solution as diagonal.
- **use**: character. Passed to `stats::cor` if raw data is given as input. Default is "pairwise.complete.obs".
- **cor_method**: character. Passed to `stats::cor`. Default is "pearson".
- **decision_rule**: character. Which rule to use to determine the number of factors to retain. Default is "means", which will use the average simulated eigenvalues. "percentile", uses the percentiles specified in percent. "crawford" uses the 95th percentile for the first factor and the mean afterwards (based on Crawford et al, 2010).
n_factors numeric. Number of factors to extract if "EFA" is included in eigen_type. Default is 1.

Additional arguments passed to EFA. For example, the extraction method can be changed here (default is "PAF"). PAF is more robust, but it will take longer compared to the other estimation methods available ("ML" and "ULS").

Details

Parallel analysis (Horn, 1965) compares the eigenvalues obtained from the sample correlation matrix against those of null model correlation matrices (i.e., with uncorrelated variables) of the same sample size. This way, it accounts for the variation in eigenvalues introduced by sampling error and thus eliminates the main problem inherent in the Kaiser-Guttman criterion (KGC).

Three different ways of finding the eigenvalues under the factor model are implemented, namely "SMC", "PCA", and "EFA". PCA leaves the diagonal elements of the correlation matrix as they are and is thus equivalent to what is done in PCA. SMC uses squared multiple correlations as communality estimates with which the diagonal of the correlation matrix is replaced. Finally, EFA performs an EFA with one factor (can be adapted to more factors) to estimate the communalities and based on the correlation matrix with these as diagonal elements, finds the eigenvalues.

Parallel analysis is often argued to be one of the most accurate factor retention criteria. However, for highly correlated factor structures it has been shown to underestimate the correct number of factors. The reason for this is that a null model (uncorrelated variables) is used as reference. However, when factors are highly correlated, the first eigenvalue will be much larger compared to the following ones, as later eigenvalues are conditional on the earlier ones in the sequence and thus the shared variance is already accounted in the first eigenvalue (e.g., Braeken & van Assen, 2017).

The PARALLEL function can also be called together with other factor retention criteria in the N_FACTORS function.

Value

A list of class PARALLEL containing the following objects

eigenvalues_PCA A matrix containing the eigenvalues of the real and the simulated data found with eigen_type = "PCA"
eigenvalues_SMC A matrix containing the eigenvalues of the real and the simulated data found with eigen_type = "SMC"
eigenvalues_EFA A matrix containing the eigenvalues of the real and the simulated data found with eigen_type = "EFA"
n_fac_PCA The number of factors to retain according to the parallel procedure with eigen_type = "PCA".
n_fac_SMC The number of factors to retain according to the parallel procedure with eigen_type = "SMC".
n_fac_EFA The number of factors to retain according to the parallel procedure with eigen_type = "EFA".
settings A list of control settings used in the print function.
plot.CD

Source

See Also
Other factor retention criteria: CD, EKC, HULL, KGC, SMT
N_FACTORS as a wrapper function for this and all the above-mentioned factor retention criteria.

Examples

# example without real data
pa_unreal <- PARALLEL(N = 500, n_vars = 10)

# example with correlation matrix with all eigen_types and PAF estimation
pa_paf <- PARALLEL(test_models$case_11b$cormat, N = 500)

# example with correlation matrix with all eigen_types and ML estimation
# this will be faster than the above with PAF)
pa_ml <- PARALLEL(test_models$case_11b$cormat, N = 500, method = "ML")

## Not run:
# for parallel computation
future::plan(future::multisession)
pa_faster <- PARALLEL(test_models$case_11b$cormat, N = 500)

## End(Not run)

---

plot.CD

Plot CD object

Description
Plot method showing a summarized output of the CD function

Usage

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

---
plot.EFA_AVERAGE

Arguments

x  a list of class CD. An output from the CD function.

...  not used.

Description

Plot method showing a summarized output of the EFA_AVERAGE function

Usage

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

Arguments

x  list. An output from the EFA_AVERAGE function.

...  not used.

Examples

EFA_aver <- EFA_AVERAGE(test_models$baseline$cormat, n_factors = 3, N = 500)
EFA_aver

plot.EKC

Plot EKC object

Description

Plot method showing a summarized output of the EKC function

Usage

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

Arguments

x  a list of class EKC. An output from the EKC function.

...  not used.
Examples

EKC_base <- EKC(test_models$baseline$cormat, N = 500)
plot(EKC_base)

plot.HULL  
Plot HULL object

Description
Plot method showing a summarized output of the HULL function

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

Arguments
x  
list of class HULL. An output from the HULL function.

...  
not used.

Examples

x <- HULL(test_models$baseline$cormat, N = 500, method = "ML")
plot(x)

plot.KGC  
Plot KGC object

Description
Plot method showing a summarized output of the KGC function

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

Arguments
x  
a list of class KGC. An output from the KGC function.

...  
not used.
Examples

```r
KGC_base <- KGC(test_models$baseline$cormat)
plot(KGC_base)
```

**Description**

Plot method showing a summarized output of the `PARALLEL` function

**Usage**

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

**Arguments**

- `x` list of class `PARALLEL`. An output from the `PARALLEL` function.
- `...` not used.

**Examples**

```r
# example with correlation matrix and "ML" estimation
x <- PARALLEL(test_models$case_11b$cormat, N = 500, method = "ML")
plot(x)
```

**plot.SCREE**

*Plot SCREE object*

**Description**

Plot method showing a summarized output of the `SCREE` function

**Usage**

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

**Arguments**

- `x` a list of class `SCREE` An output from the `SCREE` function.
- `...` not used.
Examples

```r
SCREE_base <- SCREE(test_models$baseline$cormat)
plot(SCREE_base)
```

Description

Population factor models, some of which (baseline to case_11e) used for the simulation analyses reported in Grieder and Steiner (2019). All combinations of the pattern matrices and the factor intercorrelations were used in the simulations. Many models are based on cases used in de Winter and Dodou (2012).

Usage

`population_models`

Format

A list of 3 lists "loadings", "phis_3", and "phis_6".

- **loadings** contains the following matrices of pattern coefficients:
  - **baseline** (matrix) - The pattern coefficients of the baseline model. Three factors with six indicators each, all with pattern coefficients of .6. Same baseline model as used in de Winter and Dodou (2012).
  - **case_1a** (matrix) - Three factors with 2 indicators per factor.
  - **case_1b** (matrix) - Three factors with 3 indicators per factor. Case 5 in de Winter and Dodou (2012).
  - **case_1c** (matrix) - Three factors with 4 indicators per factor.
  - **case_1d** (matrix) - Three factors with 5 indicators per factor.
  - **case_2** (matrix) - Same as baseline model but with low pattern coefficients of .3.
  - **case_3** (matrix) - Same as baseline model but with high pattern coefficients of .9.
  - **case_4** (matrix) - Three factors with different pattern coefficients *between* factors (one factor with .9, one with .6, and one with .3, respectively). Case 7 in de Winter and Dodou (2012).
  - **case_5** (matrix) - Three factors with different pattern coefficients *within* factors (each factor has two pattern coefficients of each .9, .6, and .3). Similar to cases 8/ 9 in de Winter and Dodou (2012).
  - **case_6a** (matrix) - Same as baseline model but with one cross loading of .4. Similar to case 10 in de Winter and Dodou (2012).
  - **case_6b** (matrix) - Same as baseline model but with three cross loading of .4 (One factor with 2 and one with 1 crossloading). Similar to case 10 in de Winter and Dodou (2012).
population_models

**case_7** (matrix) - Three factors with different number of indicators per factor (2, 4, and 6 respectively). Similar to cases 11/12 in de Winter and Dodou (2012).

**case_8** (matrix) - Three factors with random variation in pattern coefficients added, drawn from a uniform distribution between [-.2, .2]. Case 13 in de Winter and Dodou (2012).

**case_9a** (matrix) - Three factors with 2 indicators per factor, with different pattern coefficients within one of the factors.

**case_9b** (matrix) - Three factors with 3 indicators per factor, with different pattern coefficients.

**case_9c** (matrix) - Three factors with 4 indicators per factor, with different pattern coefficients.

**case_9d** (matrix) - Three factors with 5 indicators per factor, with different pattern coefficients.

**case_10a** (matrix) - Six factors with 2 indicators per factor, all with pattern coefficients of .6.

**case_10b** (matrix) - Six factors with 3 indicators per factor, all with pattern coefficients of .6.

**case_10c** (matrix) - Six factors with 4 indicators per factor, all with pattern coefficients of .6.

**case_10d** (matrix) - Six factors with 5 indicators per factor, all with pattern coefficients of .6.

**case_10e** (matrix) - Six factors with 6 indicators per factor, all with pattern coefficients of .6.

**case_11a** (matrix) - Six factors with 2 indicators per factor, with different pattern coefficients within and between factors (.3, .6, and .9).

**case_11b** (matrix) - Six factors with 3 indicators per factor, with different pattern coefficients within and between factors (.3, .6, and .9).

**case_11c** (matrix) - Six factors with 4 indicators per factor, with different pattern coefficients within and between factors (.3, .6, and .9).

**case_11d** (matrix) - Six factors with 5 indicators per factor, with different pattern coefficients within and between factors (.3, .6, and .9).

**case_11e** (matrix) - Six factors with 6 indicators per factor, with different pattern coefficients within and between factors (.3, .6, and .9).

**case_12a** (matrix) - One factor, with 2 equal pattern coefficients (.6).

**case_12b** (matrix) - One factor, with 3 equal pattern coefficients (.6).

**case_12c** (matrix) - One factor, with 6 equal pattern coefficients (.6).

**case_12d** (matrix) - One factor, with 10 equal pattern coefficients (.6).

**case_12e** (matrix) - One factor, with 15 equal pattern coefficients (.6).

**case_13a** (matrix) - One factor, with 2 different pattern coefficients (.3, .6).

**case_13b** (matrix) - One factor, with 3 different pattern coefficients (.3, .6, and .9).

**case_13c** (matrix) - One factor, with 6 different pattern coefficients (.3, .6, and .9).

**case_13d** (matrix) - One factor, with 10 different pattern coefficients (.3, .6, and .9).

**case_13e** (matrix) - One factor, with 15 different pattern coefficients (.3, .6, and .9).

**case_14a** (matrix) - No factor, 2 variables (0).

**case_14b** (matrix) - No factor, 3 variables (0).

**case_14c** (matrix) - No factor, 6 variables (0).

**case_14d** (matrix) - No factor, 10 variables (0).

**case_14e** (matrix) - No factor, 15 variables (0). \( \text{ph1s}_3 \) contains the following 3x3 matrices:
zero (matrix) - Matrix of factor intercorrelations of 0. Same intercorrelations as used in de Winter and Dodou (2012).

moderate (matrix) - Matrix of moderate factor intercorrelations of .3.

mixed (matrix) - Matrix of mixed (.3, .5, and .7) factor intercorrelations.

strong (matrix) - Matrix of strong factor intercorrelations of .7. Same intercorrelations as used in de Winter and Dodou (2012). phi_s_6 contains the following 6x6 matrices:

zero (matrix) - Matrix of factor intercorrelations of 0. Same intercorrelations as used in de Winter and Dodou (2012).

moderate (matrix) - Matrix of moderate factor intercorrelations of .3.

mixed (matrix) - Matrix of mixed (around .3, .5, and .7; smoothing was necessary for the matrix to be positive definite) factor intercorrelations.

strong (matrix) - Matrix of strong factor intercorrelations of .7. Same intercorrelations as used in de Winter and Dodou (2012).

Source


---

**print.BARTLETT**

*Print BARTLETT object*

**Description**

Print BARTLETT object

**Usage**

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

**Arguments**

- `x` list of class BARTLET (output from the `BARTLETT` function)
- `...` additional arguments passed to print

**Examples**

`BARTLETT(test_models$baseline$cormat, N = 500)`
print.CD

Print function for CD objects

Description
Print function for CD objects

Usage
## S3 method for class 'CD'
print(x, plot = TRUE, ...)

Arguments
x        a list of class CD. Output from CD function.
plot     logical. Whether to plot the results.
...      Further arguments for print.

Examples

# determine n factors of the GRiPS
CD(GRiPS_raw)

print.COMPARE

Print COMPARE object

Description
Print Method showing a summarized output of the COMPARE function.

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

Arguments
x        list. An object of class COMPARE to be printed
...      Further arguments for print.
Examples

# A type SPSS EFA to mimick the SPSS implementation
EFA_SPSS_5 <- EFA(IDS2_R, n_factors = 5, type = "SPSS")

# A type psych EFA to mimick the psych::fa() implementation
EFA_psych_5 <- EFA(IDS2_R, n_factors = 5, type = "psych")

# compare the two
COMPARE(EFA_SPSS_5$unrot_loadings, EFA_psych_5$unrot_loadings,
       x_labels = c("SPSS", "psych"))

print.EFA  

Print EFA object

Description

Print Method showing a summarized output of the EFA function

Usage

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

Arguments

x  
    list. An object of class EFA to be printed

...  
    Further arguments for print.

Examples

EFAtools_PAF <- EFA(test_models$baseline$cormat, n_factors = 3, N = 500,
                     type = "EFAtools", method = "PAF", rotation = "promax")
EFAtools_PAF

print.EFA_AVERAGE  

Print EFA_AVERAGE object

Description

Print Method showing a summarized output of the EFA_AVERAGE function

Usage

## S3 method for class 'EFA_AVERAGE'
print(x, stat = c("average", "range"), plot = TRUE, ...)

Arguments

x  
    list. An object of class EFA_AVERAGE to be printed

stat  
    a character vector specifying the output to be printed.

plot  
    If TRUE, a plot of the screeplot and loadings is printed.

Examples

EFAtools_PAF <- EFA(test_models$baseline$cormat, n_factors = 3, N = 500,
                     type = "EFAtools", method = "PAF", rotation = "promax")
EFAtools_PAF
Arguments

- **x**: list. An object of class EFA_AVERAGE to be printed
- **stat**: character. A vector with the statistics to print. Possible inputs are "average", "sd", "range", "min", and "max". Default is "average" and "range".
- **plot**: logical. Whether a plot of the average and min-max loadings should be created. Default is TRUE. If more than 10 factors are extracted, no plot is created.
- **...**: Further arguments for print.

Examples

EFA_aver <- EFA_AVERAGE(test_models$baseline$cormat, n_factors = 3, N = 500)
EFA_aver

---

### print.EKC

Print function for EKC objects

Description

Print function for EKC objects

Usage

```r
## S3 method for class 'EKC'
print(x, plot = TRUE, ...)
```

Arguments

- **x**: a list of class EKC. Output from EKC function.
- **plot**: logical. Whether to plot the results.
- **...**: Further arguments for print.

Examples

EKC_base <- EKC(test_models$baseline$cormat, N = 500)
EKC_base
print.HULL

Print function for HULL objects

Description
Print function for HULL objects

Usage
## S3 method for class 'HULL'
print(x, plot = TRUE, ...)

Arguments

x
a list of class HULL. Output from the HULL function.

plot
logical. Whether to plot the results.

...
Further arguments for print.

Examples

HULL(test_models$baseline$cormat, N = 500, method = "ML")

print.KGC

Print function for KGC objects

Description
Print function for KGC objects

Usage
## S3 method for class 'KGC'
print(x, plot = TRUE, ...)

Arguments

x
a list of class KGC. Output from KGC function.

plot
logical. Whether to plot the results.

...
Further arguments for print.

Examples

KGC_base <- KGC(test_models$baseline$cormat)
KGC_base
### print.KMO

**Print KMO object**

#### Description
Print KMO object

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

#### Arguments
- `x`: list of class KMO (output from the `KMO` function)
- `...`: additional arguments passed to `print`

#### Examples
```r
KMO_base <- KMO(test_models$baseline$cormat)
KMO_base
```

### print.LOADINGS

**Print LOADINGS object**

#### Description
Print LOADINGS object

#### Usage
```r
## S3 method for class 'LOADINGS'
print(x, cutoff = 0.3, digits = 3, ...)  
```

#### Arguments
- `x`: class LOADINGS matrix.
- `cutoff`: numeric. The number above which to print loadings in bold default is .3.
- `digits`: numeric. Passed to `round`. Number of digits to round the loadings to (default is 3).
- `...`: additional arguments passed to `print`
Examples

EFAtools_PAF <- EFA(test_models$baseline$cormat, n_factors = 3, N = 500,
               type = "EFAtools", method = "PAF", rotation = "promax")
EFAtools_PAF

print.N_FACTORS

print function for N_FACTORS objects

Usage

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

Arguments

x
a list of class N_FACTORS. Output from N_FACTORS function.

... Further arguments for print.

Examples

# All criteria except "CD", with correlation matrix and fit method "ML"
# (where needed)
N_FACTORS(test_models$baseline$cormat, criteria = c("EKC", "HULL", "KGC",
               "PARALLEL", "SCREE", "SMT"), N = 500, method = "ML")

print.OMEGA

Print OMEGA object

Description

Print OMEGA object

Usage

## S3 method for class 'OMEGA'
print(x, digits = 3, ...)

Arguments

- `x` output of class OMEGA (output from the OMEGA function)
- `digits` numeric. Passed to `round`. Number of digits to round to (default is 3).
- `...` additional arguments passed to print

Examples

```r
efa_mod <- EFA(test_models$baseline$cormat, N = 500, n_factors = 3,
               type = "EFAtools", method = "PAF", rotation = "promax")
sl_mod <- SL(efa_mod, type = "EFAtools", method = "PAF")
OMEGA(sl_mod, type = "EFAtools",
      factor_corres = sl_mod$sl[, c("F1", "F2", "F3") >= .2]
```

Description

Print function for PARALLEL objects

Usage

```r
## S3 method for class 'PARALLEL'
print(x, plot = TRUE, ...)
```

Arguments

- `x` a list of class PARALLEL. Output from PARALLEL function.
- `plot` logical. Whether to plot the results.
- `...` Further arguments for print.

Examples

```r
# example without real data
PARALLEL(N = 500, n_vars = 10)

# example with correlation matrix and "ML" estimation
PARALLEL(test_models$case_11b$cormat, N = 500, method = "ML")
```
**print.SCREE**  
*Print function for SCREE objects*

**Description**
Print function for SCREE objects

**Usage**

```r
## S3 method for class 'SCREE'
print(x, plot = TRUE, ...)
```

**Arguments**

- `x`  
  a list of class SCREE Output from `SCREE` function.
- `plot`  
  logical. Whether to plot the results.
- `...`  
  Further arguments for print.

**Examples**

```r
SCREE_base <- SCREE(test_models$baseline$cormat)
SCREE_base
```

---

**print.SL**  
*Print SL object*

**Description**
Print Method showing a summarized output of the SL function.

**Usage**

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

**Arguments**

- `x`  
  list. An object of class SL to be printed
- `...`  
  Further arguments for print.

**Examples**

```r
EFA_mod <- EFA(test_models$baseline$cormat, N = 500, n_factors = 3, 
               type = "EFAtools", method = "PAF", rotation = "promax")
SL(EFA_mod, type = "EFAtools", method = "PAF")
```
print.SLOADINGS

**Print SLLOADINGS object**

**Description**

Print SLLOADINGS object

**Usage**

```r
## S3 method for class 'SLLOADINGS'
print(x, cutoff = 0.2, digits = 3, ...)
```

**Arguments**

- `x` : class SLLOADINGS matrix.
- `cutoff` : numeric. The number above which to print loadings in bold (default is .2).
- `digits` : numeric. Passed to `round`. Number of digits to round the loadings to (default is 3).
- `...` : additional arguments passed to `print`

**Examples**

```r
EFA_mod <- EFA(test_models$baseline$cormat, N = 500, n_factors = 3,
               type = "EFAtools", method = "PAF", rotation = "promax")
SL(EFA_mod, type = "EFAtools", method = "PAF")
```

print.SMT

**Print SMT object**

**Description**

Print SMT object

**Usage**

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

**Arguments**

- `x` : list of class SMT (output from the SMT function)
- `...` : additional arguments passed to print
Examples

```r
SMT_base <- SMT(test_models$baseline$cormat, N = 500)
SMT_base
```

Description

A list containing the bivariate correlations (cormat) of the 9 dimensions on which participants in Fischhoff et al. (1978) rated different activities and technologies as well as the sample size (N). This was then analyzed together with ratings of the risks and benefits of these activities and technologies.

Usage

RiskDimensions

Format

An object of class list of length 2.

Source


SCREE

Description

The scree plot was originally introduced by Cattell (1966) to perform the scree test. In a scree plot, the eigenvalues of the factors / components are plotted against the index of the factors / components, ordered from 1 to N factors components, hence from largest to smallest eigenvalue. According to the scree test, the number of factors / components to retain is the number of factors / components to the left of the "elbow" (where the curve starts to level off) in the scree plot.
Usage

```r
SCREE(
  x,
  eigen_type = c("PCA", "SMC", "EFA"),
  use = c("pairwise.complete.obs", "all.obs", "complete.obs", "everything",
          "na.or.complete"),
  cor_method = c("pearson", "spearman", "kendall"),
  n_factors = 1,
  ...)
```

Arguments

- **x**: data.frame or matrix. Dataframe or matrix of raw data or matrix with correlations.
- **eigen_type**: character. On what the eigenvalues should be found. Can be either "PCA", "SMC", or "EFA", or some combination of them. If using "PCA", the diagonal values of the correlation matrices are left to be 1. If using "SMC", the diagonal of the correlation matrices is replaced by the squared multiple correlations (SMCs) of the indicators. If using "EFA", eigenvalues are found on the correlation matrices with the final communalities of an exploratory factor analysis solution (default is principal axis factoring extracting 1 factor) as diagonal.
- **use**: character. Passed to `stats::cor` if raw data is given as input. Default is "pairwise.complete.obs".
- **cor_method**: character. Passed to `stats::cor`. Default is "pearson".
- **n_factors**: numeric. Number of factors to extract if "EFA" is included in `eigen_type`. Default is 1.
- **...**: Additional arguments passed to `EFA`. For example, to change the extraction method (PAF is default).

Details

As the scree test requires visual examination, the test has been especially criticized for its subjectivity and with this low inter-rater reliability. Moreover, a scree plot can be ambiguous if there are either no clear "elbow" or multiple "elbows", making it difficult to judge just where the eigenvalues do level off. Finally, the scree test has also been found to be less accurate than other factor retention criteria. For all these reasons, the scree test has been recommended against, at least for exclusive use as a factor retention criterion (Zwick & Velicer, 1986)

The SCREE function can also be called together with other factor retention criteria in the `N_FACTORS` function.

Value

A list of class SCREE containing

- **eigen_PCA**: A vector containing the eigenvalues found with PCA.
- **eigen_SMC**: A vector containing the eigenvalues found with SMCs.
eigen_EFA  A vector containing the eigenvalues found with EFA.
settings  A list of the settings used.

Source

See Also
Other factor retention criteria: CD, EKC, NULL, PARALLEL, SMT
N_FACTORS as a wrapper function for this and all the above-mentioned factor retention criteria.

Examples
SCREE(test_models$baseline$cormat, eigen_type = c("PCA", "SMC"))

---

**SL**

*Schmid-Leiman Transformation*

Description
This function implements the Schmid-Leiman (SL) transformation (Schmid & Leiman, 1957). It takes the pattern coefficients and factor intercorrelations from an oblique factor solution as input and can reproduce the results from psych::schmid and from the SPSS implementation from Wolff & Preising (2005). Other arguments from EFA can be used to control the procedure to find the second-order loadings more flexibly. The function can also be used on a second-order confirmatory factor analysis (CFA) solution from lavaan.

Usage
SL(
  x,
  Phi = NULL,
  type = c("EFAtools", "psych", "SPSS", "none"),
  method = c("PAF", "ML", "ULS"),
  g_name = "g",
  ...)


Arguments

x object of class EFA, class psych::fa, class lavaan or matrix. If class EFA or class psych::fa, pattern coefficients and factor intercorrelations are taken from this object. If class lavaan, it must be a second-order CFA solution. In this case first-order and second-order factor loadings are taken from this object and the g_name argument has to be specified. x can also be a pattern matrix from an oblique factor solution (see Phi) or a matrix of first-order factor loadings from a higher-order confirmatory factor analysis (see L2).

Phi matrix. A matrix of factor intercorrelations from an oblique factor solution. Only needs to be specified if a pattern matrix is entered directly into x.

type character. One of "EFAtools" (default), "psych", "SPSS", or "none". This is used to control the procedure of the second-order factor analysis. See EFA for details.

method character. One of "PAF", "ML", or "ULS" to use principal axis factoring, maximum likelihood, or unweighted least squares (also called minres), respectively, used in EFA to find the second-order loadings.

g_name character. The name of the general factor. This needs only be specified if x is a lavaan second-order solution. Default is "g".

... Arguments to be passed to EFA.

Details

The SL transformation (also called SL orthogonalization) is a procedure with which an oblique factor solution is transformed into a hierarchical, orthogonalized solution. As a first step, the factor intercorrelations are again factor analyzed to find second-order factor loadings. If there is only one higher-order factor, this step of the procedure stops there, resulting in a second-order factor structure. The first-order factor and the second-order factor are then orthogonalized, resulting in an orthogonalized factor solution with proportionality constraints. The procedure thus makes a suggested hierarchical data structure based on factor intercorrelations explicit. One major advantage of SL transformation is that it enables variance partitioning between higher-order and first-order factors, including the calculation of McDonald’s omegas (see OMEGA).

Value

A list of class SL containing the following

orig_R Original correlation matrix.
sl A matrix with general factor loadings, group factor loadings, communalities, and uniquenesses.
L2 Second-order factor loadings.
vars_accounted A matrix of explained variances and sums of squared loadings.
iter The number of iterations needed for convergence in EFA.
settings list. The settings (arguments) used in EFA to get the second-order loadings.
Source


Examples

## Use with an output from the EFAtools::EFA function, both with type EFAtools
EFA_mod <- EFA(test_models$baseline$cormat, N = 500, n_factors = 3,
    type = "EFAtools", method = "PAF", rotation = "promax")
SL_EFAtools <- SL(EFA_mod, type = "EFAtools", method = "PAF")

## Use with an output from the psych::fa function with type psych in SL
fa_mod <- psych::fa(test_models$baseline$cormat, nfactors = 3, n.obs = 500,
    fm = "pa", rotate = "Promax")
SL_psych <- SL(fa_mod, type = "psych", method = "PAF")

## Use more flexibly by entering a pattern matrix and phi directly (useful if
## a factor solution found with another program should be subjected to SL
## transformation)

## For demonstration, take pattern matrix and phi from an EFA output
## This gives the same solution as the first example
EFA_mod <- EFA(test_models$baseline$cormat, N = 500, n_factors = 3,
    type = "EFAtools", method = "PAF", rotation = "promax")
SL_flex <- SL(EFA_mod$rot_loadings, Phi = EFA_mod$Phi, type = "EFAtools",
    method = "PAF")

## Use with a lavaan second-order CFA output
# Create and fit model in lavaan (assume all variables have SDs of 1)
mod <- 'F1 =~ V1 + V2 + V3 + V4 + V5 + V6
F2 =~ V7 + V8 + V9 + V10 + V11 + V12
F3 =~ V13 + V14 + V15 + V16 + V17 + V18
g =~ F1 + F2 + F3'
fit <- lavaan::cfa(mod, sample.cov = test_models$baseline$cormat,
    sample.nobs = 500, estimator = "ml")
SL_lav <- SL(fit, g_name = "g")
Description

Sequential Chi Square Model Tests (SMT) are a factor retention method where multiple EFAs with increasing numbers of factors are fitted and the number of factors for which the Chi Square value first becomes non-significant is taken as the suggested number of factors. Preacher, Zhang, Kim, & Mels (2013) suggested a similar approach with the lower bound of the 90% confidence interval of the Root Mean Square Error of Approximation (RMSEA; Browne & Cudeck, 1992; Steiger & Lind, 1980), and with the Akaike Information Criterion (AIC). For the RMSEA, the number of factors for which this lower bound first falls below .05 is the suggested number of factors to retain. For the AIC, it is the number of factors where the AIC is lowest.

Usage

SMT(
  x,
  N = NA,
  use = c("pairwise.complete.obs", "all.obs", "complete.obs", "everything",
           "na.or.complete"),
  cor_method = c("pearson", "spearman", "kendall")
)

Arguments

x data.frame or matrix. Dataframe or matrix of raw data or matrix with correlations.

N numeric. The number of observations. Needs only be specified if a correlation matrix is used.

use character. Passed to stats::cor if raw data is given as input. Default is "pairwise.complete.obs".

cor_method character. Passed to stats::cor. Default is "pearson".

Details

As a first step in the procedure, a maximum number of factors to extract is determined for which the model is still over-identified (df > 0).

Then, EFAs with increasing numbers of factors from 1 to the maximum number are fitted with maximum likelihood estimation.

For the SMT, first the significance of the chi square value for a model with 0 factors is determined. If this value is not significant, 0 factors are suggested to retain. If it is significant, a model with 1 factor is estimated and the significance of its chi square value is determined, and so on, until a non-significant result is obtained. The suggested number of factors is the number of factors for the model where the chi square value first becomes non-significant.

Regarding the RMSEA, the suggested number of factors is the number of factors for the model where the lower bound of the 90% confidence interval of the RMSEA first falls below the .05 threshold.

Regarding the AIC, the suggested number of factors is the number of factors for the model with the lowest AIC.
In comparison with other prominent factor retention criteria, SMT performed well at determining the number of factors to extract in EFA (Auerswald & Moshagen, 2019). The RMSEA lower bound also performed well at determining the true number of factors, while the AIC performed well at determining the most generalizable model (Preacher, Zhang, Kim, & Mels, 2013).

The SMT function can also be called together with other factor retention criteria in the N_FACTORS function.

**Value**

A list of class SMT containing

- `nfac_chi`: The number of factors to retain according to the significance of the chi square value.
- `nfac_RMSEA`: The number of factors to retain according to the RMSEA lower bound
- `nfac_AIC`: The number of factors to retain according to the AIC
- `p_null`: The p-value for the null model (zero factors)
- `ps_chis`: The p-values for EFA models with increasing numbers of factors, starting with 1 factor
- `RMSEA_LB_null`: The lower bounds of the 90% confidence interval for the RMSEA for the null model (zero factors).
- `RMSEA_LBs`: The lower bounds of the 90% confidence interval for the RMSEA for EFA models with increasing numbers of factors, starting with 1 factor
- `AIC_null`: The AICs for the null model (zero factors)
- `AICs`: The AICs for EFA models with increasing numbers of factors, starting with 1 factor

**Source**


**See Also**

Other factor retention criteria: `CD`, `EKC`, `HULL`, `KGC`, `PARALLEL`

N_FACTORS as a wrapper function for this and all the above-mentioned factor retention criteria.
Examples

```r
SMT_base <- SMT(test_models$baseline$cormat, N = 500)
SMT_base
```

---

**Description**

Various outputs from SPSS FACTOR for the IDS-2 (Grob & Hagmann-von Arx, 2018), the WJIV (3 to 5 and 20 to 39 years; McGrew, LaFort, & Schrank, 2014), the DOSPERT (Frey et al., 2017; Weber, Blais, & Betz, 2002), the NEO-PI-R (Costa, & McCrae, 1992), and four simulated datasets (baseline, case_1a, case_6b, and case_11b, see test_models and population_models) used in Grieder and Steiner (2020).

**Usage**

SPSS

**Format**

A list of 9 containing EFA results for each of the data sets mentioned above. Each of these nine entries is a list of 4 or 8 (see details), of the following structure:

- `paf_comm` (vector) - The final communalities obtained with the FACTOR algorithm with PAF and no rotation. For details, see Grieder and Grob (2019).
- `paf_load` (matrix) - F1 to FN = unrotated factor loadings obtained with the FACTOR algorithm with PAF. Rownames are the abbreviated subtest names.
- `paf_iter` (numeric) - Number of iterations needed for the principal axis factoring to converge.
- `var_load` (matrix) - F1 to FN = varimax rotated factor loadings obtained with the FACTOR algorithm with PAF. Rownames are the abbreviated subtest names.
- `pro_load` (matrix) - F1 to FN = promax rotated factor loadings obtained with the FACTOR algorithm with PAF. Rownames are the abbreviated subtest names.
- `pro_phi` (matrix) - F1 to FN = intercorrelations of the promax rotated loadings.
- `sl` (matrix) - g = General / second order factor of the Schmid-Leiman solution. F1 to FN = First order factors of the Schmid-Leiman solution. h2 = Communalities of the Schmid-Leiman solution. This Schmid-Leiman solution was found using the SPSS Syntax provided by Wolff and Preising (2005).
- `L2` (matrix) - Second order loadings used for the Schmid-Leiman transformation. This Schmid-Leiman solution was found using the SPSS Syntax provided by Wolff and Preising (2005).

**Details**

The IDS-2, the two WJIV, the DOSPERT, and the NEO-PI-R contain all the above entries, while the four simulated datasets contain only `paf_load`, `var_load`, `pro_load`, and `pro_phi`. 
test_models

Four test models used in Grieder and Steiner (2020)

Description

Correlation matrices created from simulated data from four of the population_models cases, each with strong factor intercorrelations. These are used in Grieder & Steiner (2020) to compare the psych and SPSS implementations in this package with the actual implementations of the programs. For details on the cases, see population_models.

Usage

test_models

Format

A list of 4 lists "baseline", "case_1a", "case_6b", and "case_11b", each with the following elements.

- **cormat** (matrix) - The correlation matrix of the simulated data.
- **n_factors** (numeric) - The true number of factors.
- **N** (numeric) - The sample size of the generated data.
Source

Description
A dataframe containing responses to the UPPS personality scale (Whiteside & Lynam, 2005) of 645 participants of Study 2 of Steiner and Frey (2020). Each column are the ratings to one of 45 items to assess urgency, premeditation, perseverance, and sensation seeking. The original data can be accessed via https://osf.io/kxp8t/.

Usage
UPPS_raw

Format
An object of class data.frame with 645 rows and 45 columns.

Source

Description
A list containing the bivariate correlations (N = 1,685) of the 47 cognitive and achievement subtests from the WJ IV for 14- to 19-year-olds from the standardization sample obtained from the WJ-IV technical manual (McGrew, LaForte, & Schrank, 2014). Tables are reproduced with permission from the publisher.

Usage
WJIV_ages_14_19
Format

A list of 2 with elements "cormat" (47 x 47 matrix of bivariate correlations) and "N" (scalar). The correlation matrix contains the following variables:

- **ORLVOC** (numeric) - Oral Vocabulary.
- **NUMSER** (numeric) - Number Series.
- **VRBATN** (numeric) - Verbal Attention.
- **LETPAT** (numeric) - Letter-Pattern Matching.
- **PHNPRO** (numeric) - Phonological Processing.
- **STYREC** (numeric) - Story Recall.
- **VISUAL** (numeric) - Visualization.
- **GENINF** (numeric) - General Information.
- **CONFIRM** (numeric) - Concept Formation.
- **NUMREV** (numeric) - Numbers Reversed.
- **NUMPAT** (numeric) - Number-Pattern Matching.
- **NWDREP** (numeric) - Nonword Repetition.
- **VAL** (numeric) - Visual-Auditory Learning.
- **PICREC** (numeric) - Picture Recognition.
- **ANLSSYN** (numeric) - Analysis-Synthesis.
- **OBJNUM** (numeric) - Object-Number Sequencing.
- **PAIRCN** (numeric) - Pair Cancellation.
- **MEMWRD** (numeric) - Memory for Words.
- **PICVOC** (numeric) - Picture Vocabulary.
- **ORLCMP** (numeric) - Oral Comprehension.
- **SEGMENT** (numeric) - Segmentation.
- **RPCNAM** (numeric) - Rapid Picture Naming.
- **SENREP** (numeric) - Sentence Repetition.
- **UNDDIR** (numeric) - Understanding Directions.
- **SNDBLN** (numeric) - Sound Blending.
- **RETFLU** (numeric) - Retrieval Fluency.
- **SNDAWR** (numeric) - Sound Awareness.
- **LWIDNT** (numeric) - Letter-Word Identification.
- **APPROB** (numeric) - Applied Problems.
- **SPELL** (numeric) - Spelling.
- **PSGCMP** (numeric) - Passage Comprehension.
- **CALC** (numeric) - Calculation.
- **WRTSMP** (numeric) - Writing Samples.
- **WRDATTK** (numeric) - Word Attack.
**ORL RDG** (numeric) - Oral Reading.
**SN RDL** (numeric) - Sentence Reading Fluency.
**MTH FLU** (numeric) - Math Facts Fluency.
**SN WRFL** (numeric) - Sentence Writing Fluency.
**RDGREC** (numeric) - Reading Recall.
**NUMMAT** (numeric) - Number Matrices.
**EDIT** (numeric) - Editing.
**WRD FLU** (numeric) - Word Reading Fluency.
**SP LSN D** (numeric) - Spelling of Sounds.
**RDG VOC** (numeric) - Reading Vocabulary.
**SCI** (numeric) - Science.
**SOC** (numeric) - Social Studies.
**HUM** (numeric) - Humanities.

**Source**

**Description**
A list containing the bivariate correlations (N = 1,251) of the 47 cognitive and achievement subtests from the WJ IV for the 20- to 39-year-olds from the standardization sample obtained from the WJ-IV technical manual (McGrew, LaForte, & Schrank, 2014). Tables are reproduced with permission from the publisher.

**Usage**

**Format**
A list of 2 with elements "cormat" (47 x 47 matrix of bivariate correlations) and "N" (scalar). The correlation matrix contains the following variables:

**ORLVOC** (numeric) - Oral Vocabulary.
**NUMSER** (numeric) - Number Series.
**VRBATN** (numeric) - Verbal Attention.
LETPAT  (numeric) - Letter-Pattern Matching.
PHNPRO  (numeric) - Phonological Processing.
STYREC  (numeric) - Story Recall.
VISUAL  (numeric) - Visualization.
GENINF  (numeric) - General Information.
CONFRM  (numeric) - Concept Formation.
NUMREV  (numeric) - Numbers Reversed.
NUMPAT  (numeric) - Number-Pattern Matching.
NWDREP  (numeric) - Nonword Repetition.
VAL     (numeric) - Visual-Auditory Learning.
PICREC  (numeric) - Picture Recognition.
ANLSYN  (numeric) - Analysis-Synthesis.
OBJNUM  (numeric) - Object-Number Sequencing.
PAIRCN  (numeric) - Pair Cancellation.
MEMWRD  (numeric) - Memory for Words.
PICVOC  (numeric) - Picture Vocabulary.
ORLCMP  (numeric) - Oral Comprehension.
SEGMNT  (numeric) - Segmentation.
RPCNAM  (numeric) - Rapid Picture Naming.
SENREP  (numeric) - Sentence Repetition.
UNDDIR  (numeric) - Understanding Directions.
SNDBLN  (numeric) - Sound Blending.
RETFLU  (numeric) - Retrieval Fluency.
SNDAWR  (numeric) - Sound Awareness.
LWIDNT  (numeric) - Letter-Word Identification.
APPROB  (numeric) - Applied Problems.
SPELL   (numeric) - Spelling.
PSGCMP  (numeric) - Passage Comprehension.
CALC    (numeric) - Calculation.
WRTSMP  (numeric) - Writing Samples.
WRDATK  (numeric) - Word Attack.
ORLRDG  (numeric) - Oral Reading.
SNRDFL  (numeric) - Sentence Reading Fluency.
MTHFLU  (numeric) - Math Facts Fluency.
SNWRFL  (numeric) - Sentence Writing Fluency.
RDGREC  (numeric) - Reading Recall.
NUMMAT  (numeric) - Number Matrices.
EDIT (numeric) - Editing.
WRDFLU (numeric) - Word Reading Fluency.
SPLSND (numeric) - Spelling of Sounds.
RDGVOC (numeric) - Reading Vocabulary.
SCI (numeric) - Science.
SOC (numeric) - Social Studies.
HUM (numeric) - Humanities.

Source

Description
A list containing the bivariate correlations (N = 435) of the 29 cognitive and achievement subtests from the WJ IV for 3- to 5-year-olds from the standardization sample obtained from the WJ IV technical Manual (McGrew, LaForte, & Schrank, 2014). Tables are reproduced with permission from the publisher.

Usage
WJIV_ages_3_5

Format
A list of 2 with elements "cormat" (29 x 29 matrix of bivariate correlations) and "N" (scalar). The correlation matrix contains the following variables:

ORLVOC (numeric) - Oral Vocabulary.
VRBATN (numeric) - Verbal Attention.
LETPAT (numeric) - Phonological Processing.
STYREC (numeric) - Story Recall.
VISUAL (numeric) - Visualization.
GENINF (numeric) - General Information.
CONFIRM (numeric) - Concept Formation.
NUMREV (numeric) - Numbers Reversed.
NUMPAT (numeric) - Number-Pattern Matching.
Woodcock-Johnson IV: ages 40 to 90 plus

Description
A list containing the bivariate correlations (N = 1,146) of the 47 cognitive and achievement subtests from the WJ IV for 40- to 90+-year-olds from the standardization sample obtained from the WJ-IV technical manual (McGrew, LaForte, & Schrank, 2014). Tables are reproduced with permission from the publisher.

Usage
WJIV_ages_40_90
Format

A list of 2 with elements "cormat" (47 x 47 matrix of bivariate correlations) and "N". The correlation matrix contains the following variables:

- **ORLVOC** (numeric) - Oral Vocabulary.
- **NUMSER** (numeric) - Number Series.
- **VRBATN** (numeric) - Verbal Attention.
- **LETPAT** (numeric) - Letter-Pattern Matching.
- **PHNPRO** (numeric) - Phonological Processing.
- **STYREC** (numeric) - Story Recall.
- **VISUAL** (numeric) - Visualization.
- **GENINF** (numeric) - General Information.
- **CONFRM** (numeric) - Concept Formation.
- **NUMREV** (numeric) - Numbers Reversed.
- **NUMPAT** (numeric) - Number-Pattern Matching.
- **NWDREP** (numeric) - Nonword Repetition.
- **VAL** (numeric) - Visual-Auditory Learning.
- **PICREC** (numeric) - Picture Recognition.
- **ANLSYN** (numeric) - Analysis-Synthesis.
- **OBJNUM** (numeric) - Object-Number Sequencing.
- **PAIRCN** (numeric) - Pair Cancellation.
- **MEMWRD** (numeric) - Memory for Words.
- **PICVOC** (numeric) - Picture Vocabulary.
- **ORLCMP** (numeric) - Oral Comprehension.
- **SEGMNT** (numeric) - Segmentation.
- **RPCNAM** (numeric) - Rapid Picture Naming.
- **SENREP** (numeric) - Sentence Repetition.
- **UNDDIR** (numeric) - Understanding Directions.
- **SNDBLN** (numeric) - Sound Blending.
- **RETFLU** (numeric) - Retrieval Fluency.
- **SNDAWR** (numeric) - Sound Awareness.
- **LWIDNT** (numeric) - Letter-Word Identification.
- **APPROB** (numeric) - Applied Problems.
- **SPELL** (numeric) - Spelling.
- **PSG CMP** (numeric) - Passage Comprehension.
- **CALC** (numeric) - Calculation.
- **WRTSMP** (numeric) - Writing Samples.
- **WRDATK** (numeric) - Word Attack.
ORLRDG (numeric) - Oral Reading.
SNRDFL (numeric) - Sentence Reading Fluency.
MTHFLU (numeric) - Math Facts Fluency.
SNWRFL (numeric) - Sentence Writing Fluency.
RDGREC (numeric) - Reading Recall.
NUMMAT (numeric) - Number Matrices.
EDIT (numeric) - Editing.
WRDFLU (numeric) - Word Reading Fluency.
SPLSND (numeric) - Spelling of Sounds.
RDGVOC (numeric) - Reading Vocabulary.
SCI (numeric) - Science.
SOC (numeric) - Social Studies.
HUM (numeric) - Humanities.

Source

Description
A list containing the bivariate correlations (N = 825) of the 47 cognitive and achievement subtests from the WJ IV for 6- to 8-year-olds from the standardization sample obtained from the WJ-IV technical manual (McGrew, LaForte, & Schrank, 2014). Tables are reproduced with permission from the publisher.

Usage
WJIV_ages_6_8

Format
A list of 2 with elements "cormat" (47 x 47 matrix of bivariate correlations) and "N". The correlation matrix contains the following variables:

ORLVOC (numeric) - Oral Vocabulary.
NUMSER (numeric) - Number Series.
VRBATN (numeric) - Verbal Attention.
LETPAT  (numeric) - Letter-Pattern Matching.
PHNPRO  (numeric) - Phonological Processing.
STYREC  (numeric) - Story Recall.
VISUAL  (numeric) - Visualization.
GENINF  (numeric) - General Information.
CONFRM  (numeric) - Concept Formation.
NUMREV  (numeric) - Numbers Reversed.
NUMPAT  (numeric) - Number-Pattern Matching.
NWDREP  (numeric) - Nonword Repetition.
VAL    (numeric) - Visual-Auditory Learning.
PICREC  (numeric) - Picture Recognition.
ANLSYN  (numeric) - Analysis-Synthesis.
OBJNUM  (numeric) - Object-Number Sequencing.
PAIRCN  (numeric) - Pair Cancellation.
MEMWRD  (numeric) - Memory for Words.
PICVOC  (numeric) - Picture Vocabulary.
ORLCMP  (numeric) - Oral Comprehension.
SEGMNT  (numeric) - Segmentation.
RPCNAM  (numeric) - Rapid Picture Naming.
SENREP  (numeric) - Sentence Repetition.
UNDDIR  (numeric) - Understanding Directions.
SNDBLN  (numeric) - Sound Blending.
RETFLU  (numeric) - Retrieval Fluency.
SNDAWR  (numeric) - Sound Awareness.
LWIDNT  (numeric) - Letter-Word Identification.
APPROB  (numeric) - Applied Problems.
SPELL   (numeric) - Spelling.
PSGCMP  (numeric) - Passage Comprehension.
CALC    (numeric) - Calculation.
WRTSMP  (numeric) - Writing Samples.
WRDATK  (numeric) - Word Attack.
ORLRDG  (numeric) - Oral Reading.
SNRDFL  (numeric) - Sentence Reading Fluency.
MTHFLU  (numeric) - Math Facts Fluency.
SNWRFL  (numeric) - Sentence Writing Fluency.
RDGREC  (numeric) - Reading Recall.
NUMMAT  (numeric) - Number Matrices.
**EDIT** (numeric) - Editing.
**WRDFLU** (numeric) - Word Reading Fluency.
**SPLSND** (numeric) - Spelling of Sounds.
**RDGVOC** (numeric) - Reading Vocabulary.
**SCI** (numeric) - Science.
**SOC** (numeric) - Social Studies.
**HUM** (numeric) - Humanities.

**Source**

---

### Description
A list containing the bivariate correlations (N = 1,572) of the 47 cognitive and achievement subtests from the WJ IV for 9- to 13-year-olds from the standardization sample obtained from the WJ-IV technical manual (McGrew, LaForte, & Schrank, 2014). Tables are reproduced with permission from the publisher.

### Usage
`WJIV_ages_9_13`

### Format
A list of 2 with elements "cormat" (47 x 47 matrix of bivariate correlations) and "N". The correlation matrix contains the following variables:

**ORLVOC** (numeric) - Oral Vocabulary.
**NUMSER** (numeric) - Number Series.
**VRBATN** (numeric) - Verbal Attention.
**LETPAT** (numeric) - Letter-Pattern Matching.
**PHNPRO** (numeric) - Phonological Processing.
**STYREC** (numeric) - Story Recall.
**VISUAL** (numeric) - Visualization.
**GENINF** (numeric) - General Information.
**CONFIRM** (numeric) - Concept Formation.
NUMREV (numeric) - Numbers Reversed.
NUMPAT (numeric) - Number-Pattern Matching.
NWDREP (numeric) - Nonword Repetition.
VAL (numeric) - Visual-Auditory Learning.
PICREC (numeric) - Picture Recognition.
ANLSYN (numeric) - Analysis-Synthesis.
OBJNUM (numeric) - Object-Number Sequencing.
PAIRCN (numeric) - Pair Cancellation.
MEMWRD (numeric) - Memory for Words.
PICVOC (numeric) - Picture Vocabulary.
ORLCMP (numeric) - Oral Comprehension.
SEGMNT (numeric) - Segmentation.
RPCNAM (numeric) - Rapid Picture Naming.
SENREP (numeric) - Sentence Repetition.
UNDDIR (numeric) - Understanding Directions.
SNDBLN (numeric) - Sound Blending.
RETFLU (numeric) - Retrieval Fluency.
SNDAWR (numeric) - Sound Awareness.
LWIDNT (numeric) - Letter-Word Identification.
APPROB (numeric) - Applied Problems.
SPELL (numeric) - Spelling.
PSGCMP (numeric) - Passage Comprehension.
CALC (numeric) - Calculation.
WRTSMP (numeric) - Writing Samples.
WRDATK (numeric) - Word Attack.
ORLRDG (numeric) - Oral Reading.
SNRDFL (numeric) - Sentence Reading Fluency.
MTHFLU (numeric) - Math Facts Fluency.
SNWRFL (numeric) - Sentence Writing Fluency.
RDGREC (numeric) - Reading Recall.
NUMMAT (numeric) - Number Matrices.
EDIT (numeric) - Editing.
WRDFLU (numeric) - Word Reading Fluency.
SPLSND (numeric) - Spelling of Sounds.
RDGVOC (numeric) - Reading Vocabulary.
SCI (numeric) - Science.
SOC (numeric) - Social Studies.
HUM (numeric) - Humanities.
Source


Index

* datasets
  DOSPERT, 12
  DOSPERT_raw, 12
  GRiPS_raw, 27
  IDS2_R, 31
  population_models, 50
  RiskDimensions, 62
  SPSS, 69
  test_models, 70
  UPPS_raw, 71
  WJIV_ages_14–19, 71
  WJIV_ages_20–39, 73
  WJIV_ages_3–5, 75
  WJIV_ages_40–90, 76
  WJIV_ages_6–8, 78
  WJIV_ages_9–13, 80
 .compute_vars, 3
 .factor_corres, 4
 .numformat, 4
 .paf_iter, 5
 .parallel_sim, 5

 BARTLETT, 6, 35, 36, 38, 52
 base::mean, 20

 CD, 7, 26, 31, 34, 37, 38, 46, 47, 53, 64, 68
 COMPARE, 9, 53

 DOSPERT, 12
 DOSPERT_raw, 12

 EFA, 13, 19–21, 26, 29, 30, 33, 37, 38, 45, 54, 63–65
 EFA_AVERAGE, 19, 47, 54
 EKC, 9, 24, 31, 34, 38, 46, 47, 55, 64, 68

 FACTOR_SCORES, 26
 future::plan, 30
 future_lapply, 44

 GPArotation::GPFoblq, 15, 21

 GRiPS_raw, 27

 HULL, 9, 26, 28, 33, 34, 37, 38, 46, 48, 56, 64, 68

 IDS2_R, 31

 KGC, 9, 24–26, 31, 32, 37, 38, 45, 46, 48, 56, 68
 KMO, 7, 34, 38, 57

 lavaan, 65

 N_FACTORS, 7–9, 25, 26, 30, 31, 33–36, 36, 45, 46, 58, 63, 64, 68

 OMEGA, 39, 59, 65

 PARALLEL, 9, 25, 26, 28–31, 33, 34, 37, 38, 43, 49, 59, 64, 68

 plot.CD, 46
 plot.EFA_AVERAGE, 47
 plot.EKC, 47
 plot.HULL, 48
 plot.KGC, 48
 plot.PARALLEL, 49
 plot.SCREE, 49
 population_models, 50, 69, 70
 print.BARTLETT, 52
 print.CD, 53
 print.COMPARE, 53
 print.EFA, 54
 print.EFA_AVERAGE, 54
 print.EKC, 55
 print.HULL, 56
 print.KGC, 56
 print.KMO, 57
 print.LOADINGS, 57
 print.N_FACTORS, 58
 print.OMEGA, 58
 print.PARALLEL, 59
 print.SCREE, 60

 83
print.SL, 60
print.SLOADINGS, 61
print.SMT, 61
psych::cortest.bartlett, 7
psych::fa, 14, 15, 21, 65
psych::factor.model, 40, 41
psych::factor.scores, 26, 27
psych::KMO, 35
psych::omega, 39, 41
psych::schmid, 39–41, 64

RiskDimensions, 62
round, 57, 59, 61

schmid, 40
SCREE, 37, 38, 49, 60, 62
SL, 39–41, 60, 64
SMT, 9, 26, 31, 33, 34, 38, 46, 61, 64, 66
SPSS, 69
stats::cor, 6, 8, 14, 15, 21, 25, 29, 33, 35, 37, 44, 63, 67
stats::factanal, 15, 17, 21, 30
stats::varimax, 15, 16, 21
stats::optim, 17, 23

test_models, 69, 70

UPPS_raw, 71

WJIV_ages_14_19, 71
WJIV_ages_20_39, 73
WJIV_ages_3_5, 75
WJIV_ages_40_90, 76
WJIV_ages_6_8, 78
WJIV_ages_9_13, 80