Package: EFAtools (via r-universe)

October 20, 2026

Title Fast and Flexible Implementations of Exploratory Factor Analysis Tools

Version 1.1.0

Description Provides a complete workflow for exploratory factor analysis (EFA). It covers data screening and factorability checks, a suite of factor retention criteria for choosing the number of factors, and factor extraction by principal axis factoring, maximum likelihood, unweighted least squares, or diagonally weighted least squares from Pearson, Spearman, Kendall, polychoric, tetrachoric, or two-stage full-information maximum likelihood correlations. A built-in rotation engine offers a range of orthogonal and oblique rotations, and standard errors for loadings and related quantities can be obtained by analytic, robust, or bootstrap methods. Further tools support model averaging across analytic choices, multigroup EFA with factor congruence, EFA on multiply imputed data, Schmid-Leiman transformation, reliability coefficients (including McDonald's omegas), factor score estimation, data simulation, and power analysis. Computationally intensive procedures are implemented in 'C++' for speed.

Depends R (>= 4.1.0)

License GPL-3

Encoding UTF-8

LazyData true

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

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

Imports psych, stats, ggplot2 (>= 3.4.0), cli, Rcpp, future.apply, checkmate, progressr, rlang, clue, lifecycle

LinkingTo Rcpp, RcppArmadillo, roptim

Suggests testthat (>= 3.0.0), future, GPArotation (>= 2022.4-1), lavaan, lavaan.mi, MASS, mice, nFactors, knitr, rmarkdown, vdiffr, polycor, mnormt, semTools, withr

VignetteBuilder knitr

Config/testthat/edition 3

Config/roxygen2/version 8.0.0

NeedsCompilation yes

Author Markus Steiner [aut, cre] (ORCID: <https://orcid.org/0000-0002-8126-0757>), Silvia Steiner [aut] (ORCID: <https://orcid.org/0000-0002-0118-7722>), William Revelle [ctb], Max Auerswald [ctb], Morten Moshagen [ctb], John Ruscio [ctb], Brendan Roche [ctb], Urbano Lorenzo-Seva [ctb], David Navarro-Gonzalez [ctb], Johan Braeken [ctb], Andreas Soteriades [ctb]

Maintainer Markus Steiner <markus.d.steiner@gmail.com>

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

Date/Publication 2026-08-21 12:20:02 UTC

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

RemoteRef HEAD

RemoteSha ce93c76517ad6d5fe2a15a0b1a19bbd6915ef6b3

Contents

BARTLETT 4
CD 5
COMPARE 6
DOSPERT 8
DOSPERT_raw 9
EFA 9
efa_average 13
EFA_AVERAGE 21
efa_bartlett 25
efa_cd 26
efa_compare 28
efa_ekc 31
efa_fit 33
efa_group 47
efa_hull 52
efa_kgc 56
efa_kmo 58
efa_map 60
efa_mi 62
efa_nest 70
efa_parallel 73
EFA_POOLED 76
efa_power 78
efa_procrustes 84
efa_reliability 87

efa_retain . . . 94
efa_schmid_leiman . . . 99
efa_scores . . . 102
efa_scree . . . 106
efa_screen . . . 108
efa_simulate . . . 112
efa_smt . . . 119
EKC . . . 121
estimate_control . . . 122
FACTOR_SCORES . . . 126
format.efa_retain . . . 128
format.efa_retention . . . 129
GRiPS_raw . . . 129
HULL . . . 130
IDS2_R . . . 132
KGC . . . 133
KMO . . . 134
MAP . . . 135
N_FACTORS . . . 136
NEST . . . 139
OMEGA . . . 140
PARALLEL . . . 145
plot.efa_average . . . 147
plot.efa_compare . . . 148
plot.efa_group . . . 149
plot.efa_power . . . 150
plot.efa_retain . . . 151
plot.efa_retention . . . 152
population_models . . . 152
print.efa . . . 155
print.efa_average . . . 158
print.efa_bartlett . . . 159
print.efa_compare . . . 160
print.efa_control . . . 161
print.efa_group . . . 162
print.efa_kmo . . . 163
print.efa_loadings . . . 164
print.efa_power . . . 166
print.efa_reliability . . . 167
print.efa_retain . . . 168
print.efa_retention . . . 169
print.efa_schmid_leiman . . . 169
print.efa_scores . . . 170
print.efa_screen . . . 171
print.efa_simulated . . . 173
print.efa_sl_loadings . . . 174
print.OMEGA . . . 175
PROCRUSTES . . . 176

residuals.efa . . . 178
RiskDimensions . . . 179
SCREE . . . 180
SL . . . 181
SMT . . . 182
SPSS_23 . . . 183
SPSS_27 . . . 185
test_models . . . 186
UPPS_raw . . . 187
WJIV_ages_14_19 . . . 187
WJIV_ages_20_39 . . . 189
WJIV_ages_3_5 . . . 191
WJIV_ages_40_90 . . . 192
WJIV_ages_6_8 . . . 194
WJIV_ages_9_13 . . . 196

Index **199**

BARTLETT *Bartlett's test of sphericity*

Description

[Superseded]

BARTLETT() has been superseded by efa_bartlett(), which is the recommended interface going forward. It remains available and unchanged so existing code keeps working.

Usage

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

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. The missing-data policy for raw data. Passed to stats::cor() for "pearson", "spearman", and "kendall"; for "poly" / "tetra" the same policies are applied to the raw data before the polychoric estimation, where "all.obs" and "everything" abort on a missing value instead of returning NA correlations. Default is "pairwise.complete.obs".

cor_method — character. Correlation computed from raw data: "pearson", "spearman", or "kendall" (passed to stats::cor()), or "poly" / "tetra" for polychoric / tetrachoric correlations of ordinal / binary data (a two-step estimator). Default is "pearson".

Value

A list of class c("efa_bartlett", "BARTLETT"), identical to the value of efa_bartlett(); see there for the components.

See Also

efa_bartlett()

CD *Comparison data*

Description

[Superseded]

CD() has been superseded by efa_cd(), which is the recommended interface going forward. It remains available and unchanged so existing code keeps working.

Usage

```
CD(
  x,
  n_factors_max = NA,
  N_pop = 10000,
  N_samples = 500,
  alpha = 0.3,
  cor_method = c("pearson", "spearman", "kendall", "poly", "tetra"),
  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.

cor_method	character. One of "pearson", "spearman", or "kendall", passed to stats::cor(). "poly" and "tetra" are not supported because CD compares the data against simulated continuous reference data. Default is "pearson".
max_iter	numeric. The maximum number of iterations after which the iterative PAF procedure inside the comparison-data generation is halted; it does not cap an EFA of x. Default is 50.

Value

An object of class efa_retention, identical to the value of efa_cd(); see there for the components.

See Also

efa_cd()

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

Description

[Superseded]

COMPARE() has been superseded by efa_compare(), which is the recommended interface going forward. It remains available and unchanged so existing code keeps working.

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), the columns of y are matched to those of x by a joint one-to-one assignment that maximizes the total Tucker's congruence coefficient (a standard measure of similarity between two loading vectors) across all columns at once, and each matched column's sign is flipped if needed. This way, mismatched factor order or sign between two solutions does not distort the comparison. It applies to matrices only, and warns when x and y are vectors. If "names", the columns of a matrix – or the elements of a vector – are put in alphabetical order of their names; the rows of a matrix are assumed to be aligned already and are left untouched. If "none", no reordering is done.
corres	logical. Whether factor correspondences should be compared if a matrix is entered. Default is TRUE.
thresh	numeric. The threshold at or above which a loading is classified 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. The number of agreeing decimals below which the report highlights the agreement in red (i.e., if 3 is used, the value is shown in red when the compared numbers agree to fewer than 3 decimals, otherwise 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 from the difference summaries and factor-correspondence classifications. With FALSE, a missing loading makes the correspondence counts undefined (NA). 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, and to name the direction of the signed elementwise differences in the printed report (see print.efa_compare()). Default is "x" and "y".
plot	**[Superseded]** Accepted and validated, but without effect; retained for backwards compatibility. The difference plot is drawn by plot.efa_compare(). Default is TRUE.
plot_red	numeric. Threshold above which to plot the absolute differences in red. Default is .01.

Value

A list of class `c("efa_compare", "COMPARE")`, identical to the value of `efa_compare()`; see there for the components.

See Also

`efa_compare()`

`DOSPERT` *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

A list of 2 with elements "cormat" (40 x 40 matrix of bivariate correlations) and "N" (scalar).

cormat (matrix) - Bivariate correlations of the 40 DOSPERT items, which span the six risk domains (social, recreational, gambling, health/safety, investment, and ethical).

N (numeric) - The sample size the correlations are based on.

Details

The underlying data deposit is licensed under CC BY 4.0 (https://creativecommons.org/licenses/by/4.0/legalcode). These correlations are a derivative of it and are attributed to Frey et al. (2017), as the licence requires.

Source

Weber, E. U., Blais, A.-R., & Betz, N. E. (2002). A domain specific risk-attitude scale: Measuring risk perceptions and risk behaviors. Journal of Behavioral Decision Making, 15(4), 263–290. doi: 10.1002/bdm.414

Frey, R., Pedroni, A., Mata, R., Rieskamp, J., & Hertwig, R. (2017). Risk preference shares the psychometric structure of major psychological traits. Science Advances, 3, e1701381.

https://osf.io/rce7g

DOSPERT_raw *DOSPERT_raw*

Description

A data.frame containing responses to the risk subscale of the Domain Specific Risk Taking Scale (DOSPERT; Weber, Blais, & Betz, 2002) based on the publicly available dataset (at https://osf.io/pjt57/) by Frey, Duncan, and Weber (2023). The items measure risk-taking propensity on five different domains: social, recreational, financial, health/ safety, and ethical.

Usage

```
DOSPERT_raw
```

Format

A data.frame with 3,123 rows (participants) and 30 columns, named by a domain prefix and item number, with six items in each of five risk domains:

ethR_1 to ethR_6 (numeric) - Ethical-domain risk-taking items.

finR_1 to finR_6 (numeric) - Financial-domain risk-taking items.

heaR_1 to heaR_6 (numeric) - Health/safety-domain risk-taking items.

recR_1 to recR_6 (numeric) - Recreational-domain risk-taking items.

socR_1 to socR_6 (numeric) - Social-domain risk-taking items.

Source

Blais, A.-R., & Weber, E. U. (2006). A domain-specific risk-taking (DOSPERT) scale for adult populations. Judgment and Decision Making, 1(1), 33–47. doi: 10.1017/S1930297500000334

Frey, R., Duncan, S. M., & Weber, E. U. (2023). Towards a typology of risk preference: Four risk profiles describe two-thirds of individuals in a large sample of the U.S. population. Journal of Risk and Uncertainty, 66(1), 1–17. doi:10.1007/s11166-022-09398-5

EFA *Exploratory factor analysis (EFA)*

Description

[Superseded]

`EFA()` has been superseded by `efa_fit()`, which is the recommended interface going forward. `efa_fit()` keeps the primary choices (data, factors, estimator, rotation, standard errors) as top-level arguments and collects the estimation and rotation tuning knobs into two control objects, `estimate_control()` and `rotate_control()`. `EFA()` remains available and unchanged – its full flat argument list still works exactly as before – so existing code keeps running.

Usage

```
EFA(
  x,
  n_factors,
  N = NA,
  method = c("PAF", "ML", "ULS", "MINRES", "DWLS"),
  rotation = c("none", "varimax", "equamax", "quartimax", "geominT", "bentlerT",
   "bifactorT", "promax", "oblimin", "quartimin", "simplimax", "bentlerQ", "geominQ",
    "bifactorQ"),
  se = c("none", "information", "sandwich", "np-boot"),
  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", "poly", "tetra", "fiml"),
  b_boot = 1000,
  ci = 0.95,
  random_starts = 100,
  seed = NULL,
  P_type = lifecycle::deprecated(),
  randomStarts = lifecycle::deprecated(),
  ...
)
```

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. Must be at least 1 and smaller than the number of variables (the common factor model is not identified otherwise). Use efa_retain() to decide on a value.
N	numeric. The number of observations. Needs only be specified if a correlation matrix is used; with raw data, N is found from the data instead. • With N = NA, not all fit indices can be computed; a positive N that is very small relative to the number of variables leaves the chi-square-derived indices unavailable as well, with a warning.

- With raw data, N is the number of cases the correlation matrix was actually computed from – see use for the general rule and how missing values change it. Under cor_method = "fiml", use is ignored and N is instead the number of cases carrying at least one observed value.

method — character. The estimator used to fit the EFA; passed to efa_fit() as its estimator argument. One of "PAF", "ML", "ULS", "MINRES" (an accepted alias of "ULS"), or "DWLS"; see the efa_fit() documentation for their properties and data requirements.

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", "simplimax", "bentlerQ", "geominQ", or "bifactorQ"). See the *Rotations* section in Details for their properties and known issues.

se — character. Whether and how to compute standard errors (and matching confidence intervals): "none" (default), "information" (analytic standard errors from the expected Fisher information of the ML solution), "sandwich" (robust "sandwich" standard errors from raw data, which stay reliable under non-normality or a misspecified estimator weight), or "np-boot" (non-parametric bootstrap). The methods differ in their assumptions, their data requirements, and which estimator, rotation, and cor_method combinations they support; see the *Standard errors* section in Details.

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 (2022; "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 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". It is ignored when cor_method = "fiml" (which handles the missingness itself, so every case contributes), and it is overridden to listwise deletion whenever an asymptotic covariance is required (the "DWLS" estimator, or se = "sandwich"), because the covariance must describe the same cases as the correlation matrix.

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, following the original procedure from Kaiser (1958) (see details). 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. 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" reorders the factors by descending explained variance; "ss_factors" reorders the factors by descending (unweighted) 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.

cor_method — character. How the correlation is computed from raw data: "pearson", "spearman", or "kendall" (passed to stats::cor()); "poly" / "tetra" for polychoric / tetrachoric correlations of ordinal / binary data; or "fiml" for a two-stage full-information maximum-likelihood correlation from raw data with missing values. See the *Correlation methods* section in Details for their properties and the combinations they support. Default is "pearson".

b_boot — numeric. The number of bootstrap samples to draw. Default is 1000. Must be at least 2, the smallest number from which a standard error is defined. Under cor_method = "fiml" each bootstrap sample re-runs the EM moment estimation, so a smaller value may be advisable.

ci — numeric. The level of the confidence intervals: the percentile intervals from the bootstrap samples under se = "np-boot", and the analytic Wald intervals under

	se = "information" and se = "sandwich", the corrected two-stage intervals of cor_method = "fiml" included. Must be greater than 0 and smaller than 1. Default is .95 for 95% CIs.
random_starts	numeric. The number of random starts to use in the rotation to guard against local minima. Default is 100.
seed	numeric. An optional seed for the random-number generator.
P_type, randomStarts	**[Superseded]** Former names of p_type and random_starts. Still accepted (silently) for backwards compatibility; please use the new names.
...	Additional arguments passed to the rotation procedure (e.g., maxit for the maximum number of iterations).

Value

The value of efa_fit(), a list of class c("efa", "EFA"); see there for the components.

See Also

efa_fit(), estimate_control(), rotate_control()

---

efa_average *Model averaging across different EFA estimators and types*

Description

Not all EFA procedures always arrive at the same solution. This function allows you perform a number of EFAs from different estimators (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_fit() function) and provide a summary across the different solutions.

Usage

```
efa_average(
  x,
  n_factors,
  N = NA,
  estimator = "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", "poly", "tetra", "fiml"),
  show_progress = TRUE,
  seed = NULL,
  P_type = lifecycle::deprecated()
)
```

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 = NA (default), not all fit indices can be computed.
estimator	character vector. Any combination of "PAF", "ML", and "ULS", to use principal axis factoring, maximum likelihood, or unweighted least squares, respectively, to fit the EFAs. "MINRES" is accepted as a synonym for "ULS" (the same estimator). Default is "PAF". "DWLS", which `efa_fit()` does accept, is deliberately not offered here: it weights each residual correlation by the inverse of its asymptotic variance, which is only available from raw ordinal data analysed with `cor_method = "poly"` or `"tetra"`, whereas every EFA in the grid is fitted to the single correlation matrix computed once from `x`. Fit a DWLS solution with `efa_fit()` directly.
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") take the extraction and rotation tuning of the respective implementation: this package's default procedure, the psych package's, and SPSS's. A specific psych implementation exists

for PAF, ML, varimax, and promax. The SPSS implementation exists for PAF, varimax, and promax. For details, see `efa_fit()`. The factor ordering is the one setting a named `type` does not bring here: every solution in the grid is fitted with the eigenvalue-based ordering, so that the solutions can be realigned to a common target before averaging.

`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. It is only evaluated for the "PAF" solutions run under `type` "none": a named `type` brings the iteration cap that defines it ("SPSS" 25, "psych" 50, and "EFAtools" 300), and "ML" and "ULS" do not iterate this way. 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_fit()`). 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_fit()`). 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_fit()`). 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_fit() and Grieder & Steiner, 2022). 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 if "simplimax" or "oblique" is among the specified rotations. Default is ncol(x), where x is the entered data. It counts loadings, so each value must be a whole number no larger than the number of loadings in the solution; a simplimax fit given anything else fails and is reported as an errored solution in the grid.

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_fit()). 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". It is ignored when cor_method = "fiml", which handles the missingness itself, so every case contributes.

cor_method — character. Correlation computed from raw data: "pearson", "spearman", or "kendall" (passed to stats::cor()), "poly" / "tetra" for polychoric / tetrachoric correlations of ordinal / binary data (a two-step estimator), or "fiml" for a two-stage full-information maximum-likelihood correlation from raw data with missing values. With "fiml" the saturated multivariate-normal mean and covariance are estimated by an EM algorithm assuming the data are missing at random and the standardized covariance is analysed, reproducing psych::corFiml() followed by psych::fa() and lavaan(missing = "two.stage"), *not* lavaan::efa(missing = "ml") (see efa_fit() and the details). Default is "pearson".

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

seed — numeric or NULL. An optional seed making the averaging run reproducible and independent of the number of parallel workers (the grid runs with future_lapply(), for which a parallel plan can be set via future::plan()). It matters whenever a criterion-based rotation is included, since those draw random starts. When

supplied, the caller's random-number stream is restored afterwards, leaving no side effect. Default is NULL.

P_type **[Superseded]** Former name of p_type. Still accepted (silently) for backwards compatibility; please use p_type.

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_fit() 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.

When raw data are supplied, the correlation matrix is computed once before the grid is run and reused for every EFA in it. Under cor_method = "fiml" this means the saturated multivariate-normal moments are EM-estimated a single time (from the raw data with missing values, assuming the data are missing at random) and the resulting two-stage correlation is analysed by every solution in the grid; the EM is not re-run per solution. Under cor_method = "fiml", use does not select cases (every case contributes to the EM). The averaged loadings and communalities are then the two-stage FIML estimates, but the averaged Chi-Square and the indices derived from it (CFI, TLI, RMSEA, AIC, BIC, ECVI) are the ordinary ML/ULS discrepancy statistics on the EM correlation, not the corrected two-stage (Satorra-Bentler) statistics that a standalone efa_fit() with cor_method = "fiml" reports; in particular the averaged AIC and BIC are finite here rather than NA.

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: combinations that resolve to the same model are run only once. Two combinations are the same model only if every setting the fit consumes agrees, and for "PAF" that includes the iteration cap max_iter. Since a named type brings its own cap, a type of c("none", "SPSS") whose specific settings match the SPSS combination in every other respect still gives two "PAF" models, unless max_iter is also set to the cap of the SPSS implementation. We include here a list of arguments that are only evaluated under specific conditions:

The arguments init_comm, criterion, criterion_type, abs_eigen, and max_iter are only evaluated if "PAF" is included in estimator 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 estimator.

Every solution in the grid is fitted with the eigenvalue-based factor ordering, including under a named type: the solutions are realigned to a common target before averaging, so a per-fit ordering (SPSS orders by the sum of squared loadings) would not survive into the averaged result. That target is the first solution the grid retains, in grid order; solutions dropped for an error, non-convergence, or a Heywood case cannot become it. The averaged loadings are therefore in the factor order and sign of that solution. This is the one setting a named type does not carry, and it is visible in the

two places the individual fits are: the solutions returned in `efa_list` are eigenvalue-ordered, and so is the single `efa_fit()` object returned when the grid collapses to one row. Their loadings can therefore appear in a different factor order than the same fit run through `efa_fit()` under that `type`, even though the solutions are the same.

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 (improper) case (a communality at or above 1, or, for `ML`/`ULS`, a uniqueness pinned at the estimator's lower bound). 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 of 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 averaged loading matrix is likewise a cell-wise summary rather than a fitted solution: it is not itself the solution of any EFA, does not in general reproduce the correlation matrix, and need not reproduce the averaged communalities. The fit indices described below are correspondingly the mean (or, under `averaging = "median"`, the median) of the per-solution fit indices, not the fit of the averaged loadings, so the averaged loadings and the reported fit do not describe one and the same model.

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 the chi-square-based indices (the chi-square statistic and its significance, CFI, TLI, RMSEA, AIC, BIC, and ECVI) are NA. The common part accounted for (CAF) index (Lorenzo-Seva, Timmerman, & Kiers, 2011) and the residual-based SRMR and RMSR are still computed for PAF. As a consequence, if only "PAF" is included in the `estimator` argument, averaging is performed for the CAF, SRMR, and RMSR, while the chi-square-based indices are NA. If a combination of "PAF" and "ML" and/or "ULS" are included in the `estimator` argument, the CAF, SRMR, and RMSR are averaged across all non-problematic factor solutions, while the chi-square-based 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, SRMR, and RMSR compared to the chi-square-based indices.

Each reported fit index is summarised across the (non-problematic) solutions in the same descriptive

way: the average, standard deviation, minimum, and maximum of the per-solution values. This includes the chi-square significance level (p_chi), which is therefore the mean (or median) of the per-solution p-values and is purely descriptive; it is not the p-value of any pooled chi-square test.

Value

A list of class c("efa_average", "EFA_AVERAGE") containing the components below. Throughout, range is the width maximum - minimum of each cell across the factor solutions, not the interval [minimum, maximum], and average is the (trimmed) mean or the median, following averaging.

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). average is a cell-wise summary of the solutions and not itself a fitted solution (see Details).

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 if rotation was "none" or only one factor was extracted, otherwise on the rotated loadings. Each entry is a matrix with rows "SS loadings", "Prop Tot Var", and "Prop Comm Var"; the last is omitted for a single-factor solution, where it is identically 1, so those matrices have two rows there and three otherwise.

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 estimator argument contains ML or ULS: Fit indices derived from the unrotated factor loadings: Chi Square (chisq), including significance level, Comparative Fit Index (CFI), Tucker-Lewis Index (TLI), Root Mean Square Error of Approximation (RMSEA), Akaike Information Criterion (AIC), Bayesian Information Criterion (BIC), Expected Cross-Validation Index (ECVI), and the common part accounted for (CAF) index as proposed by Lorenzo-Seva, Timmerman, & Kiers (2011). The residual-based Standardized Root Mean Square Residual (SRMR) and Root Mean Square Residual (RMSR) and the CAF are also computed for PAF; for PAF the remaining (chi-square-based) indices are not available (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 convergence code (0 = converged; for ML and ULS the same codes as stats::optim()'s "L-BFGS-B", for PAF 1 if the maximum number of iterations was reached without

meeting the convergence criterion and 0 otherwise), 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, including `seed` (`NULL` when none was supplied).

If the supplied arguments admit only a single EFA, there is nothing to average across: that one `efa_fit()` object is returned instead, with a warning. Its `settings` are that fit's, with `seed` recording the seed the run was governed by. They therefore describe the concrete arguments the fit ran under rather than the `type` that supplied them: a row taken from a named preset records `type = "none"` together with the preset's resolved values (for example `max_iter = 25` for "SPSS"), and `order_type = "eigen"` as for every other row in the grid.

Source

Grieder, S., & Steiner, M. D. (2022). Algorithmic jingle jungle: A comparison of implementations of principal axis factoring and promax rotation in R and SPSS. Behavior Research Methods, 54, 54–74. doi: 10.3758/s13428-021-01581-x

Hendrickson, A. E., & White, P. O. (1964). Promax: A quick method for rotation to oblique simple structure. British Journal of Statistical Psychology, 17 , 65–70. doi: 10.1111/j.2044-8317.1964.tb00244.x

Lorenzo-Seva, U., Timmerman, M. E., & Kiers, H. A. L. (2011). The Hull Method for Selecting the Number of Common Factors, Multivariate Behavioral Research, 46, 340-364, doi: 10.1080/00273171.2011.564527

Kaiser, H. F. (1958). The varimax criterion for analytic rotation in factor analysis. Psychometrika, 23, 187–200. doi: 10.1007/BF02289233

See Also

Other factor analysis: `efa_fit()`, `efa_group()`, `efa_mi()`, `plot.efa_group()`, `print.efa_group()`

Examples

```
# 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,
                         estimator = c("PAF", "ULS", "ML"), type = "EFAtools",
                         start_method = "psych")


# 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,
                             estimator = c("PAF", "ULS", "ML"))

# 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,
                        estimator = "ML", rotation = "oblique", type = "EFAtools",
                        start_method = "psych")



# Two-stage FIML correlations from raw data with missing values: the EM
# saturated moments are estimated once and the resulting correlation is
# averaged across the grid of EFAs.
x_miss <- GRiPS_raw
x_miss[cbind(1:20, 1)] <- NA
Aver_fiml <- efa_average(x_miss, n_factors = 1, estimator = c("PAF", "ML"),
                        cor_method = "fiml")
```

EFA_AVERAGE *Model averaging across different EFA methods and types*

Description

[Superseded]

EFA_AVERAGE() has been superseded by efa_average(), which is the recommended interface going forward. It remains available and unchanged so existing code keeps working.

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", "poly", "tetra", "fiml"),
  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 = NA (default), not all fit indices can be computed.
method	character vector. Any combination of "PAF", "ML", and "ULS", the estimators to average across; passed to efa_average() as its estimator argument. 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") take the extraction and rotation tuning of the respective implementation: this package's default procedure, the psych package's, and SPSS's. A specific psych implementation exists for PAF, ML, varimax, and promax. The SPSS implementation exists for PAF, varimax, and promax. For details, see efa_fit(). The factor ordering is the one setting a named type does not bring here: every solution in the grid is fitted with the eigenvalue-based ordering, so that the solutions can be realigned to a common target before averaging.
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. It is only evaluated for the "PAF" solutions run under `type` "none": a named `type` brings the iteration cap that defines it ("SPSS" 25, "psych" 50, and "EFAtools" 300), and "ML" and "ULS" do not iterate this way. 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_fit()`). 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_fit()`). 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_fit()`). 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_fit()` and Grieder & Steiner, 2022). 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 if "simplimax" or "oblique" is among the specified rotations. Default is `ncol(x)`, where x is the entered data. It counts loadings, so each value must be a whole number no larger than the number of loadings in the solution; a simplimax fit given anything else fails and is reported as an errored solution in the grid.

P_type — character vector. Any combination of `"norm"` and `"unnorm"`. How the promax target matrix P is computed if `"none"` is among the specified types and `"promax"` or `"oblique"` is among the specified rotations: `"unnorm"` uses the unnormalized target matrix of Hendrickson and White (1964), `"norm"` a normalized one. This frozen argument keeps its original name; `efa_average()` takes the same setting as `p_type`. 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". It is ignored when `cor_method = "fiml"`, which handles the missingness itself, so every case contributes.

cor_method — character. Correlation computed from raw data: `"pearson"`, `"spearman"`, or `"kendall"` (passed to `stats::cor()`), `"poly"` / `"tetra"` for polychoric / tetrachoric correlations of ordinal / binary data (a two-step estimator), or `"fiml"` for a two-stage full-information maximum-likelihood correlation from raw data with missing values. With `"fiml"` the saturated multivariate-normal mean and covariance are estimated by an EM algorithm assuming the data are missing at random and the standardized covariance is analysed, reproducing `psych::corFiml()` followed by `psych::fa()` and `lavaan(missing = "two.stage")`, *not* `lavaan::efa(missing = "ml")` (see `efa_fit()` and the details). Default is "pearson".

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

Value

The value of `efa_average()`, normally a list of class `c("efa_average", "EFA_AVERAGE")`; see there for the components.

See Also

`efa_average()`

`efa_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

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

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. The missing-data policy for raw data. Passed to `stats::cor()` for `"pearson"`, `"spearman"`, and `"kendall"`; for `"poly"` / `"tetra"` the same policies are applied to the raw data before the polychoric estimation, where `"all.obs"` and `"everything"` abort on a missing value instead of returning `NA` correlations. Default is "pairwise.complete.obs".
`cor_method`	character. Correlation computed from raw data: `"pearson"`, `"spearman"`, or `"kendall"` (passed to `stats::cor()`), or `"poly"` / `"tetra"` for polychoric / tetrachoric correlations of ordinal / binary data (a two-step estimator). 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

$$chi^2 = -log(det(R))(N - 1 - (2 * p + 5)/6)$$

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 (`efa_kmo()`) can still be used.

This function was heavily influenced by the psych::cortest.bartlett() function from the psych package.

The efa_bartlett function can also be called together with the (efa_kmo()) function and with factor retention criteria in the efa_retain() function.

Value

A list containing

chisq	The chi square statistic, or NA, with a warning, if N is too small for the Bartlett correction (i.e. $N - 1 - (2p + 5)/6 \leq 0$).
p_value	The p value of the chi square statistic, or NA when chisq is NA.
df	The degrees of freedom for the chi square statistic.
settings	A list of the settings used.

Source

Bartlett, M. S. (1951). The effect of standardization on a Chi-square approximation in factor analysis. Biometrika, 38, 337-344.

See Also

efa_kmo() for another measure to determine suitability for factor analysis.

efa_retain() as a wrapper function for this function, efa_kmo() and several factor retention criteria.

Other factor analysis suitability: efa_kmo(), efa_screen(), print.efa_screen()

Examples

```
efa_bartlett(test_models$baseline$cormat, N = 500)
```

---

efa_cd *Comparison data*

Description

Factor retention method introduced by Ruscio and Roche (2012). The code was adapted from the CD code published by Auerswald and Moshagen (2019), available at https://osf.io/x5cz2/.

Usage

```
efa_cd(
  x,
  n_factors_max = NA,
  N_pop = 10000,
  N_samples = 500,
  alpha = 0.3,
  cor_method = c("pearson", "spearman", "kendall", "poly", "tetra"),
  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.
cor_method	character. One of "pearson", "spearman", or "kendall", passed to stats::cor(). "poly" and "tetra" are not supported because CD compares the data against simulated continuous reference data. Default is "pearson".
max_iter	numeric. The maximum number of iterations after which the iterative PAF procedure inside the comparison-data generation is halted; it does not cap an EFA of x. Default is 50.

Details

Comparison data (CD) extends parallel analysis by reproducing the observed correlation matrix rather than generating random data: datasets with a known factor structure are generated with an increasing number of factors, and the smallest number for which adding a further factor no longer significantly improves the reproduction of the observed eigenvalues is retained (Ruscio & Roche, 2012).

Because it reproduces the observed correlation structure instead of a null model, CD was among the more accurate criteria across a broad range of conditions in Ruscio and Roche (2012). It is, however, the only criterion in this family that requires raw data, and by some margin the most computationally intensive one: a finite population of N_pop cases is generated and N_samples samples are drawn from it at every candidate factor count. It is therefore a good choice when the raw data are at hand and the runtime is acceptable, and a poor one for a quick look at a correlation matrix.

The comparison data are obtained by simulation, so the suggested number of factors varies slightly from run to run. Call base::set.seed() beforehand to make a run reproducible.

Note that if the data contains missing values, these will be removed for the comparison data procedure using stats::na.omit(). If missing data should be treated differently, e.g., by imputation, do this outside efa_cd() and then pass the complete data.

Value

An object of class `efa_retention` (see `print.efa_retention()` and `plot.efa_retention()` for the print and plot methods). Its main fields are:

`n_factors`	A named numeric vector (`"CD"`) with the suggested number of factors according to comparison data results.
`results`	A list with a single record holding the mean RMSE between the eigenvalues of the generated and the entered data per number of factors (used for the plot) and, in `rmse_eigenvalues`, the per-sample RMSE matrix (rows are samples, columns are factor counts; columns beyond the last tested factor count are left as zero).
`settings`	A list of the settings used.

Source

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

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

See Also

`efa_retain()` as a wrapper function for this and the other factor retention criteria.

Other factor retention criteria: `efa_ekc()`, `efa_hull()`, `efa_kgc()`, `efa_map()`, `efa_nest()`, `efa_parallel()`, `efa_retain()`, `efa_scree()`, `efa_smt()`

Examples

```
# determine n factors of the GRiPS
efa_cd(GRiPS_raw, N_pop = 500, N_samples = 20)

# determine n factors of the DOSPERT risk subscale
efa_cd(DOSPERT_raw, N_pop = 500, N_samples = 20)
```

---

efa_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 `efa_compare` containing summary information of the differences of the objects.

Usage

```
efa_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), the columns of y are matched to those of x by a joint one-to-one assignment that maximizes the total Tucker's congruence coefficient (a standard measure of similarity between two loading vectors) across all columns at once, and each matched column's sign is flipped if needed. This way, mismatched factor order or sign between two solutions does not distort the comparison. It applies to matrices only, and warns when x and y are vectors. If "names", the columns of a matrix – or the elements of a vector – are put in alphabetical order of their names; the rows of a matrix are assumed to be aligned already and are left untouched. If "none", no reordering is done.
corres	logical. Whether factor correspondences should be compared if a matrix is entered. Default is TRUE.
thresh	numeric. The threshold at or above which a loading is classified 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. The number of agreeing decimals below which the report highlights the agreement in red (i.e., if 3 is used, the value is shown in red when the compared numbers agree to fewer than 3 decimals, otherwise 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 from the difference summaries and factor-correspondence classifications. With FALSE, a missing loading makes the correspondence counts undefined (NA). 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, and to name the direction of the signed elementwise differences in the printed report (see print.efa_compare()). Default is "x" and "y".

plot — **[Superseded]** Accepted and validated, but without effect; retained for backwards compatibility. The difference plot is drawn by plot.efa_compare(). Default is TRUE.

plot_red — numeric. Threshold above which to plot the absolute differences in red. Default is .01.

Details

digits, m_red, range_red, round_red, print_diff, and plot_red only control how the result is displayed; each is stored in the returned object's settings and can be overridden later without recomputing the comparison – digits, m_red, range_red, round_red, and print_diff in a call to print.efa_compare(), and plot_red in a call to plot.efa_compare().

Value

A list of class efa_compare with the following components:

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 agree in absolute value. The comparison is on magnitudes, so two elements that are equal in size but opposite in sign count as agreeing; signed disagreements are reflected in diff and the mean / median / min / max absolute differences. 0 means the two agree in their integer parts but in no decimal place. NA means there is no agreement at all: either they already differ in their integer parts, or na.rm = FALSE and an element is missing.

`diff_corres` The number of differing variable-to-factor correspondences between x and y, when only the highest loading is considered. `NA` whenever the correspondences were not compared: for vector input, for a matrix with a single column, with `corres = FALSE`, and when a loading is missing under `na.rm = FALSE`.

`diff_corres_cross` The number of differing variable-to-factor correspondences between x and y when all loadings `>= thresh` are considered. `NA` under the same conditions as `diff_corres`.

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

`settings` List of the settings used.

See Also

`efa_fit()` for the solutions being compared, and `efa_procrustes()` to rotate one solution onto another before comparing.

Examples

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

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

# compare the two
efa_compare(EFA_SPSS_6$unrot_loadings, EFA_psych_6$unrot_loadings,
            x_labels = c("SPSS", "psych"))
```

---

efa_ekc *Empirical Kaiser criterion*

Description

The empirical Kaiser criterion incorporates random sampling variations of the eigenvalues from the Kaiser-Guttman criterion (`efa_kgc()`; see Auerswald & Moshagen, 2019; Braeken & van Assen, 2017). The implementation follows Braeken and van Assen (2017).

Usage

```
efa_ekc(
  x,
  N = NA,
  use = c("pairwise.complete.obs", "all.obs", "complete.obs", "everything",
    "na.or.complete"),
```

```
  cor_method = c("pearson", "spearman", "kendall", "poly", "tetra"),
  type = lifecycle::deprecated()
)
```

Arguments

x	data.frame or matrix. Dataframe or matrix of raw data or matrix with correlations.
N	numeric. The number of observations. Only needed if x is a correlation matrix. Must be larger than the number of variables.
use	character. Passed to `stats::cor()` if raw data is given as input. Default is "pairwise.complete.obs".
cor_method	character. Correlation computed from raw data: `"pearson"`, `"spearman"`, or `"kendall"` (passed to `stats::cor()`), or `"poly"` / `"tetra"` for polychoric / tetrachoric correlations of ordinal / binary data (a two-step estimator). Default is `"pearson"`. Note that the EKC reference values rest on the Marchenko-Pastur law for the eigenvalues of a sample correlation matrix of independent variables, which assumes the sampling behaviour of product-moment correlations; with `"poly"` / `"tetra"` (and, to a lesser degree, the rank-based methods) the reference series is therefore an approximation.
type	**[Deprecated]** Accepted and ignored. It selected between two ways to compute the reference values. The `"AM2019"` reference values do not depend on the observed eigenvalues, so they do not apply the empirical correction that defines the criterion, and they are no longer computed.

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 `efa_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 `efa_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).

Value

An object of class `efa_retention` (see `print.efa_retention()` and `plot.efa_retention()` for the print and plot methods). Its main fields are:

n_factors	A numeric vector of length one, named "BvA2017", with the suggested number of factors. The factors up to the first observed eigenvalue that fails to exceed its reference value are retained. The "all-exceed" convention of parallel analysis (efa_parallel()), which retains all J factors when no such crossing is found, cannot be reached here: the reference values are never below 1, while the eigenvalues of a correlation matrix sum to J and are sorted downwards, so the last of them is never above 1.
results	A list with one record, holding the eigenvalues, the reference eigenvalues, and the retained solution used for printing and plotting.
settings	A list with the settings used.

Source

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

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

Zwick, W. R., & Velicer, W. F. (1986). Comparison of five rules for determining the number of components to retain. Psychological Bulletin, 99, 432–442. https://doi.org/10.1037/0033-2909.99.3.432

See Also

efa_retain() as a wrapper function for this and the other factor retention criteria.

Other factor retention criteria: efa_cd(), efa_hull(), efa_kgc(), efa_map(), efa_nest(), efa_parallel(), efa_retain(), efa_scree(), efa_smt()

Examples

```
efa_ekc(test_models$baseline$cormat, N = 500)
```

---

efa_fit *Exploratory factor analysis (EFA)*

Description

This function does an EFA with either PAF, ML, ULS/MINRES, or DWLS with or without subsequent rotation. Estimation and rotation are controlled through the control objects built by estimate_control() and rotate_control(); each accepts a type ("EFAtools", "SPSS", "psych", or "none") that fills in its remaining settings.

Usage

```
efa_fit(
  x,
  n_factors,
  N = NA,
  estimator = c("PAF", "ML", "ULS", "MINRES", "DWLS"),
  rotation = c("none", "varimax", "equamax", "quartimax", "geominT", "bentlerT",
   "bifactorT", "promax", "oblimin", "quartimin", "simplimax", "bentlerQ", "geominQ",
    "bifactorQ"),
  se = c("none", "information", "sandwich", "np-boot"),
  cor_method = c("pearson", "spearman", "kendall", "poly", "tetra", "fiml"),
  use = c("pairwise.complete.obs", "all.obs", "complete.obs", "everything",
    "na.or.complete"),
  estimate_control = NULL,
  rotate_control = NULL,
  b_boot = 1000,
  ci = 0.95,
  seed = NULL,
  ...
)
```

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. Must be at least 1 and smaller than the number of variables (the common factor model is not identified otherwise). Use efa_retain() to decide on a value.
N	numeric. The number of observations. Needs only be specified if a correlation matrix is used; with raw data, N is found from the data instead. • With N = NA, not all fit indices can be computed; a positive N that is very small relative to the number of variables leaves the chi-square-derived indices unavailable as well, with a warning. • With raw data, N is the number of cases the correlation matrix was actually computed from – see use for the general rule and how missing values change it. Under cor_method = "fiml", use is ignored and N is instead the number of cases carrying at least one observed value.
estimator	character. The estimator used to fit the EFA: "PAF" (principal axis factoring), "ML" (maximum likelihood), "ULS" (unweighted least squares; "MINRES" is an accepted alias returning identical results), or "DWLS" (diagonally weighted least squares, for ordinal data). See the *Estimators* section in Details for their properties and data requirements. Lower-case versions (e.g., "paf") are also accepted.
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", "simplimax",

"bentlerQ", "geominQ", or "bifactorQ"). See the *Rotations* section in Details for their properties and known issues.

`se` character. Whether and how to compute standard errors (and matching confidence intervals): "none" (default), "information" (analytic standard errors from the expected Fisher information of the ML solution), "sandwich" (robust "sandwich" standard errors from raw data, which stay reliable under non-normality or a misspecified estimator weight), or "np-boot" (non-parametric bootstrap). The methods differ in their assumptions, their data requirements, and which estimator, rotation, and `cor_method` combinations they support; see the *Standard errors* section in Details.

`cor_method` character. How the correlation is computed from raw data: `"pearson"`, `"spearman"`, or `"kendall"` (passed to `stats::cor()`); `"poly"` / `"tetra"` for polychoric / tetrachoric correlations of ordinal / binary data; or `"fiml"` for a two-stage full-information maximum-likelihood correlation from raw data with missing values. See the *Correlation methods* section in Details for their properties and the combinations they support. Default is "pearson".

`use` character. Passed to `stats::cor()` if raw data is given as input. Default is "pairwise.complete.obs". It is ignored when `cor_method = "fiml"` (which handles the missingness itself, so every case contributes), and it is overridden to listwise deletion whenever an asymptotic covariance is required (the "DWLS" estimator, or `se = "sandwich"`), because the covariance must describe the same cases as the correlation matrix.

`estimate_control`
a control object from `estimate_control()` bundling the estimation control arguments: the `type` preset; the principal-axis-factoring iteration settings `init_comm`, `criterion`, `criterion_type`, `max_iter`, and `abs_eigen`; and the maximum-likelihood `start_method`. Defaults to `estimate_control()`, which uses the `"EFAtools"` type. See `estimate_control()`.

`rotate_control` a control object from `rotate_control()` bundling the rotation control arguments: the `type` preset; Kaiser `normalize`; the convergence `precision`; the factor `order_type`; the varimax/promax settings `varimax_type` and `p_type`; the simplimax/promax `k`; and `random_starts`. Defaults to `rotate_control()`, which uses the `"EFAtools"` type. The estimation and rotation presets are independent. See `rotate_control()` for details.

`b_boot` numeric. The number of bootstrap samples to draw. Default is 1000. Must be at least 2, the smallest number from which a standard error is defined. Under `cor_method = "fiml"` each bootstrap sample re-runs the EM moment estimation, so a smaller value may be advisable.

`ci` numeric. The level of the confidence intervals: the percentile intervals from the bootstrap samples under `se = "np-boot"`, and the analytic Wald intervals under `se = "information"` and `se = "sandwich"`, the corrected two-stage intervals of `cor_method = "fiml"` included. Must be greater than 0 and smaller than 1. Default is .95 for 95% CIs.

`seed` numeric. An optional seed for the random-number generator.

`...` Additional arguments forwarded to the rotation engine (usually not needed); an unrecognized one is an error, not a setting that is silently ignored.

Details

Estimators:

The estimator is chosen with `estimator`.

- **PAF** (principal axis factoring) iteratively estimates the communalities and makes no distributional assumptions, which makes it robust and a good general-purpose default. Because it minimises no likelihood or weighted discrepancy it provides no model chi-square, and hence no chi-square-based fit indices (see *Fit indices*).
- **ML** (maximum likelihood) maximises the normal-theory likelihood. It yields the full set of fit indices and is the only estimator with analytic expected-information standard errors (`se = "information"`), but it assumes multivariate normality and is the most prone to Heywood (improper) cases. Its starting values are set by `start_method`.
- **ULS** (unweighted least squares) minimises the sum of squared correlation residuals. "MINRES" (minimum residual) is the same estimator under a different name and returns identical results. It makes no normality assumption, is robust to mild non-normality, and yields the full set of fit indices.
- **DWLS** (diagonally weighted least squares) is the recommended estimator for ordinal data. It weights each off-diagonal correlation residual by the inverse asymptotic variance of the corresponding polychoric correlation (Muthén, du Toit, & Spisic, 1997), reproducing the loadings of a diagonally weighted least squares fit. It therefore requires raw ordinal data with `cor_method = "poly"` or `"tetra"`. Because the weighting follows the polychoric asymptotic covariance, the matrix and the weights are estimated on the listwise-complete cases. Its fit-index behaviour is described under *Fit indices*.

Correlation methods:

When raw data are supplied, `cor_method` selects how the correlation matrix is computed (it is ignored when a correlation matrix is entered directly).

- **"pearson"** (default), **"spearman"**, and **"kendall"** are passed to `stats::cor()`. The factor model assumes a Pearson correlation, but a rank correlation is analysed on its own scale, not converted to a Pearson-equivalent value; Kendall's tau in particular gives the most attenuated loadings of the three. For ordinal items prefer `"poly"` / `"tetra"` below, which estimate the correlation of the underlying continuous variables.
- **"poly"** / **"tetra"** compute polychoric / tetrachoric correlations for ordinal / binary data, assuming an underlying bivariate-normal latent variable. They use a two-step estimator. The polychoric asymptotic covariance that underlies both the DWLS weights and the scaled (sandwich) statistic relies on large-sample theory that degrades for empty or near-empty response-category combinations; with very sparse cells the resulting weights and standard errors can be unreliable (a warning is issued when empty cells are present), so interpret them with caution and consider collapsing rare categories. Each of the `p(p - 1)/2` variable pairs is a separate numerical optimisation, so a polychoric matrix takes much longer to compute than a Pearson one, and the difference grows quadratically in the number of variables; with `se = "np-boot"` the whole matrix is re-estimated for every bootstrap replicate.
- **"fiml"** estimates a two-stage full-information maximum-likelihood correlation. The saturated multivariate-normal mean and covariance are estimated from raw data with missing values by an EM algorithm assuming the data are missing at random (Yuan, Marshall, & Bentler, 2002; Little & Rubin, 2002), and the standardized covariance is then analysed. The model fit indices are corrected two-stage statistics wherever the correction can be formed

(see *Fit indices*). `"fiml"` uses every case and handles the missingness itself, so `use` is ignored; it supplies a continuous (Pearson-type) correlation only and is therefore not compatible with `estimator = "DWLS"`. Standard errors are available analytically for `estimator = "ML"` or `"ULS"` and, for any estimator, by the non-parametric bootstrap (see *Standard errors*). For multiply imputed data, `efa_mi()` is the alternative route to handling missingness. Both routes assume the values are missing at random (MAR), and which one to prefer is largely practical: FIML is a single, efficient fit and is the simpler default when the analysis model is the whole story, whereas multiple imputation is more flexible when the imputation model should draw on auxiliary variables not in the factor model, or when the same imputations feed several downstream analyses.

Rotations:

A rotation transforms the unrotated loadings toward a simpler, more interpretable pattern; all rotations are performed by rotation engines built into the package. Orthogonal rotations keep the factors uncorrelated, whereas oblique rotations let them correlate (returning a pattern matrix, a structure matrix, and the factor intercorrelations `Phi`) and are usually more realistic for psychological constructs.

For an oblique solution the **pattern** matrix (`rot_loadings`) holds each variable's unique contribution from each factor, with the other factors partialled out; it is what is normally interpreted and reported. The **structure** matrix (`Structure = rot_loadings %*% Phi`) holds the plain variable-factor correlations, which are inflated by the factor intercorrelations. The two coincide only when `Phi` is the identity, which is why an orthogonal rotation returns `rot_loadings` alone.

Orthogonal rotations:

- **varimax** maximises the variance of the squared loadings within each factor (column simplicity). It is the most widely used orthogonal rotation and spreads variance across factors rather than concentrating it in a general factor.
- **quartimax** simplifies the variables (rows) so that each loads mainly on one factor; it tends to produce a strong general factor.
- **equamax** is a Crawford-Ferguson compromise between varimax (column) and quartimax (row) simplicity.
- **geominT** uses a geometric-mean criterion that rewards a sparse pattern and tolerates variables with cross-loadings; a smaller offset `delta` gives a sparser solution but sharper local minima.
- **bentlerT** uses Bentler's invariant pattern simplicity criterion.
- **bifactorT** is the Jennrich-Bentler orthogonal bifactor criterion: a general factor plus group factors (bifactor simple structure). It needs at least two group factors (`n_factors >= 3`): with two factors the criterion is identically zero, so no rotation is performed and the unrotated loadings are returned with a warning.

Oblique rotations:

- **promax** is a fast two-step rotation: a varimax solution is raised to a power (controlled by `k` and `p_type`) to form a target that is then fitted obliquely.
- **quartimin** simplifies the variables (rows), like quartimax, so each loads mainly on one factor, but factors are allowed to correlate.
- **oblimin** is quartimin with a tunable argument, `gam` (default 0, i.e. quartimin itself); turning it up trades some of that row simplicity for more strongly correlated factors, and can drive the solution toward factor collapse, so inspect `Phi` before interpreting a fit with `gam > 0`.

- **simplimax** drives the k smallest loadings toward zero. Its criterion is only piecewise smooth and strongly multimodal, which makes it by far the most start-dependent rotation offered here: different random seeds can reach noticeably different solutions. Because every start is fully optimised (there is no screening stage), raising `random_starts` to several hundred costs proportionally more time but buys a better optimum.
- **bentlerQ** is the oblique Bentler invariant pattern simplicity criterion.
- **geominQ** is the oblique geomin criterion; it handles complex (cross-loading) structure well but is multimodal, so it benefits from more `random_starts` (and uses a more thorough multi-start search internally).
- **bifactorQ** is the oblique (correlated) Jennrich-Bentler bifactor criterion, with the same two-group-factor requirement as **bifactorT**.

Prefer an oblique rotation unless there is a substantive reason to force the factors to be uncorrelated: if `Phi` comes back near zero the oblique solution is essentially the orthogonal one anyway, whereas imposing orthogonality on genuinely correlated factors distorts the pattern.

The criterion-based rotations (all except varimax and promax) are fitted by gradient projection with `random_starts` random starts to guard against local minima; the complexity criteria (simplimax and geominQ in particular) are the most multimodal. The starts are drawn from the random-number generator, so different starts can reach genuinely different optima and such a fit is reproducible only when the generator is controlled: pass `seed`, or call `base::set.seed()` beforehand.

Standard errors:

`se` selects whether and how standard errors (and matching confidence intervals) are computed. Which quantities they cover depends on the method. The analytic methods (`"information"` and `"sandwich"`) cover the unrotated loadings, the uniquenesses and the communalities and, when a rotation is applied, the rotated loadings and – for oblique rotations – the factor correlations and the structure coefficients. The bootstrap (`"np-boot"`) covers the unrotated loadings, the residuals, and the fit indices and, when a rotation is applied, the rotated loadings and – for oblique rotations – the factor correlations and the structure coefficients; it reports no uniqueness or communality standard errors.

- **"none"** (default) computes no standard errors.
- **"information"** returns analytic standard errors from the expected (Fisher) information matrix of the maximum-likelihood solution, and therefore requires `estimator = "ML"` and `cor_method = "pearson"` (or `"fiml"`, see below). The rotated standard errors are obtained by propagating the unrotated-loading covariance through the rotation (Jennrich, 1973); because rotated quantities do not depend on how the unrotated solution happens to be oriented, they are directly comparable across programs. Unlike the bootstrap it also works from a correlation matrix as long as `N` is supplied. These standard errors assume multivariate normality and a correctly specified model; under heavy-tailed data or model misfit they can understate the sampling variability, where `"sandwich"` or `"np-boot"` are more robust.
  The rotated loadings, `Phi`, the structure coefficients, the uniquenesses, and the communalities do not depend on how the unrotated solution happens to be oriented, and so are comparable across programs. The **unrotated** loading standard errors are not: a program using a different orientation will report different unrotated loading standard errors for the same fit. That orientation can also fail to be well defined – for example when two factors are of near-equal strength – in which case the unrotated loadings have no well-defined sampling distribution and their standard errors diverge. `efa_fit()` detects this and returns `NA` for the unrotated

loading standard errors with a warning, rather than reporting a number that looks like a standard error but is an artefact of the orientation; the rotated loadings, `Phi`, the structure coefficients, the uniquenesses, and the communalities are unaffected and still reported. Use those, or `se = "np-boot"`, when the unrotated loadings themselves are the quantity of interest. This detection applies to every analytic route (the Pearson and polychoric paths and the two-stage `cor_method = "fiml"` sandwich alike). A Heywood case (a uniqueness at its lower boundary) is separate and more severe: there the whole `SE`/`CI` block comes back `NA` with a warning, because the standard-error approximation fails for every parameter. The scaled chi-square (from `"sandwich"`, or from `cor_method = "fiml"`) does not rely on that approximation and is still reported, so the fit indices are unaffected.

- **"sandwich"** returns robust ("sandwich") standard errors estimated from raw data, combining the estimator weight with a distribution-free covariance of the correlations, so it stays valid under non-normality and weight misspecification (Browne, 1984; Satorra & Bentler, 1994). It is available either for ordinal data with `cor_method = "poly"` or `"tetra"` and `estimator` one of `"ML"`, `"ULS"`, or `"DWLS"`, or for continuous data with `cor_method = "pearson"` and `estimator = "ML"` or `"ULS"`. It reports the same coverage as `"information"` and additionally fills the model fit's chi-square block with a scaled chi-square (see *Fit indices*); the statistic reported as `chi` is always the scaled-and-shifted one (flagged by `chi_scaled_type`), and a mean-adjusted alternative is returned alongside it as `chi_mean_adjusted`. Because the covariance must describe the same cases as the correlation matrix, the sandwich (like `estimator = "DWLS"`) is computed on the listwise-complete cases; on data with missing values the reported `N`, the correlation matrix, and the point estimate therefore reflect the complete cases regardless of `use`.
- **"np-boot"** draws a non-parametric (case-resampling) bootstrap and needs raw data. A correlation matrix carries no cases to resample; alone among the unsupported combinations this one does not error but warns and downgrades `se` to `"none"`, so the fit is returned without an `SE` slot. It is the most general method – available for any `estimator`, `rotation`, and `cor_method` – and the most robust to non-normality and misfit, at the cost of speed; its intervals are bootstrap percentile intervals. The replicate fits run across replicates with the `future` framework; by default they run sequentially, but registering a plan with `future::plan()` runs them in parallel instead, as the examples show. With a fixed `seed` the bootstrap is reproducible and yields the same result regardless of the number of workers. Under `cor_method = "fiml"` each resample also re-runs the EM moment estimation and is therefore slow, so a smaller `b_boot` may be advisable.
  The percentile intervals are centred on the point estimate for the loadings, the factor correlations, the structure coefficients, and the residuals, but **not** for the indices derived from the chi-square (`RMSEA`, `AIC`, `BIC`, and `ECVI`): a resample already carries the model's own misfit, so those intervals ride upward and can even place the point estimate below their own lower bound. Read them as a spread rather than a range for the point estimate; correcting that shift needs resampling from a population transformed to fit the model (Bollen & Stine, 1992), which is not what this bootstrap does. `CFI` and `TLI` are not affected, being ratios in which the baseline chi-square shifts along with the model one.

The analytic methods (`"information"` and `"sandwich"`) are not available with the `"promax"` or `"simplimax"` rotations, which have no supported analytic route for the rotated standard errors; use `"np-boot"` there. Under `cor_method = "fiml"`, `"information"` and `"sandwich"` instead return, for `estimator = "ML"` or `"ULS"`, corrected two-stage sandwich standard errors (Yuan & Bentler, 2000; Savalei & Bentler, 2009). `estimator = "PAF"` carries no Stage-2 weight to build the sandwich from, so use `se = "np-boot"` there instead.

Fit indices:

For ML and ULS, `efa_fit()` returns the model chi-square (with its p-value and degrees of freedom), the Comparative Fit Index (CFI; Bentler, 1990), the Tucker-Lewis Index (TLI, also called the non-normed fit index; Tucker & Lewis, 1973), the Root Mean Square Error of Approximation (RMSEA) with its 90% confidence interval (Browne & Cudeck, 1992), the Akaike and Bayesian Information Criteria (AIC, BIC), the Expected Cross-Validation Index (ECVI; Browne & Cudeck, 1989), the Root Mean Squared Residual (RMSR), the Standardized Root Mean Squared Residual (SRMR; Bentler, 1995), and the common-part-accounted-for (CAF) index (Lorenzo-Seva, Timmerman, & Kiers, 2011). They come with the independence-baseline statistics `chi_null`, `df_null`, and `p_null`. On the unscaled ML and ULS paths `chi_null` is Bartlett's test of sphericity; a scaled chi-square, and the two-stage statistic of a `cor_method = "fiml"` fit, each carry their own baseline instead.

The degrees of freedom depend on the number of variables and factors; the baseline degrees of freedom `df_null` are $p(p-1)/2$ for p variables.

RMSR is the root mean square of the off-diagonal residuals; SRMR rescales it by a fixed factor that depends only on the number of variables. The print and summary methods show SRMR, not RMSR; RMSR remains in the returned object for backward compatibility. Both are computed over the same residuals, so an unavailable residual leaves both `NA`. (**psych**'s `rms` uses a different divisor and equals RMSR$/\sqrt{2}$; the two are not directly comparable.)

The model chi-square is the Bartlett-corrected discrepancy (matching `stats::factanal()` for ML). For ULS it is the same maximum-likelihood discrepancy, evaluated at the ULS-fitted solution, as in `psych::fa()` – not the least-squares criterion **lavaan** reports as its standard ULS test statistic, so the two are not comparable. AIC and BIC are built on this chi-square and can therefore be negative; ECVI (built on the same chi-square plus a non-negative penalty) cannot. Because of the Bartlett correction, ECVI differs slightly from the uncorrected Browne-Cudeck form reported by **lavaan** and Mplus (their AIC/BIC use an unrelated log-likelihood-based formula, so they are not comparable). On the unscaled ML/ULS path, CFI and TLI are computed on a slightly different discrepancy scale than the reported chi-square, so you cannot recompute one from the other by hand there; on the scaled (sandwich) and FIML paths, the reported chi and chi_null are exactly the pair the indices use.

Which indices are reported depends on the estimator:

- **ML and ULS** compute the full set above.
- **PAF** returns only the descriptive residual indices (RMSR, SRMR, CAF) and df; the printed model-fit block shows CAF and SRMR. The chi-square-based indices are `NA`, because PAF minimises no discrepancy.
- **DWLS** by default returns only RMSR, SRMR, CAF, and df, because the ordinary maximum-likelihood discrepancy is not its fit function. When `se = "sandwich"`, a scaled chi-square and the CFI, TLI, and RMSEA derived from it are reported (AIC and BIC remain `NA`); that statistic is a two-stage correction applied to the polychoric correlation residuals (Browne, 1984), not identical to the full WLSMV test of **lavaan** or Mplus, which also corrects for the response thresholds.
- `cor_method = "fiml"` (with ML or ULS) reports two-stage-corrected statistics (Yuan, Marshall, & Bentler, 2002); AIC, BIC, and ECVI are left `NA`, as for any scaled chi-square. The correction itself can be degenerate – typically with a small sample, a high proportion of missing values, or near-collinear variables – in which case an uncorrected likelihood-ratio statistic is reported in its place with a warning, and the print methods label that line `uncorrected` rather than `scaled`; read its p-value and the CFI, TLI, and RMSEA derived from it as indicative only.

Beyond the estimator, the chi-square and everything derived from it are NA whenever the statistic itself is undefined: when N is not supplied, when the model is underidentified (a negative df), and when a positive N is too small relative to the number of variables and factors for the small-sample correction to remain valid. Each case raises its own warning. The residual summaries (RMSR, SRMR, CAF) and the degrees of freedom are still returned there, but residual size does not establish that a model is identified: below zero degrees of freedom a near-zero residual is an artefact of over-parameterisation, not close fit.

Whenever the chi-square is a scaled one (se = "sandwich", or a cor_method = "fiml" fit whose correction could be formed), AIC, BIC, and ECVI are NA; see the fit_indices entry in Value for the additional components then returned. AIC, BIC, and ECVI are NA on every cor_method = "fiml" fit, including the uncorrected fallback above. Lorenzo-Seva, Timmerman, and Kiers (2011) describe CAF as ranging from 0 to 1, with values near 1 indicating close fit; that does not hold here, where a well-fitting model produces a CAF near 0.5, not near 1. Read it as a relative rather than an absolute measure.

Available combinations:

Not every estimator, rotation, standard-error, and correlation method can be combined:

- **Estimator and correlation method.** estimator = "DWLS" requires ordinal data with cor_method = "poly" or "tetra". cor_method = "fiml" works with PAF, ML, and ULS (not DWLS) and needs raw data with missing values.
- **Standard errors.** se = "information" requires estimator = "ML" and cor_method = "pearson" or "fiml", and can be computed from a correlation matrix when N is supplied. se = "sandwich" requires raw data, with either a polychoric/tetrachoric cor_method (ML, ULS, or DWLS) or a Pearson cor_method (ML or ULS); it is not available for PAF. Under cor_method = "fiml", "information" and "sandwich" are available for ML and ULS only and both return the corrected two-stage sandwich. se = "np-boot" requires raw data and works with any estimator, rotation, and correlation method. Neither "information" nor "sandwich" is available with the "promax" or "simplimax" rotations.
- **Fit indices.** The chi-square-based indices are available for ML and ULS (and, as scaled statistics, for cor_method = "fiml" and for DWLS with se = "sandwich"); PAF and DWLS otherwise report only the descriptive residual indices.

Value

A list of class c("efa", "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	For PAF, the number of iterations until convergence. For ML, ULS, and DWLS, the number of objective-function evaluations used by the optimiser (not the number of optimiser iterations).

`convergence` Integer convergence code (0 = converged), using the codes of `stats::optim()`. For ML and ULS it is the code from the bounded optimiser; for DWLS the fit runs a bounded warm start followed by an unconstrained polish, and the reported code is from the final polish. For PAF it is 1 if the maximum number of iterations was reached without meeting the convergence criterion and 0 otherwise. A non-zero code is also reported with a warning.

`heywood` A named integer vector indicating which variables have a Heywood (improper) case in the unrotated solution; empty if there are none.

`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. Its rows are `"SS loadings"` and `"Prop Tot Var"`, followed by `"Cum Prop Tot Var"`, `"Prop Comm Var"`, and `"Cum Prop Comm Var"`; those last three are omitted for a single-factor solution, where they would only repeat the two above them. So code that indexes a row by name (for example `["Prop Comm Var", ]`) must allow for the two-row form at `n_factors = 1`.

`fit_indices` A named list of fit indices computed from the unrotated loadings. ML and ULS report the full set: the model Chi Square (with its p-value and df), CFI, TLI, RMSEA with its 90% confidence interval, AIC, BIC, ECVI, RMSR, SRMR, and CAF. PAF and DWLS report only RMSR, SRMR, CAF, and df; the Chi-Square-based indices are `NA` there, unless DWLS uses `se = "sandwich"`, which fills the full block from a scaled Chi Square instead. RMSR and SRMR are both included, but the print and summary methods display only SRMR (see *Fit indices* in Details). Whenever the Chi Square is scaled (`se = "sandwich"`, or a `cor_method = "fiml"` fit whose correction could be formed), AIC, BIC, and ECVI are `NA`, and a few extra scaled-statistic fields are appended for advanced diagnostics (see *Fit indices* in Details). The list also carries `Fm`, the estimator's own objective value at the solution – on its own scale, not the discrepancy the Chi Square is built from – and the independence-model baseline `chi_null`, `df_null`, and `p_null` that CFI and TLI are computed from.

`model_implied_R` The model implied correlation matrix.

`residuals` Residual correlations, i.e., orig_R - model_implied_R

`standardized_residuals` Residual correlations standardized by their bootstrap standard errors. Only returned, if `se = "np-boot"`.

`rot_loadings` Loading matrix containing the final rotated loadings. For an oblique rotation this is the pattern matrix – each variable's unique contribution from each factor, with the other factors partialled out – and is the matrix normally interpreted (see *Rotations*).

`Phi` The factor intercorrelations (only for oblique rotations).

`Structure` The structure matrix `rot_loadings %*% Phi`, holding the plain variable-factor correlations, which are inflated by the factor intercorrelations (only for oblique rotations).

`rotmat` The rotation matrix. The rotated loadings are recovered from the unrotated loadings as `unrot_loadings %*% rotmat` for orthogonal rotations and for promax, and as `unrot_loadings %*% t(solve(rotmat))` for the other oblique rotations.

vars_accounted_rot
: Matrix of explained variances and sums of squared loadings. Based on rotated loadings and, for oblique rotations, the factor intercorrelations. Same rows as vars_accounted; it is returned only when a rotation was actually applied, which never happens for a single-factor solution, so it always has the five-row form.

settings
: A list of the settings used, including seed (NULL when none was supplied), input_type ("raw" or "correlation", what x was), and cor_method_used (the correlation method that actually ran, NA_character_ for a correlation-matrix input, which consumes none). cor_method keeps the requested value whether or not it was used. For the criterion rotations fitted by gradient projection it additionally carries rotation_diagnostics, a list summarising the multi-start run:

  - converged: whether the start whose solution is returned reached the convergence tolerance (reported separately from the counts below because the returned solution is the one with the lowest criterion value, which need not be a converged one).
  - n_starts_total: the random_starts random starts plus the rational start.
  - n_optimized: how many of those starts were actually optimized – fewer than n_starts_total whenever the solver screens the random starts and optimizes only the most promising ones.
  - n_converged: how many optimized starts reached the convergence tolerance.
  - n_distinct_minima: how many distinct local optima those converged starts found; more than one means the criterion is multimodal on these data.
  - criterion_spread: the range of the criterion values they attained.
  - criterion_best: the criterion value of the returned solution.

  When normalize = TRUE, criterion_best and criterion_spread are evaluated on the Kaiser-normalized loadings the criterion is optimized on, not on the returned rot_loadings, so they are not directly comparable to a criterion recomputed from the returned loadings.

fiml
: Diagnostics of the FIML correlation's EM estimation, present only for cor_method = "fiml": whether it converged before fiml_max_iter, how many iterations it took (iter), the number of distinct missingness patterns (n_patterns), and the number of cases used (n, the reported N). A converged of FALSE means the analysed correlation is the EM's last iterate, not the true FIML estimate; this is flagged in the printed output. Set fiml_max_iter and fiml_tol through estimate_control().

SE
: A named list of standard error matrices, returned only when se is not "none" (see *Standard errors* in Details for what each method assumes and when a component comes back NA). For se = "np-boot": bootstrap SDs of the loadings (rotated too, if a rotation was applied), the residuals, and the fit indices, plus – for oblique rotations – Phi and the structure coefficients; valid_replicates (and, when rotated, valid_target_rotations) records how many bootstrap replicates each of those is based on. For se = "information" and se = "sandwich": Wald-type SEs for the loadings, the uniquenesses, and the communalities – identical to each other, since communality is just 1 - uniqueness – plus Phi and the

structure coefficients for oblique rotations. "sandwich" stays valid under non-normality; "information" assumes the model is correctly specified.

CI
: A named list of confidence intervals of width ci, matching the components of SE: percentile intervals for se = "np-boot", Wald intervals for se = "information" and se = "sandwich". Only returned if se is not "none".

replicates
: A named list of raw bootstrap replicate arrays – the aligned loadings, Phi, structure coefficients, residuals, and fit indices behind SE and CI – with one replicate per array's last dimension (first dimension for fit_indices). Failed replicates are left NA. Populated only for se = "np-boot"; NULL otherwise.

vcov_unrot_loadings
: The full unrotated-loading covariance matrix behind SE$unrot_loadings: a p * n_factors by p * n_factors matrix in column-major vec(Lambda) order, with rows and columns labelled "<variable>_<factor>". Populated for se = "information" and se = "sandwich" – always the unrotated block, even when a rotation is applied – and NA-filled if the analytic covariance is unreliable (a Heywood case or a singular information matrix); NULL for se = "np-boot" and se = "none". It can be populated even when SE$unrot_loadings is NA: a weakly determined rotational orientation invalidates only the marginal SEs, not the underlying covariance.

Gamma
: The asymptotic covariance of the off-diagonal sample correlations – the meat of the robust sandwich SEs. A p (p - 1) / 2 by p (p - 1) / 2 matrix, rows and columns ordered by utils::combn() over the column pairs and labelled "<var_i>-<var_j>". Populated for se = "sandwich" on the polychoric/tetrachoric and Pearson paths; NULL otherwise (including under cor_method = "fiml", whose meat is not returned). It typically dominates a sandwich fit's size (about 11 MB at p = 50) and is kept because efa_mi() needs it from each per-imputation fit to build the pooled covariance.

Source

Bollen, K. A., & Stine, R. A. (1992). Bootstrapping goodness-of-fit measures in structural equation models. Sociological Methods & Research, 21, 205–229. doi: 10.1177/0049124192021002004

Grieder, S., & Steiner, M. D. (2022). Algorithmic jingle jungle: A comparison of implementations of principal axis factoring and promax rotation in R and SPSS. Behavior Research Methods, 54, 54–74. doi: 10.3758/s13428-021-01581-x

Hendrickson, A. E., & White, P. O. (1964). Promax: A quick method for rotation to oblique simple structure. British Journal of Statistical Psychology, 17 , 65–70. doi: 10.1111/j.2044-8317.1964.tb00244.x

Lorenzo-Seva, U., Timmerman, M. E., & Kiers, H. A. L. (2011). The Hull Method for Selecting the Number of Common Factors, Multivariate Behavioral Research, 46, 340-364, doi: 10.1080/00273171.2011.564527

Kaiser, H. F. (1958). The varimax criterion for analytic rotation in factor analysis. Psychometrika, 23, 187–200. doi: 10.1007/BF02289233

Lawley, D. N., & Maxwell, A. E. (1971). Factor analysis as a statistical method (2nd ed.). Butterworths.

Cudeck, R. (1989). Analysis of correlation matrices using covariance structure models. Psychological Bulletin, 105, 317–327. doi: 10.1037/0033-2909.105.2.317

Olkin, I., & Siotani, M. (1976). Asymptotic distribution of functions of a correlation matrix. In S. Ikeda (Ed.), Essays in probability and statistics (pp. 235–251). Shinko Tsusho.

Jennrich, R. I. (1973). Standard errors for obliquely rotated factor loadings. Psychometrika, 38, 593–604. doi: 10.1007/BF02291497

Zhang, G., & Preacher, K. J. (2015). Factor rotation and standard errors in exploratory factor analysis. Journal of Educational and Behavioral Statistics, 40, 579–603. doi: 10.3102/1076998615606098

Browne, M. W. (1984). Asymptotically distribution-free methods for the analysis of covariance structures. British Journal of Mathematical and Statistical Psychology, 37, 62–83. doi: 10.1111/j.2044-8317.1984.tb00789.x

Satorra, A., & Bentler, P. M. (1994). Corrections to test statistics and standard errors in covariance structure analysis. In A. von Eye & C. C. Clogg (Eds.), Latent variables analysis: Applications for developmental research (pp. 399–419). Sage.

Asparouhov, T., & Muthén, B. (2010). Simple second order chi-square correction. Mplus Technical Appendix.

Muthén, B., du Toit, S. H. C., & Spisic, D. (1997). Robust inference using weighted least squares and quadratic estimating equations in latent variable modeling with categorical and continuous outcomes. Unpublished manuscript.

Yuan, K.-H., & Bentler, P. M. (2000). Three likelihood-based methods for mean and covariance structure analysis with nonnormal missing data. Sociological Methodology, 30, 165–200. doi: 10.1111/0081-1750.00078

Yuan, K.-H., Marshall, L. L., & Bentler, P. M. (2002). A unified approach to exploratory factor analysis with missing data, nonnormal data, and in the presence of outliers. Psychometrika, 67, 95–121. doi: 10.1007/BF02294711

Savalei, V., & Bentler, P. M. (2009). A two-stage approach to missing data: Theory and application to auxiliary variables. Structural Equation Modeling, 16, 477–497. doi: 10.1080/10705510903008238

Little, R. J. A., & Rubin, D. B. (2002). Statistical analysis with missing data (2nd ed.). Wiley.

Bartlett, M. S. (1951). The effect of standardization on a Chi-square approximation in factor analysis. Biometrika, 38, 337–344.

Bentler, P. M. (1990). Comparative fit indexes in structural models. Psychological Bulletin, 107, 238–246. doi: 10.1037/0033-2909.107.2.238

Tucker, L. R., & Lewis, C. (1973). A reliability coefficient for maximum likelihood factor analysis. Psychometrika, 38, 1–10. doi: 10.1007/BF02291170

Browne, M. W., & Cudeck, R. (1989). Single sample cross-validation indices for covariance structures. Multivariate Behavioral Research, 24, 445–455. doi: 10.1207/s15327906mbr2404_4

Browne, M. W., & Cudeck, R. (1992). Alternative ways of assessing model fit. Sociological Methods & Research, 21, 230–258. doi: 10.1177/0049124192021002005

Bentler, P. M. (1995). EQS structural equations program manual. Multivariate Software.

See Also

estimate_control() and rotate_control() for the estimation and rotation tuning knobs. efa_retain() for choosing n_factors, and efa_scores(), efa_reliability(), efa_schmid_leiman(), and efa_compare() for working with the fitted solution.

Other factor analysis: efa_average(), efa_group(), efa_mi(), plot.efa_group(), print.efa_group()

Examples

```
# Principal axis factoring with oblimin rotation
mod_oblimin <- efa_fit(test_models$baseline$cormat, n_factors = 3, N = 500,
                       rotation = "oblimin")
mod_oblimin
summary(mod_oblimin)

# ML estimation with oblimin rotation
mod_oblimin <- efa_fit(test_models$baseline$cormat, n_factors = 3, N = 500,
                       estimator = "ML", rotation = "oblimin")
mod_oblimin
summary(mod_oblimin)

# Tuning knobs are supplied through the control objects. Here the SPSS preset is
# used for the estimation and rotation, with the maximum PAF iterations raised.
mod_spss <- efa_fit(test_models$baseline$cormat, n_factors = 3, N = 500,
                    rotation = "promax",
                    estimate_control = estimate_control(type = "SPSS", max_iter = 500),
                    rotate_control = rotate_control(type = "SPSS"))
mod_spss

# Analytic (expected-information) standard errors for the above
ML_info <- efa_fit(test_models$baseline$cormat, n_factors = 3, N = 500,
                   estimator = "ML", rotation = "oblimin", se = "information")
ML_info
summary(ML_info)


# Robust (sandwich) standard errors and a scaled chi-square for ordinal raw data.
# These need a polychoric/tetrachoric correlation method and estimator ML, ULS, or DWLS.
DWLS_rob <- efa_fit(DOSPERT_raw, n_factors = 6, cor_method = "poly",
                    estimator = "DWLS", rotation = "oblimin", se = "sandwich")
DWLS_rob
summary(DWLS_rob)

# The same robust SEs and scaled chi-square for continuous data: a Pearson
# correlation with estimator ML or ULS (the fourth-moment ADF covariance).
ML_rob <- efa_fit(GRiPS_raw, n_factors = 1, cor_method = "pearson",
                  estimator = "ML", rotation = "none", se = "sandwich")
ML_rob
summary(ML_rob)
```

```
# Two-stage FIML correlations from raw data with missing values: the saturated
# multivariate-normal moments are EM-estimated (assuming the data are missing at
# random) and the standardized covariance is analysed.
x_miss <- GRiPS_raw
x_miss[cbind(1:20, 1)] <- NA
efa_fiml <- efa_fit(x_miss, n_factors = 1, estimator = "ML", cor_method = "fiml")
efa_fiml


## Not run:
# Bootstrap standard errors from raw data, reproducible via a fixed seed and run
# in parallel across replicates. future::plan() returns the plan it replaces, so
# on.exit() puts the session back as it was -- also if the fit fails.
efa_boot <- local({
  old_plan <- future::plan(future::multisession, workers = 2)
  on.exit(future::plan(old_plan), add = TRUE)
  efa_fit(GRiPS_raw, n_factors = 1, estimator = "PAF", rotation = "none",
          se = "np-boot", b_boot = 1000, seed = 42)
})

## End(Not run)
```

---

efa_group *Multigroup exploratory factor analysis*

Description

Fit an exploratory factor analysis in each of several groups at a common number of factors and bring the per-group solutions into one shared orientation so their loadings can be compared. Each group is fitted with `efa_fit()`; the solutions are then aligned either to a symmetric consensus target or to a chosen reference group (see *Alignment*).

Usage

```
efa_group(
  x,
  groups = NULL,
  n_factors,
  N = NA,
  reference_group = NULL,
  b_boot = 0L,
  ci = 0.95,
  seed = NULL,
  delta = 0.1,
  invariance = FALSE,
  se = NULL,
  ...
)
```

Arguments

x	A data frame or matrix of raw data (with groups), or a named list of per-group data sets – either raw data frames/matrices or correlation matrices (all of one kind).
groups	A vector with one value per row of x, giving each row's group. Only used when x is a single raw data set; leave NULL when x is a list. Rows with a missing group value are dropped with a warning.
n_factors	numeric. The common number of factors extracted in every group.
N	numeric. The number of observations per group, used only for correlation-matrix input: either a single value applied to all groups or one value per group. Ignored for raw data, where N is taken from each group's data. Default is NA.
reference_group	The group to align the others to (a group name or an integer index). If NULL (default), orthogonal and unrotated solutions use the symmetric consensus target; oblique solutions with more than one factor fall back to the first group as reference. Supplying a value forces the reference alignment.
b_boot	numeric. The number of non-parametric bootstrap replicates used to form percentile confidence intervals for the between-group Tucker congruences. 0 (the default) skips the bootstrap and returns the congruence point estimates only. Bootstrapping requires raw data; it is skipped with a warning for correlation-matrix input.
ci	numeric. The confidence level for the bootstrap congruence intervals, a single value in (0, 1). Default is 0.95.
seed	numeric or NULL. An optional seed making the analysis reproducible. It covers the per-group fits, some of whose rotations draw random starts, whether or not a bootstrap is run. With a bootstrap, it also makes the result independent of how many parallel workers are used (bootstrap replicates run with future_lapply(), configurable via future::plan()). The caller's random-number stream is restored afterwards, leaving no side effect. Default is NULL.
delta	numeric. The salience threshold for the per-item loading-difference flag table: an item's loading on a factor is flagged for a group pair when the groups' aligned loadings differ by at least delta in absolute value. This is a descriptive salience heuristic, not a significance test; common alternatives are 0.15 and 0.20. The threshold applies to whatever loading metric the chosen rotation produces (pattern coefficients for an oblique rotation). 0 flags every cell. Default is 0.1. The geomin rotations take a criterion parameter of the same name. A delta given directly is always this salience threshold and never reaches the rotation; give the geomin parameter as rotate_control(delta = ...). With rotation = "geominT" or "geominQ", a supplied delta gives a warning that says which of the two applies.
invariance	logical. Whether to add an approximate-invariance verdict per factor and group pair from the Lorenzo-Seva and ten Berge (2006) congruence bands (see *Value*). Default is FALSE.
se	Not used. efa_group() itself sets the standard-error method of the per-group efa_fit() calls, so a supplied value is dropped with a warning. Ask for boot-

strap confidence intervals of the between-group congruences with b_boot. Default is NULL.

... Additional arguments passed to efa_fit() for every group (for example estimator, rotation, or cor_method). The estimate_control() and rotate_control() objects are accepted through ... as well, although they are not declared formals: pass them as estimate_control = / rotate_control = exactly as you would to efa_fit(). A name that is neither an efa_fit() argument nor a rotation-engine extra is rejected.

A rotation-engine extra that shares a name with an efa_group() argument cannot reach the rotation through ..., because the argument takes the name first (for example geomin's delta; see delta above).

Details

Input:

Groups can be supplied in two ways: raw data together with a grouping vector (x a data frame or matrix, groups one value per row), or a named list of per-group data sets in x (with groups left NULL). The list may hold raw data frames or correlation matrices (supply N), but not a mix of the two. All groups must contain the same items in the same order; a different item set or order is an error rather than being silently reordered.

Every group is fitted at the same n_factors. Extra arguments in ... (for example estimator, rotation, cor_method, or an estimate_control() / rotate_control() carrying the tuning knobs) are forwarded unchanged to each efa_fit() call, so the estimator and rotation are common to all groups.

The requested number of factors must be small enough, relative to the number of items, for the n_factors-factor model to be identified for the shared item set. Unlike a single efa_fit() fit – which only warns on an under-identified model – a multigroup fit aborts when this fails, because a shared alignment target across an under-identified group is not interpretable.

Alignment:

A factor solution is identified only up to a rotation of its factors, so the per-group solutions must be brought into a common orientation before their loadings can be compared. Two strategies are available and are chosen automatically:

- **Consensus** (the default for orthogonal rotations and for unrotated solutions): a symmetric target is built across all groups using Generalized Procrustes Analysis (Gower, 1975), and every group's loadings are rotated to it. Because this target's own orientation is arbitrary, it is then rotated once more into a fixed convention (called the gauge), and the same transform is applied to every group. The gauge uses the same simple-structure criterion as the requested rotation, applied to the target itself, so the shared loadings are in the same kind of frame as the per-group solutions they summarise. Where no rotation criterion identifies a unique frame (an unrotated solution, or a two-factor bifactorT request), the target's principal-axes orientation is used instead. Either way the columns are ordered by decreasing sum of squares and signed by their column sums, and the shared orientation – and hence every reported congruence, difference, and flag – does not depend on the order the groups are supplied, to well beyond the precision loadings are reported at.
- **Reference**: every group's loadings are aligned by Procrustes rotation to one reference group's loadings, which are kept fixed. This path is used when reference_group is given, and

is used automatically for oblique rotations because the consensus iteration is not defined for oblique transforms with more than one factor. When an oblique rotation triggers the reference path without an explicit `reference_group`, the first group is used and a message reports this; the requested rotation is never silently changed.

In both cases the returned per-group loadings share the column order and sign of the returned `target`.

Comparing the aligned loadings:

Because the per-group loadings share one orientation, they can be compared cell by cell. `efa_group()` reports a per-pair summary of their differences (`diffs`) and a per-item, per-factor flag table (`flags`) marking cells whose absolute difference reaches `delta`; a bootstrap (`b_boot > 0`) additionally reports, for every cell, whether its difference's confidence interval excludes zero. With `invariance = TRUE`, each factor and group pair also gets an approximate-invariance verdict based on the matched Tucker congruence (see *Value* for the similarity bands and how a bootstrap is used).

Value

An object of class `efa_group`, a list containing:

`loadings` A named list of the aligned per-group loading matrices. Their columns match the columns of `target` in order and sign.

`target` The alignment target: the symmetric consensus target, or the reference group's own loadings.

`Phi` A named list of the aligned per-group factor intercorrelations for an oblique rotation; `NULL` otherwise.

`congruence` Tucker congruence between the aligned group loadings, a list with: `matrices`, a nested list whose `[[g]][[h]]` element is the factor-by-factor congruence matrix between the aligned loadings of groups `g` and `h`; `matched`, a groups-by-groups-by-factors array of the matched-factor congruences (the diagonal of each pairwise matrix); and `degenerate`, a groups-by-groups logical matrix flagging pairs whose congruence is undefined (for example, a near-zero factor), for which the corresponding entries are `NA`. When `b_boot > 0` (raw data), three further elements are added: `matched_se`, the bootstrap standard error of each matched congruence; `matched_ci`, a list of `lower` and `upper` percentile confidence limits (each a groups-by-groups-by-factors array); and `n_boot`, the number of bootstrap replicates that aligned in every group and so contributed to the intervals (a replicate whose fit did not converge is retained, as in `efa_fit()`).

`diffs` A data frame with one row per group pair summarising the differences between their aligned loadings: the mean, median, minimum, and maximum absolute difference, the root-mean-square difference (`rmse`), and `n_flagged`, the number of loading cells whose absolute difference reaches `delta`.

`flags` A data frame with one row per group pair, item, and factor giving the signed loading difference (`diff`), its absolute value (`abs_diff`), and `flagged`, whether it reaches `delta`. When a bootstrap was run (`b_boot > 0`, raw data), `ci_lower`, `ci_upper`, and `ci_excludes_0` add the percentile confidence interval for the difference and whether it excludes zero; otherwise these are `NA`.

`invariance` When `invariance = TRUE`, a data frame with one row per group pair and factor giving the matched Tucker congruence (`phi`), its bootstrap CI lower bound (`phi_lower`, `NA` without a bootstrap), and an approximate-invariance `verdict` based on the Lorenzo-Seva and ten Berge (2006) similarity bands: `phi >= 0.95` is "equal" and `[0.85, 0.95)` is "fair"; congruences `< 0.85`, below their bands, are labelled "incongruent". The verdict is read from `phi_lower` when a bootstrap is available (conservative) and from `phi` otherwise. A wide interval therefore lowers the verdict: `phi` = 0.989 with `phi_lower` = 0.726 is labelled "incongruent", because the band is applied to the lower bound. Tucker's congruence is invariant to a proportional rescaling of a factor's loadings, so a factor can be graded "equal" even when one group's loadings on it are uniformly stronger; read the verdict alongside `diffs`. `NULL` when `invariance = FALSE`.

`efa` The named list of per-group `efa_fit()` objects (each retains its own diagnostics, e.g. `heywood`).

`alignment` The alignment result: the consensus object (see `efa_procrustes()`), or a list with the reference group, the target, and the per-group Procrustes results. On the consensus path, this is the raw Procrustes iteration output: its `target/aligned_loadings` are in a different (pre-gauge) orientation than the gauged `target/loadings` returned above. Use `target/loadings` above for comparisons.

`settings` A list of the settings used, including the per-group `N`, the alignment method, the group that seeded the consensus frame (`alignment_start`, `NULL` on the reference path), the orientation the shared frame was put in (`gauge`: the rotation's own name, `"principal_axes"`, or `"identity"` for a single factor; `NULL` on the reference path), the rotation, the estimator, the input type, whether a bootstrap is available (`can_bootstrap`, `FALSE` for correlation-matrix input), and `seed` (`NULL` when none was supplied).

References

Efron, B., & Tibshirani, R. J. (1993). *An Introduction to the Bootstrap*. Chapman & Hall.

Gower, J. C. (1975). Generalized Procrustes analysis. *Psychometrika*, 40, 33-51. doi: 10.1007/BF02291478

Lorenzo-Seva, U., and ten Berge, J. M. F. (2006). Tucker's congruence coefficient as a meaningful index of factor similarity. *Methodology*, 2, 57-64. doi: 10.1027/1614-2241.2.2.57

See Also

Other factor analysis: `efa_average()`, `efa_fit()`, `efa_mi()`, `plot.efa_group()`, `print.efa_group()`

Examples

```
# Raw data split by a grouping vector (unrotated, consensus alignment)
g <- rep(c("g1", "g2"), length.out = nrow(GRiPS_raw))
mg <- efa_group(GRiPS_raw, groups = g, n_factors = 1)
mg$loadings

# Per-pair difference summary and the per-item salience-flag table
mg$diffs
mg$flags
```

```
# Percentile bootstrap confidence intervals for the between-group congruences, with an
# approximate-invariance verdict read conservatively off the congruence CI lower bound
mg_ci <- efa_group(GRiPS_raw, groups = g, n_factors = 1, b_boot = 100, seed = 42,
                   invariance = TRUE)
mg_ci$congruence$matched_ci
mg_ci$invariance

# A named list of correlation matrices sharing the same items, common
# three-factor model, orthogonal rotation -> symmetric consensus target
bands <- list(age_6_8 = WJIV_ages_6_8$cormat, age_14_19 = WJIV_ages_14_19$cormat)
Ns <- c(WJIV_ages_6_8$N, WJIV_ages_14_19$N)
efa_group(bands, n_factors = 3, N = Ns, rotation = "varimax")

# An oblique rotation aligns to a reference group (reported via a message)
efa_group(bands, n_factors = 3, N = Ns, rotation = "promax")
```

`efa_hull` *Hull method for determining the number of factors to retain*

Description

Implementation of the Hull method suggested by Lorenzo-Seva, Timmerman, and Kiers (2011), with an extension to principal axis factoring. See details for parallelization.

Usage

```
efa_hull(
  x,
  N = NA,
  n_fac_theor = NA,
  estimator = 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", "poly", "tetra"),
  n_datasets = 1000,
  percent = 95,
  decision_rule = c("means", "percentile", "crawford"),
  n_factors = 1,
  estimate_control = NULL,
  ...
)
```

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 efa_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. One plus the larger of this number and the number of factors suggested by efa_parallel is used as the upper bound *J* of factors to extract in the Hull method.
estimator	character. The estimator to use. One of "PAF", "ULS", or "ML", for principal axis factoring, unweighted least squares, and maximum likelihood, respectively. Default is "PAF".
gof	character. The goodness of fit index to use. Either "CAF", "CFI", or "RMSEA", or any combination of them. With the "PAF" estimator, 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 efa_parallel().
use	character. Passed to stats::cor() if raw data is given as input. Default is "pairwise.complete.obs".
cor_method	character. One of "pearson", "spearman", or "kendall", passed to stats::cor(). "poly" and "tetra" are not supported because HULL derives its factor-search bound from an internal parallel analysis against continuous reference data. Default is "pearson".
n_datasets	numeric. The number of datasets to simulate. Must be at least 1. Default is 1000. This is passed to efa_parallel().
percent	numeric. The percentile to take from the simulated eigenvalues. Default is 95. This is passed to efa_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 efa_parallel().
n_factors	numeric. Number of factors to extract if "EFA" is included in eigen_type. Default is 1. This is passed to efa_parallel().
estimate_control	an estimate_control() object with the estimation settings for the efa_fit() fits of the 0 to *J* factor solutions, and for the fit inside the internal efa_parallel() call. NULL (default) uses the efa_fit() defaults. This object carries estimation settings only; the fits are always unrotated, which the hull statistics (CFI, RMSEA, CAF) do not depend on.

... Further arguments passed to efa_fit(), also in efa_parallel(). The estimation tuning knobs are not passed here; they live in estimate_control, a rotation setting is not accepted because the fits are unrotated, and neither are the standard-error arguments (se, b_boot, ci, seed), because the fits are internal steps whose standard errors are not reported.

Details

The Hull method aims to find a model with an optimal balance between model fit and number of parameters, retaining only major factors (Lorenzo-Seva, Timmerman, & Kiers, 2011). It fits 0 to *J* factors – where *J* is the number of factors suggested by parallel analysis (or n_fac_theor, if that is larger), plus one – keeps the solutions on the upper boundary of the convex hull of goodness-of-fit against degrees of freedom, and selects the one at the sharpest elbow, i.e. with the highest *st* value.

Because it trades fit against parsimony instead of testing against a null model of uncorrelated variables, the Hull method does not lose accuracy for the correlated-factor structures where parallel analysis (efa_parallel()) tends to under-extract; the CAF variant in particular was among the more accurate criteria in Auerswald and Moshagen (2019). It needs at least six indicators and fits a model at every candidate factor count, so it is comparatively slow and is not an option for very short scales.

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

The upper bound *J* comes from efa_parallel(), which compares against simulated data, so the suggested number of factors varies slightly from run to run; a criterion-based rotation passed through ... adds its own random starts. Call base::set.seed() beforehand to make a run reproducible; the result is then also independent of the parallel plan.

Note that if gof = "RMSEA" is used, 1 - RMSEA is actually used to compare the different solutions. This is necessary due to how the heuristic to locate the elbow of the hull works.

The solutions are fitted without inequality constraints, so a solution can be inadmissible (a Heywood case, or a fit that did not converge). Only the selected solution is checked for this; if it is inadmissible a warning is raised and the retained number of factors should be interpreted with caution.

The ML estimation method uses the psych::fa() starting values. See also the efa_fit documentation.

Value

An object of class efa_retention (see print.efa_retention() and plot.efa_retention() for the print and plot methods). Its main fields are:

n_factors A named numeric vector with the suggested number of factors for each requested goodness-of-fit index ("CAF", "CFI", and/or "RMSEA").

results A list with one record per goodness-of-fit index, each holding the goodness-of-fit values, the degrees of freedom, the hull membership, and the retained solution used for printing and plotting. Each record also carries st, the elbow sharpness of every solution (see details): the retained solution has the largest value, and the runner-up shows how close the selection was. st is NA for the solutions where it is undefined, that is for those not on the hull and for the two hull endpoints,

which have no neighbouring hull solution on one side. When fewer than three solutions remain on the hull, `st` is undefined throughout and the whole vector is `NA`; the retained solution is then the one with the highest goodness of fit, and a warning says so.

`settings` A list of the settings used, including `n_fac_max`, the upper bound J of the number of factors to extract (see details). For backwards compatibility the estimator is also repeated in `settings$method`.

For backwards compatibility the per-index suggestions are additionally available as the top-level fields `n_fac_CAF`, `n_fac_CFI` and `n_fac_RMSEA`, each `NA` if that index was not requested in `gof`. New code should read them from `n_factors` instead.

Source

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

Lorenzo-Seva, U., Timmerman, M. E., & Kiers, H. A. (2011). The Hull method for selecting the number of common factors. Multivariate Behavioral Research, 46(2), 340-364.

See Also

`efa_retain()` as a wrapper function for this and the other factor retention criteria.

Other factor retention criteria: `efa_cd()`, `efa_ekc()`, `efa_kgc()`, `efa_map()`, `efa_nest()`, `efa_parallel()`, `efa_retain()`, `efa_scree()`, `efa_smt()`

Examples

```
# using PAF (this will print a message if gof is not specified manually
# and CAF will be used automatically)
efa_hull(test_models$baseline$cormat, N = 500, gof = "CAF", n_datasets = 100)

# using ML with all available fit indices (CAF, CFI, and RMSEA)
efa_hull(test_models$baseline$cormat, N = 500, estimator = "ML", n_datasets = 100)

# using ULS with only RMSEA
efa_hull(test_models$baseline$cormat, N = 500, estimator = "ULS", gof = "RMSEA",
         n_datasets = 100)


## Not run:
# using parallel processing (Note: plans can be adapted, see the future
# package for details). future::plan() returns the plan it replaces, so
# on.exit() puts the session back as it was -- also if the call fails.
local({
  old_plan <- future::plan(future::multisession, workers = 2)
  on.exit(future::plan(old_plan), add = TRUE)
  efa_hull(test_models$baseline$cormat, N = 500, gof = "CAF")
})

## End(Not run)
```

efa_kgc *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

```
efa_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", "poly", "tetra"),
  n_factors = 1,
  estimate_control = NULL,
  ...
)
```

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. Default is `c("PCA", "SMC", "EFA")`, i.e. all three; `"EFA"` is the only one that fits a model.
use	character. Passed to `stats::cor()` if raw data is given as input. Default is "pairwise.complete.obs".
cor_method	character. Correlation computed from raw data: `"pearson"`, `"spearman"`, or `"kendall"` (passed to `stats::cor()`), or `"poly"` / `"tetra"` for polychoric / tetrachoric correlations of ordinal / binary data (a two-step estimator). Default is "pearson".
n_factors	numeric. Number of factors to extract if "EFA" is included in `eigen_type`. Default is 1.
estimate_control	an `estimate_control()` object with the estimation settings for the `efa_fit()` fit that provides the communalities when `"EFA"` is included in `eigen_type`.

NULL (default) uses the efa_fit() defaults. The fit is unrotated, so no rotation settings apply.

... Additional arguments passed to efa_fit(). For example, estimator, to change the estimator (PAF is default). The estimation tuning knobs are not passed here; they live in estimate_control, and the standard-error arguments (se, b_boot, ci, seed) are not accepted because the fit is an internal step that keeps only its communalities.

Details

Originally, the Kaiser-Guttman criterion was intended for the use with principal 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 several of the other criteria available here, such as parallel analysis (efa_parallel()), the Hull method (efa_hull()), the empirical Kaiser criterion (efa_ekc()), or sequential chi^2 model tests (efa_smt(); see Auerswald & Moshagen, 2019). Which criteria are informative depends on the data at hand, so rather than substituting one for another, run several of them together and compare their suggestions.

The efa_kgc function can also be called together with other factor retention criteria in the efa_retain() function.

Value

An object of class efa_retention (see print.efa_retention() and plot.efa_retention() for the print and plot methods). Its main fields are:

n_factors A named numeric vector with the suggested number of factors for each requested eigenvalue type ("PCA", "SMC", and/or "EFA").

results A list with one record per eigenvalue type, each holding the eigenvalues and the retained solution used for printing and plotting.

settings A list of the settings used.

Source

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

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

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

Zwick, W. R., & Velicer, W. F. (1986). Comparison of five rules for determining the number of components to retain. Psychological Bulletin, 99, 432–442. https://doi.org/10.1037/0033-2909.99.3.432

See Also

`efa_retain()` as a wrapper function for this and the other factor retention criteria.

Other factor retention criteria: `efa_cd()`, `efa_ekc()`, `efa_hull()`, `efa_map()`, `efa_nest()`, `efa_parallel()`, `efa_retain()`, `efa_scree()`, `efa_smt()`

Examples

```
efa_kgc(test_models$baseline$cormat, eigen_type = c("PCA", "SMC"))
```

---

efa_kmo *Kaiser-Meyer-Olkin criterion*

Description

This function computes the Kaiser-Meyer-Olkin (KMO) criterion overall and for each variable in a correlation matrix. The KMO represents the degree to which each observed variable is predicted by the other variables in the dataset and with this indicates the suitability for factor analysis.

Usage

```
efa_kmo(
  x,
  use = c("pairwise.complete.obs", "all.obs", "complete.obs", "everything",
    "na.or.complete"),
  cor_method = c("pearson", "spearman", "kendall", "poly", "tetra")
)
```

Arguments

x	data.frame or matrix. Dataframe or matrix of raw data or matrix with correlations.
use	character. The missing-data policy for raw data. Passed to `stats::cor()` for `"pearson"`, `"spearman"`, and `"kendall"`; for `"poly"` / `"tetra"` the same policies are applied to the raw data before the polychoric estimation, where `"all.obs"` and `"everything"` abort on a missing value instead of returning `NA` correlations. Default is "pairwise.complete.obs".
cor_method	character. Correlation computed from raw data: `"pearson"`, `"spearman"`, or `"kendall"` (passed to `stats::cor()`), or `"poly"` / `"tetra"` for polychoric / tetrachoric correlations of ordinal / binary data (a two-step estimator). 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 to determine a given correlation matrix's (R) suitability for factor analysis. The index is

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

with $Q = SR^{-1}S$ and S = $(diagR^{-1})^{-1/2}$ where $\sum_{i \neq j} r_{ij}^2$ is the sum of squares of the off-diagonal elements of R and $\sum_{i \neq j} q_{ij}^2$ is the sum of squares of the off-diagonal elements of Q (see also Cureton & D'Agostino, 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 `efa_bartlett()` for another test of suitability for factor analysis.

The `efa_kmo` function can also be called together with the `efa_bartlett()` function and with factor retention criteria in the `efa_retain()` function.

Value

A list containing

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

Source

Kaiser, H. F. (1970). A second generation little jiffy. Psychometrika, 35, 401-415.

Kaiser, H. F. & Rice, J. (1974). Little jiffy, mark IV. Educational and Psychological Measurement, 34, 111-117.

Cureton, E. E. & D'Agostino, R. B. (1983). Factor analysis: An applied approach. Hillsdale, N.J.: Lawrence Erlbaum Associates, Inc.

See Also

`efa_bartlett()` for another measure to determine suitability for factor analysis.

`efa_retain()` as a wrapper function for this function, `efa_bartlett()` and several factor retention criteria.

Other factor analysis suitability: `efa_bartlett()`, `efa_screen()`, `print.efa_screen()`

Examples

```
efa_kmo(test_models$baseline$cormat)
```

`efa_map` *Velicer's minimum average partial (MAP) criterion*

Description

Computes Velicer's Minimum Average Partial (MAP) criterion for determining the number of factors/components to retain. The function implements the original MAP criterion (Velicer, 1976), expressed via the TR2 representation, and the revised TR4 variant proposed by Velicer, Eaton, and Fava (2000).

Usage

```
efa_map(
  x,
  use = c("pairwise.complete.obs", "all.obs", "complete.obs", "everything",
    "na.or.complete"),
  cor_method = c("pearson", "spearman", "kendall", "poly", "tetra")
)
```

Arguments

`x`	A numeric `matrix` or `data.frame`. Can be either (a) a correlation matrix, or (b) raw data (rows = observations, columns = variables) from which correlations are computed.
`use`	Character string specifying the treatment of missing values when computing correlations. Passed to `stats::cor()`. Defaults to `"pairwise.complete.obs"`.
`cor_method`	Character string specifying the correlation coefficient to be computed if raw data are supplied. One of `"pearson"`, `"spearman"`, or `"kendall"` (passed to `stats::cor()`), or `"poly"` / `"tetra"` for polychoric / tetrachoric correlations of ordinal / binary data (a two-step estimator). Defaults to `"pearson"`.

Details

MAP partials successive principal components out of the correlation matrix and, after removing m components, summarizes the off-diagonal partial correlations r^*_{ij} that remain in the m-th partial correlation matrix M (which has a unit diagonal); the suggested number of factors is the m that minimizes the criterion. Two criteria are returned, each rescaling the trace of a matrix power of M by the number of off-diagonal cells $p(p-1)$:

- **TR2 (original MAP; Velicer, 1976):** the average squared off-diagonal partial correlation,

$$\mathrm{TR2}_m = \frac{\mathrm{tr}(M^2) - p}{p(p-1)} = \frac{\sum_{i \neq j} (r^*_{ij})^2}{p(p-1)},$$

  where subtracting p removes the p unit diagonal entries.

- **TR4 (revised MAP; Velicer, Eaton, & Fava, 2000):** the analogous fourth-power summary, formed from the trace of the fourth matrix power,

$$\mathrm{TR4}_m = \frac{\mathrm{tr}(M^4) - p}{p(p-1)}.$$

  Moving from the squared to the fourth power downweights the small partial correlations relative to the large ones, which can sharpen the minimum. Unlike TR2, $\mathrm{tr}(M^4)$ is *not* the sum of the fourth powers of the individual partial correlations; the matrix power is intended and is what Velicer, Eaton, and Fava (2000) describe.

Both criteria are returned for every call and they can suggest different numbers of factors on the same correlation matrix. Both are in use in the literature and neither is treated as the default here, so be sure to state which of the two you report, as you would for any other analysis choice.

MAP is most dependable when the components are well determined, that is with many indicators per factor and substantial loadings. It has a well-documented tendency to under-extract, particularly with few indicators per factor or weak loadings (Zwick & Velicer, 1986; Auerswald & Moshagen, 2019), so it is best read as a lower bound and paired with a criterion that errs in the other direction, such as the Kaiser-Guttman criterion (`efa_kgc()`).

The criterion is evaluated over $m = 0, \ldots, p-1$. Each step standardizes the partial covariance matrix by its residual standard deviations, which requires every residual variance to stay positive. Partialling out all but one component leaves a rank-one residual, so the final point $m = p - 1$ is undefined for most correlation matrices and is routinely returned as `NA`. A residual variance can also reach zero earlier, most often on a near-singular matrix; the search then stops there, the criterion values that could be computed are kept, the remaining values stay `NA`, and a warning (class `efa_map_truncated`) reports how far the grid was searched. In that case the suggested m is the minimum over the evaluated range only, so it should be read together with the returned series.

A non-positive-definite input correlation matrix (e.g. from sampling error) is smoothed with `psych::cor.smooth()`.

Value

An object of class `efa_retention` (see `print.efa_retention()` for the print method). MAP has no plot; `plot.efa_retention()` returns `NULL` with a message for it. Its main elements are:

- `n_factors`: A named numeric vector (`"TR2"`, `"TR4"`) with the index m that minimizes the original (TR2) and revised (TR4) MAP criterion.
- `results`: A list with one record per criterion, each holding the criterion values over m and, in `m_last`, the largest m at which the criterion could be evaluated (see details).
- `settings`: A list containing `use` and `cor_method`.

Source

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

Velicer, W. F. (1976). Determining the number of components from the matrix of partial correlations. *Psychometrika, 41*, 321–327.

Velicer, W. F., Eaton, C. A., & Fava, J. L. (2000). Construct explication through factor or component analysis: A review and evaluation of alternative procedures for determining the number of factors or components. In Goffin, R. D. & Helmes, E. (Eds.), *Problems and Solutions in Human Assessment: Honoring Douglas N. Jackson at Seventy* (pp. 41–71). Boston: Kluwer.

Zwick, W. R., & Velicer, W. F. (1986). Comparison of five rules for determining the number of components to retain. *Psychological Bulletin, 99*, 432–442. https://doi.org/10.1037/0033-2909.99.3.432

See Also

`efa_retain()` as a wrapper function for this and the other factor retention criteria.

Other factor retention criteria: `efa_cd()`, `efa_ekc()`, `efa_hull()`, `efa_kgc()`, `efa_nest()`, `efa_parallel()`, `efa_retain()`, `efa_scree()`, `efa_smt()`

Examples

```
## Example with raw data
res <- efa_map(GRiPS_raw)
res

## Example with a correlation matrix
res2 <- efa_map(test_models$baseline$cormat)
res2
```

---

efa_mi *Exploratory factor analysis on multiple data imputations*

Description

Fits `efa_fit()` to each of several imputed datasets, aligns the factor solutions to a common factor space, and pools the resulting estimates and selected fit quantities across imputations.

Usage

```
efa_mi(
  data_list,
  p = 0.05,
  target_method = c("first_target", "consensus"),
  align_unrotated = c("signed_tucker_congruence", "none", "procrustes"),
  fit_pool_method = c("D2"),
  consensus_args = list(),
  procrustes_args = list(),
  rmsea_ci_level = 0.9,
  rmsr_upper = lifecycle::deprecated(),
  ...
)
```

Arguments

data_list	A list of length m, where m is the number of imputations. Each list element is a data frame or matrix of raw data, or a correlation matrix. See argument x in efa_fit(). A mids object from **mice** must be converted first, with mice::complete(x, "all").
p	Numeric in $(0, 1)$. One minus the confidence level for the pooled confidence intervals, whichever se method produced them ("information", "np-boot", or "sandwich"). For example, p = .05 gives 95% intervals.
target_method	Character. How rotated solutions are aligned across imputations before pooling: "first_target" (the default) aligns every imputation to the first imputation's rotated solution, while "consensus" refines a centroid target by Generalized Procrustes Analysis, started from the medoid imputation so that the pooled rotated solution does not depend on the order of data_list (orthogonal rotations only). See *Aligning solutions across imputations* in Details.
align_unrotated	Character. How unrotated loadings are aligned before pooling: "signed_tucker_congruence" (the default; sign/permutation via Tucker congruence, anchored on the medoid imputation and returned in the extraction's canonical gauge), "procrustes" (orthogonal Procrustes to the first imputation), or "none". See *Aligning solutions across imputations* in Details.
fit_pool_method	Character. Only "D2" is implemented for pooling chi-square-type fit. If no chi-square is available, only residual-based fit and descriptive quantities are returned. See *Pooling the model chi-square and fit indices* in Details.
consensus_args	List of additional arguments controlling the GPA-consensus iteration when target_method = "consensus". Recognised tuning parameters include the convergence tolerances tol and loss_tol, the iteration bounds min_iter and max_iter, the target-update damping alpha, the multi-start controls multi_start and starts, and start, which overrides the medoid imputation the iteration is otherwise started from.
procrustes_args	List of efa_procrustes() algorithm controls for fixed-target alignment, for example oblique_maxit or oblique_random_starts. The loadings A, the alignment Target, the rotation family, and the cross-product S are derived from the imputations and cannot be set here.
rmsea_ci_level	Numeric. Confidence level for the RMSEA CI.
rmsr_upper	**[Deprecated]** Deprecated and ignored. efa_mi() now always computes RMSR the same way, from the unique off-diagonal residuals; SRMR is reported alongside it. Supplying it to efa_mi() signals a deprecation warning; the superseded EFA_POOLED() accepts it silently.
...	Additional arguments passed to efa_fit() (e.g. estimator, rotation, se, n_factors, N). These select the estimator, rotation, standard-error method, and fit indices used for every imputation; see efa_fit() for the available options, their properties, and which combinations are valid. Two of them shape the pooled object rather than a single fit: seed sets the random state once for the

whole `efa_mi()` call – every component bootstrap and every random-start rotation draws from it, so a seeded call is reproducible as a whole, and the caller's random stream is restored afterwards – and `b_boot` sets the number of bootstrap replicates drawn per imputation under `se = "np-boot"`, which is what the pooled within-imputation variances are estimated from and is recorded in `settings$b_boot`. The `estimate_control()` and `rotate_control()` objects are accepted through `...` as well, although they are not declared formals: pass them as `estimate_control =` / `rotate_control =` exactly as you would to `efa_fit()`.

Details

`efa_mi()` is the multiple-imputation route to handling missing data: several imputed datasets are each fitted with `efa_fit()` and the solutions pooled. A single-fit alternative is full-information maximum likelihood, available directly in `efa_fit()` as `cor_method = "fiml"`, which EM-estimates a two-stage correlation from one raw dataset with missing values. Both feed the same correlation-scale EFA core and differ only in how the missingness is handled; FIML is intentionally not routed through `efa_mi()`, which is a multi-fit pooler by construction.

Both routes assume the values are missing at random (MAR). Which one to prefer is largely practical: FIML is a single, efficient fit and is the simpler default when the analysis model is the whole story, whereas multiple imputation is more flexible when the imputation model should draw on auxiliary variables not in the factor model, or when the same imputations feed several downstream analyses.

Standard-error pooling routes:

The pooling pathway is selected automatically from the `se` method recorded on the component `efa_fit()` fits, which must be identical across imputations:

- `se = "none"`: no standard errors are pooled.
- `se = "information"`: the per-imputation expected-information standard errors are pooled with Rubin's (1987) rules (Wald intervals).
- `se = "sandwich"`: the two-stage pooled-inputs (MI2S) approach fits a single model on the Rubin-pooled correlation matrix and asymptotic covariance.
- `se = "np-boot"`: the non-parametric bootstrap replicates are re-aligned to the multiple-imputation target and Rubin-pooled.

On the information and np-boot routes, if pooled standard errors cannot be produced (for example an unreliable analytic covariance or too few bootstrap replicates) the pool falls back to point-estimate-only pooling and downgrades `settings$se` to `"none"`. The MI2S route is the exception: its single fit fuses the point estimates and standard errors through the pooled asymptotic covariance, so a structural failure aborts directly rather than falling back.

Aligning solutions across imputations:

The same `efa_fit()` model is fitted to each imputed dataset and the solutions are put into a common factor space before averaging. For oblique solutions the factor intercorrelations are aligned together with the loadings so the model stays internally consistent.

`target_method` controls how rotated solutions are aligned. `"first_target"` (the default) aligns every imputation to the first imputation's rotated solution by one Procrustes rotation each. `"consensus"` instead refines a centroid target by Generalized Procrustes Analysis (Gower 1975; van Ginkel &

Kroonenberg 2014; Lorenzo-Seva & Van Ginkel 2016), starting from the *medoid* imputation's rotated solution – the one closest in aligned squared distance to all the others. `"consensus"` is supported for orthogonal rotations only.

Factor loadings are only unique up to rotation; "gauge" here means which particular rotation, or orientation, a solution is expressed in. The two methods differ in the rotational gauge the pooled solution ends up in, and so in how it responds to the order of `data_list`. The GPA iteration moves its target toward the centroid but keeps the gauge of the solution it started from, so starting it at the medoid – a property of the set, not of the list order – makes the pooled rotated solution invariant to that order, as the pooled unrotated solution already is. `"first_target"` anchors on the first imputation by construction, so an atypical first imputation fixes the orientation for every other one. Permuting `data_list` therefore moves the pooled pattern – by a few hundredths of a loading unit when imputations are similar, more when they disagree. For oblique rotations, the factor correlations move with it. Where the two anchors coincide – and, more generally, where the imputations agree – the two methods give effectively the same pooled estimate and `"consensus"` is simply the more expensive; where they do not, the pooled patterns differ by the rotation between the two gauges. Passing `start` through `consensus_args` overrides the medoid anchor and makes the consensus order dependent again.

`align_unrotated` controls how unrotated loadings are aligned before pooling: `"signed_tucker_congruence"` (the default) matches them up to factor reordering and sign changes, `"procrustes"` aligns them to the first imputation by orthogonal Procrustes rotation, and `"none"` averages them as returned by `efa_fit()`.

The default anchors the matching on the *medoid* imputation (defined above) rather than on whichever imputation happens to come first, so the pooled *unrotated* solution does not depend on the order of `data_list`. The rotated solution is aligned separately, against a reference chosen by `target_method`, and still depends on that reference.

The pooled unrotated matrix is then returned in the same gauge as a single `efa_fit()` fit. Its identifying constraint differs by extraction method – a principal-axis extraction and a maximum-likelihood extraction fix the rotation differently (Anderson & Rubin 1956; Lawley & Maxwell 1971) – and is read off each component fit automatically, so the pooled matrix can be compared element-by-element with an `efa_fit()` solution. A solution that meets neither constraint – an improper one, say – is left as aligned. The correction is a common orthogonal rotation, so communalities, the total variance accounted for, the model-implied correlation matrix, the residuals, and RMSR are unchanged; only the split of variance across factors moves. `"procrustes"` and `"none"` keep their first-imputation anchor and are returned as aligned.

Pooling point estimates:

Point estimates are pooled by arithmetic averaging after alignment. For oblique rotations the structure matrix is recomputed from the pooled pattern matrix and pooled factor correlations, $Structure = \Lambda\Phi$, and communalities are the diagonal of the reproduced correlation matrix, $diag(\Lambda\Phi\Lambda')$ for oblique rotations and $diag(\Lambda\Lambda')$ otherwise. Residuals are not averaged across imputations; they are the pooled observed correlation matrix minus the model-implied correlation of the pooled solution, so RMSR/SRMR are based on these pooled residuals. Both are returned, though the print and summary methods show SRMR only.

Pooling the model chi-square and fit indices:

The model chi-square and the indices derived from it (ECVI and the descriptive AIC/BIC) are pooled with the D2 rule (Li, Meng, Raghunathan & Rubin, 1991), not arithmetically averaged. RMSEA is pooled by the same rule but from a second D2 pool of the per-imputation discrepancies

taken on the uncorrected $N - 1$ scale. The printed RMSEA therefore does not reconcile by hand with the printed chi-square; the statistic it is formed from is `chi_cfi` in `mi_diagnostics`. Because D2 shrinks the pooled chi-square in proportion to the between-imputation variability, the pooled RMSEA can fall below the mean of the per-imputation RMSEAs; read it together with the per-imputation fit. The incremental indices CFI (Bentler, 1990) and TLI (Tucker & Lewis, 1973) are instead the average of the per-imputation indices, which keeps them consistent with the component fits and avoids out-of-range values; the separately pooled model and baseline chi-squares those indices would be formed from remain available in `mi_diagnostics`. AIC and BIC, if returned, are chi-square-derived descriptive quantities and are not likelihood-based MI information criteria. They are reported only where the component fits report them: whenever a component withholds them – any `cor_method = "fiml"` fit, and any fit whose chi-square is a scaled statistic, such as `se = "sandwich"` – the pooled AIC, BIC, and ECVI are `NA` too, matching what `efa_fit()` returns for a single such fit. On the sandwich/MI2S route the chi-square is the single fit's scaled statistic rather than a D2 pool.

Bootstrap pooling (np-boot):

If each component `efa_fit()` call was run with `se = "np-boot"`, pooled bootstrap SEs and Wald-type MI confidence intervals are computed for loadings, communalities, residuals, and, when applicable, factor correlations and structure coefficients. The unrotated bootstrap replicates are re-aligned to the final MI target before the within-imputation covariance is estimated, and Rubin pooling is applied with $T = Ubar + (1 + 1/m)B$. The confidence level of the pooled intervals is set by `p`, not by the component `efa_fit()` calls' `ci`.

Analytic pooling (information):

With `se = "information"`, the analytic unrotated-loading and uniqueness SEs returned by each fit are pooled element-wise with Rubin's rules ($T = Ubar + (1 + 1/m)B$), with Wald intervals on the plain Rubin (1987) degrees of freedom (the analytic loadings are asymptotically normal, so the Barnard-Rubin (1999) adjustment reduces to this form). NA propagation is fail-closed: if any imputation is NA at an element, all pooled outputs for that element are NA. When a rotation was requested, the rotated loadings, communalities, and (for oblique rotations) factor correlations and structure coefficients are pooled as well; residual SE pooling is available only on the bootstrap path. Under `align_unrotated = "procrustes"` the full unrotated covariance `vcov_unrot_loadings` (populated by `se = "information"`) is propagated through the alignment, so it must be present and reliable on every fit. The default alignment also mixes loading columns, through the common canonical-gauge rotation, and so propagates the same covariance; where a fit does not carry it, the unrotated standard errors are returned as `NA` rather than aborting, and the remaining families still pool.

A rotated-loading standard error is conditional on the rotation criterion used (see References). For both orthogonal and oblique rotations the within-imputation variance is therefore each fit's own criterion-aware delta-method rotated SE (the quantity `efa_fit()` returns), reused after a signed-permutation alignment to the MI target, and the between-imputation variance is the sample variance of the aligned rotated loadings. This is a deliberate approximation – each SE is conditional on its own fit's rotation optimum rather than on a common gauge – and is flagged by `MI$<param>$method = "signed_permutation_approx"`. Communalities are rotation-invariant and pool element-wise. For a fully gauge-consistent rotated uncertainty, cross-check with `se = "np-boot"`.

Two-stage pooling (sandwich / MI2S):

With `se = "sandwich"` (robust SEs from a polychoric/tetrachoric or continuous-Pearson asymptotic covariance), pooling follows the two-stage, pooled-inputs approach (Chung & Cai 2019; Sriutaisuk, Liu, Chung, Kim & Gu 2025): the correlation matrix and the asymptotic covariance of its off-diagonal entries are Rubin-pooled across imputations, and a single `EFA` model is fitted to the pooled correlation with the pooled covariance, $\tilde{\Gamma}$, as the robust meat (its diagonal as the weights for `estimator = "DWLS"`). Because there is only one fit and one rotational gauge, this route bypasses the per-imputation alignment: `target_method` and `align_unrotated` do not apply. The fitted object carries native scaled-shifted chi-square statistics and sandwich SEs that already reflect the multiple-imputation uncertainty, so the chi-square is not D2-pooled and the likelihood-ratio-based AIC/BIC/ECVI are `NA`; it is returned in the `mi_fit` slot, with the per-imputation `fits` retained for diagnostics. The pooled fit uses the same `estimate_control()` and `rotate_control()` tuning (including any rotation-engine extras) as the per-imputation fits. At least 20 imputations are recommended for the scaled-shifted statistic, and more (around 100) at higher rates of missingness (Sriutaisuk et al. 2025). The polychoric/tetrachoric (ordinal) case is the primary, best-evaluated target; the continuous-Pearson case uses the same recipe but is less benchmarked.

Value

A list of class `c("efa_mi", "EFA_POOLED", "efa", "EFA")` containing pooled estimates, residuals, fit indices, the individual fits, and MI diagnostics. The trailing legacy classes keep `inherits(x, "EFA_POOLED")` and the single-fit EFA accessors and S3 dispatch working. In addition to the slots inherited from `efa_fit()` (including `SE`, `CI`, and, on the bootstrap path, `replicates`), the object carries:

SE, CI Pooled standard errors and confidence intervals, named as `efa_fit()` names them: where a pooled communality standard error and interval are produced, they are `SE$communalities` and `CI$communalities` on every route. The Rubin routes (`se = "information"`, `se = "np-boot"`) additionally return them under the compatibility alias `h2`, which holds the same values. The analytic route builds the communality family only when a rotation was requested; an unrotated analytic pool reports `uniquenesses` instead.

fit_indices The pooled fit indices. Every route reports `chi`, `df`, `p_chi`, `CAF`, `CFI`, `TLI`, `RMSEA`, `RMSEA_LB`, `RMSEA_UB`, `AIC`, `BIC`, `ECVI`, `RMSR`, `SRMR`, `chi_null`, `df_null`, `p_null`, and `pool_method` under those names and in that order. `pool_method` records the rule the model chi-square was pooled with (`"D2"`); it is `NA` on the `se = "sandwich"` (MI2S) path, which fits once on the pooled inputs and reports that fit's own scaled statistic rather than pooling several. On that path a few extra scaled-statistic fields are appended after the common block, for advanced diagnostics.

standardized_residuals The pooled residuals divided by their pooled bootstrap standard errors, with a zero diagonal. Returned on the `se = "np-boot"` path only, the one route that pools a residual standard error.

MI Multiple-imputation diagnostics for each pooled parameter family. On the bootstrap path: `unrot_loadings`, `communalities`, `residuals`, optionally `rot_loadings`, `Phi`, `Structure`, and `fit_indices_descriptive`, plus integer vectors `bootstrap_source_failures` (replicates the component `efa_fit()` could not fit), `bootstrap_rotation_failures` (replicates whose Procrustes alignment to the target was invalid), and `bootstrap_rotation_valid` (those that entered the pool, `B - source - rotation` failures). Both paths use the plain Rubin (1987) df. On the analytic path (`se = "information"`): `unrot_loadings` and `uniquenesses`, plus, when a rotation was requested, `rot_loadings`, `communalities`, and (oblique) `Phi` and `Structure`.

The communality family is keyed by its canonical name here, without the SE/CI alias, so each family is counted once in the printed FMI/RIV summary. Each per-family entry is a list with RIV (relative increase in variance), FMI (the fraction of missing information, reported as Rubin's asymptotic $\lambda = RIV/(1 + RIV)$), and df; the rotated families on the analytic path additionally carry a method string recording the gauge alignment used ("gauge_invariant" for communalities and "signed_permutation_approx" for rotated loadings and, for oblique rotations, factor correlations and structure coefficients). fit_indices_descriptive, on the bootstrap path, pools every fit index the bootstrap replicates carry, so the structural constants among them (df, df_null) appear with a standard error of 0. The RMSEA confidence bounds are not among them: the replicate fits run without confidence intervals, so no per-replicate value exists to pool.

mi_fit On the se = "sandwich" (MI2S) path only: the single efa_fit() fit on the pooled correlation matrix $\bar{r}$ and pooled asymptotic covariance $\bar{\Gamma}$. Its orig_R is $\bar{r}$ and its Gamma is $\bar{\Gamma}$; the pooled SE and CI are taken from it, as are the pooled fit_indices (put into the common order above and extended with pool_method, while mi_fit keeps efa_fit()'s own layout). MI is NULL on this path because the imputation uncertainty is carried by $\bar{\Gamma}$ rather than by per-parameter Rubin pooling.

mi_diagnostics Diagnostics for the pooled model fit, NULL on the se = "sandwich" (MI2S) path, where there is one fit and no D2 pool. m is the number of imputations that entered the pool. D2_F, D2_df1, D2_df2, D2_chi_asymptotic, ARIV and FMI describe the D2 pool of the model chi-square (the average relative increase in variance and the fraction of missing information it implies), and chi_bar_naive is the plain mean of the per-imputation statistics for comparison; the *_null entries are the same quantities for the independence baseline. D2_F is the rule's raw statistic and is reported unfloored, so it is negative whenever the between-imputation variability of the component statistics exceeds the pooled discrepancy – a diagnostic of the pool rather than a fit statistic. The reported fit is not affected: the pooled chi-square is floored at zero and its p-value is 1 in that case. chi_cfi and chi_null_cfi are the pooled model and baseline **chi-squares** on the common $N - 1$ noncentrality scale. chi_cfi is the statistic the reported RMSEA is formed from; the pair also gives a reference CFI formed the conventional way, 1 - (chi_cfi - df) / (chi_null_cfi - df_null) (and analogously for TLI) – a different quantity from the reported CFI/TLI, which average the per-imputation indices.

mi_admissibility Admissibility and convergence of the component fits, kept on the pooled object so a saved solution carries the record independently of fits: m (the number of imputations that entered the pool), heywood_imputations (the indices of the fits with at least one Heywood case – an improper solution where a variable's communality is at or above 1, or its uniqueness is fixed at the estimation boundary), n_heywood_items (the number of flagged variables per imputation), nonconverged (the indices whose extraction reported a non-zero convergence code), and iter (the iterations each extraction used). Averaging aligned solutions pulls boundary communalities back inside the admissible range, so a pooled matrix with no Heywood case can still rest on component fits that had them; summary() reports the pooled count together with these.

fits The list of m component efa_fit() fits, in the order of data_list, kept for per-imputation diagnostics. On the MI2S path these are the per-imputation fits whose inputs were pooled, not the pooled fit itself (which is mi_fit).

alignment Metadata from aligning the rotated solutions, NULL when no rotation was requested or on the MI2S path (one fit, one gauge). Under target_method = "first_target": the method used, the target it aligned to, the per-imputation target_rotations, the indices

of any point_rotation_failures, and whether every inner alignment converged. Under target_method = "consensus" it is the full efa_procrustes()-based GPA record: the converged target, the aligned_loadings and aligned_phi, the iteration history, convergence flags, and the multi-start summary.

settings The component fits' efa_fit() settings with the pooling settings added: pooled (always TRUE), pooled_N and N (the mean N across imputations), n_imputations, component_se (the se the component fits used), target_method, align_unrotated, fit_pool_method, p, ci and rmsea_ci_level. se records what was actually pooled, so it is "none" when pooled standard errors could not be produced although the component fits computed them (component_se keeps the request).

Conditions

Errors and warnings raised by efa_mi() are classed, with an efa_pooled_ prefix (efa_consensus_ for the consensus target) – except the dots validation shared with efa_fit(), which signals efa_flat_knob_in_dots or efa_renamed_arg – so they can be caught programmatically. The message shown explains what went wrong and, where relevant, how to fix it.

Author(s)

Andreas Soteriades, Markus Steiner

References

Anderson, T. W., & Rubin, H. (1956). Statistical inference in factor analysis. In *Proceedings of the Third Berkeley Symposium on Mathematical Statistics and Probability* (Vol. 5, pp. 111-150). University of California Press.

Archer, C. O., & Jennrich, R. I. (1973). Standard errors for rotated factor loadings. *Psychometrika*, 38(4), 581-592.

Barnard, J., & Rubin, D. B. (1999). Small-sample degrees of freedom with multiple imputation. *Biometrika*, 86(4), 948-955.

Bentler, P. M. (1990). Comparative fit indexes in structural models. *Psychological Bulletin*, 107(2), 238-246.

Chung, S., & Cai, L. (2019). Alternative multiple imputation inference for categorical structural equation modeling. *Multivariate Behavioral Research*, 54(3), 323-337.

Gower, J. C. (1975). Generalized Procrustes analysis. *Psychometrika*, 40(1), 33-51.

Jennrich, R. I. (1973). Standard errors for obliquely rotated factor loadings. *Psychometrika*, 38(4), 593-604.

Jennrich, R. I. (1974). Simplified formulae for standard errors in maximum-likelihood factor analysis. *British Journal of Mathematical and Statistical Psychology*, 27(1), 122-131.

Lawley, D. N., & Maxwell, A. E. (1971). *Factor analysis as a statistical method* (2nd ed.). Butterworths.

Li, K. H., Meng, X.-L., Raghunathan, T. E., & Rubin, D. B. (1991). Significance levels from repeated p-values with multiply-imputed data. *Statistica Sinica*, 1(1), 65-92.

Lorenzo-Seva, U., & Van Ginkel, J. R. (2016). Multiple imputation of missing values in exploratory factor analysis of multidimensional scales. *Anales de Psicologia*, 32(2), 596-608.

Rubin, D. B. (1987). *Multiple imputation for nonresponse in surveys*. Wiley.

Schoenemann, P. H. (1966). A generalized solution of the orthogonal Procrustes problem. *Psychometrika*, 31(1), 1-10.

Sriutaisuk, S., Liu, Y., Chung, S., Kim, H., & Gu, F. (2025). Evaluating imputation-based fit statistics in structural equation modeling with ordinal data: The MI2S approach. *Educational and Psychological Measurement*, 85(1), 82-113.

Tucker, L. R., & Lewis, C. (1973). A reliability coefficient for maximum likelihood factor analysis. *Psychometrika*, 38(1), 1-10.

van Ginkel, J. R., & Kroonenberg, P. M. (2014). Using generalized Procrustes analysis for multiple imputation in principal component analysis. *Journal of Classification*, 31(2), 242-269.

Zhang, G., & Preacher, K. J. (2015). Factor rotation and standard errors in exploratory factor analysis. *Journal of Educational and Behavioral Statistics*, 40(6), 579-603.

Zhang, G., Preacher, K. J., & Jennrich, R. I. (2012). The infinitesimal jackknife with exploratory factor analysis. *Psychometrika*, 77(4), 634-648.

See Also

Other factor analysis: `efa_average()`, `efa_fit()`, `efa_group()`, `plot.efa_group()`, `print.efa_group()`

Examples

```
# create a list of three datasets, mimicking a list you would obtain from
# e.g. mice.
dat_list <- lapply(1:3, function(x) GRiPS_raw[sample(1:nrow(GRiPS_raw), replace = TRUE),])
mod <- efa_mi(dat_list, n_factors = 1, estimator = "ML")
mod


# add computation of standard errors and CIs
mod <- efa_mi(dat_list, n_factors = 1, estimator = "ML", se = "np-boot")
mod
```

`efa_nest` *Next eigenvalue sufficiency test (NEST)*

Description

NEST uses many synthetic datasets to generate reference eigenvalues against which to compare the empirical eigenvalues. This is similar to parallel analysis, but other than parallel analysis, NEST does not just rely on synthetic eigenvalues based on an identity matrix as null model. It was introduced by Achim (2017), see also Brandenburg and Papenberg (2024) and Caron (2025) for further simulation studies including NEST.

Usage

```
efa_nest(
  x,
  N = NA,
  alpha = 0.05,
  use = c("pairwise.complete.obs", "all.obs", "complete.obs", "everything",
    "na.or.complete"),
  cor_method = c("pearson", "spearman", "kendall", "poly", "tetra"),
  n_datasets = 1000,
  estimate_control = NULL,
  ...
)
```

Arguments

x	data.frame or matrix. Dataframe or matrix of raw data or matrix with correlations.
N	numeric. The number of observations. Only needed if x is a correlation matrix. Must be larger than the number of variables.
alpha	numeric. The alpha level to use (i.e., 1-alpha percentile of eigenvalues is used for reference values).
use	character. Passed to stats::cor() if raw data is given as input. Default is "pairwise.complete.obs".
cor_method	character. One of "pearson", "spearman", or "kendall", passed to stats::cor(). "poly" and "tetra" are not supported because NEST compares the data against simulated continuous reference data. Default is "pearson".
n_datasets	numeric. The number of datasets to simulate. Default is 1000.
estimate_control	an estimate_control() object with the estimation settings for the efa_fit() reference-model fits. NULL (default) uses the efa_fit() defaults. The reference models are unrotated, so no rotation settings apply.
...	Additional arguments passed to efa_fit(). For example, estimator, to change the estimator (default is "PAF"). PAF is more robust, but it will take longer compared to the other estimators available ("ML" and "ULS"). The estimation tuning knobs are not passed here; they live in estimate_control, and the standard-error arguments (se, b_boot, ci, seed) are not accepted because the reference-model fits are internal steps.

Details

NEST compares the first empirical eigenvalue against the first eigenvalues of n_dataset synthetic datasets based on a null model (i.e., with uncorrelated variables; same as in parallel analysis, see efa_parallel()). The following eigenvalues are compared against synthetic datasets based on an EFA-model with one fewer factors than the position of the respective empirical eigenvalue. E.g, the second empirical eigenvalue is compared against synthetic data based on a one-factor model. In each comparison the k-th empirical eigenvalue is tested against the k-th largest eigenvalue of

the synthetic datasets. The alpha-level defines against which percentile of the synthetic eigenvalue distribution to compare the empirical eigenvalues against, i.e., an alpha of .05 (the default) uses the 95th percentile as reference value.

The number of factors tested is capped at $\lfloor 0.8 \times p \rfloor$ (with p the number of variables; Achim, 2017) and additionally limited so that the $(k-1)$-factor reference model used at each step stays over-identified. If no empirical eigenvalue falls at or below its reference within this range, every tested factor is accepted and this capped number is returned.

Because each reference model carries the factors already retained, NEST does not lose accuracy for the strongly correlated factor structures where parallel analysis tends to under-extract, and it was among the more accurate criteria in the simulation studies of Brandenburg and Papenberg (2024) and Caron (2025). The price is runtime: a fresh set of n_datasets reference datasets is drawn and eigen-decomposed at every candidate factor count, which makes NEST one of the slowest criteria available here.

The reference models are fitted without inequality constraints. A Heywood case in one of them leaves no unique variance to simulate the reference data from, so NEST aborts rather than continuing from an inadmissible reference.

The reference eigenvalues are obtained from simulated data, so the suggested number of factors varies slightly from run to run. Call base::set.seed() beforehand to make a run reproducible.

For details on the method, including simulation studies, see Achim (2017), Brandenburg and Papenberg (2024), and Caron (2025).

The efa_nest function can also be called together with other factor retention criteria in the efa_retain() function.

Value

An object of class efa_retention (see print.efa_retention() and plot.efa_retention() for the print and plot methods). Its main fields are:

n_factors	A named numeric vector ("NEST") with the suggested number of factors according to the NEST procedure.
results	A list with a single record holding the empirical eigenvalues and the reference eigenvalues. Only the positions the search actually tested carry a reference value; beyond the position at which it stopped the series is NA.
settings	A list of control settings used.

Source

Achim, A. (2017). Testing the number of required dimensions in exploratory factor analysis. The Quantitative Methods for Psychology, 13(1), 64–74. https://doi.org/10.20982/tqmp.13.1.p064

Brandenburg, N., & Papenberg, M. (2024). Reassessment of innovative methods to determine the number of factors: A simulation-based comparison of exploratory graph analysis and Next Eigenvalue Sufficiency Test. Psychological Methods, 29(1), 21–47. https://doi.org/10.1037/met0000527

Caron, P.-O. (2025). A Comparison of the Next Eigenvalue Sufficiency Test to Other Stopping Rules for the Number of Factors in Factor Analysis. Educational and Psychological Measurement, Online-first publication. https://doi.org/10.1177/00131644241308528

See Also

efa_retain() as a wrapper function for this and the other factor retention criteria.

Other factor retention criteria: efa_cd(), efa_ekc(), efa_hull(), efa_kgc(), efa_map(), efa_parallel(), efa_retain(), efa_scree(), efa_smt()

Examples

```
# with correlation matrix
efa_nest(test_models$baseline$cormat, N = 500)

# with raw data
efa_nest(GRiPS_raw)
```

---

efa_parallel *Parallel analysis*

Description

Various methods for performing parallel analysis. This function uses future_lapply() for which a parallel processing plan can be selected. To do so, register a plan with future::plan(), for example future::plan(future::multisession, workers = 2); see examples.

Usage

```
efa_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", "poly", "tetra"),
  decision_rule = c("means", "percentile", "crawford"),
  n_factors = 1,
  estimate_control = NULL,
  ...
)
```

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. Must be larger than the number of variables.

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. Must be at least 1. 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. Default is `c("PCA", "SMC", "EFA")`, i.e. all three, which costs roughly six times a single non-EFA type: `"EFA"` fits an EFA to every simulated dataset and dominates that total. Pass a single type if the run is time-critical.

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

cor_method — character. One of `"pearson"`, `"spearman"`, or `"kendall"`, passed to `stats::cor()`. `"poly"` and `"tetra"` are not supported because PARALLEL compares the data against simulated continuous reference data. 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). All three rules retain the factors up to the first observed eigenvalue that fails to exceed its reference value; an eigenvalue further down the series that rises above its own reference again therefore adds no factor. Because the average simulated eigenvalue is a lower reference than the percentile, `"means"` tends to retain more factors than the more conservative `"percentile"` rule (Glorfeld, 1995).

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

estimate_control — an `estimate_control()` object with the estimation settings for the `efa_fit()` fits (of both the real and the simulated data) when `"EFA"` is included in `eigen_type`. `NULL` (default) uses the `efa_fit()` defaults. The fits are unrotated, so no rotation settings apply.

... — Additional arguments passed to `efa_fit()`. For example, `estimator`, to change the estimator (default is "PAF"). PAF is more robust, but it will take longer compared to the other estimators available ("ML" and "ULS"). The estimation tuning knobs are not passed here; they live in `estimate_control`, and the standard-error arguments (`se`, `b_boot`, `ci`, `seed`) are not accepted because the fits are internal steps that keep only their eigenvalues.

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 (`efa_kgc()`).

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 reference eigenvalues are obtained from simulated data, so the suggested number of factors varies slightly from run to run. Call `base::set.seed()` beforehand to make a run reproducible; the result is then also independent of the parallel plan set via `future::plan()`, so it can be reproduced on a machine with a different number of cores. For `"PCA"` and `"SMC"` the simulation is drawn in independently seeded blocks; a block that fails – which happens when a simulated correlation matrix is singular, so that no eigenvalues can be taken from it – is redrawn on its own, leaving the blocks that succeeded with the draws they already made. The `"EFA"` series instead redraws the single dataset that could not be fitted; if that dataset still cannot be fitted, the call stops with an error.

When both `"PCA"` and `"SMC"` are requested, the two are read off the *same* simulated datasets rather than from two independent simulations: they differ only in the diagonal substituted into the simulated correlation matrix, so one set of draws serves both and the two reference series are paired dataset by dataset. A draw that cannot be used for the SMC series – a simulated matrix with no inverse, and hence no squared multiple correlations – is discarded for the `"PCA"` series as well, so that the pairing stays exact. `"EFA"` fits a model to each simulated dataset and draws its own.

The `efa_parallel` function can also be called together with other factor retention criteria in the `efa_retain()` function.

Value

An object of class `efa_retention` (see `print.efa_retention()` and `plot.efa_retention()` for the print and plot methods). Its main fields are:

`n_factors`	A named numeric vector with the suggested number of factors for each requested eigenvalue type (`"PCA"`, `"SMC"`, and/or `"EFA"`). These are `NA` when no real data are supplied (i.e. only `N` and `n_vars` are given). When every observed eigenvalue exceeds its reference value (no crossing is found), all `n_vars` components are retained and a warning is issued.
`results`	A list with one record per eigenvalue type, each holding the observed eigenvalues (when real data were supplied) and the simulated reference values (means and percentiles) used for printing and plotting.
`settings`	A list of the settings used.

Source

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

Crawford, A. V., Green, S. B., Levy, R., Lo, W. J., Scott, L., Svetina, D., & Thompson, M. S. (2010). Evaluation of parallel analysis methods for determining the number of factors. Educational and Psychological Measurement, 70(6), 885-901.

Glorfeld, L. W. (1995). An improvement on Horn's parallel analysis methodology for selecting the correct number of factors to retain. Educational and Psychological Measurement, 55(3), 377-393.

Horn, J. L. (1965). A rationale and test for the number of factors in factor analysis. Psychometrika, 30(2), 179–185. https://doi.org/10.1007/BF02289447

See Also

`efa_retain()` as a wrapper function for this and the other factor retention criteria.

Other factor retention criteria: `efa_cd()`, `efa_ekc()`, `efa_hull()`, `efa_kgc()`, `efa_map()`, `efa_nest()`, `efa_retain()`, `efa_scree()`, `efa_smt()`

Examples

```
# example without real data
pa_unreal <- efa_parallel(N = 500, n_vars = 10, n_datasets = 100)

# example with correlation matrix with all eigen_types and PAF estimation
pa_paf <- efa_parallel(test_models$case_11b$cormat, N = 500, n_datasets = 100)

# example with correlation matrix with all eigen_types and ML estimation
# this will be faster than the above with PAF)
pa_ml <- efa_parallel(test_models$case_11b$cormat, N = 500, estimator = "ML",
                      n_datasets = 100)


## Not run:
# for parallel computation. future::plan() returns the plan it replaces, so
# on.exit() puts the session back as it was -- also if the call fails.
pa_faster <- local({
  old_plan <- future::plan(future::multisession, workers = 2)
  on.exit(future::plan(old_plan), add = TRUE)
  efa_parallel(test_models$case_11b$cormat, N = 500)
})

## End(Not run)
```

EFA_POOLED *Exploratory factor analysis on multiple data imputations*

Description

[Superseded]

`EFA_POOLED()` has been superseded by `efa_mi()`, which is the recommended interface going forward. It remains available and unchanged so existing code keeps working.

Usage

```
EFA_POOLED(
  data_list,
  p = 0.05,
  target_method = c("first_target", "consensus"),
  align_unrotated = c("signed_tucker_congruence", "none", "procrustes"),
  fit_pool_method = c("D2"),
  consensus_args = list(),
  procrustes_args = list(),
  rmsea_ci_level = 0.9,
  rmsr_upper = TRUE,
  ...
)
```

Arguments

`data_list`	A list of length m, where m is the number of imputations. Each list element is a data frame or matrix of raw data, or a correlation matrix. See argument `x` in `efa_fit()`. A `mids` object from **mice** must be converted first, with `mice::complete(x, "all")`.
`p`	Numeric in $(0, 1)$. One minus the confidence level for the pooled confidence intervals, whichever `se` method produced them (`"information"`, `"np-boot"`, or `"sandwich"`). For example, `p = .05` gives 95% intervals.
`target_method`	Character. How rotated solutions are aligned across imputations before pooling: `"first_target"` (the default) aligns every imputation to the first imputation's rotated solution, while `"consensus"` refines a centroid target by Generalized Procrustes Analysis, started from the medoid imputation so that the pooled rotated solution does not depend on the order of `data_list` (orthogonal rotations only). See *Aligning solutions across imputations* in Details.
`align_unrotated`	Character. How unrotated loadings are aligned before pooling: `"signed_tucker_congruence"` (the default; sign/permutation via Tucker congruence, anchored on the medoid imputation and returned in the extraction's canonical gauge), `"procrustes"` (orthogonal Procrustes to the first imputation), or `"none"`. See *Aligning solutions across imputations* in Details.
`fit_pool_method`	Character. Only `"D2"` is implemented for pooling chi-square-type fit. If no chi-square is available, only residual-based fit and descriptive quantities are returned. See *Pooling the model chi-square and fit indices* in Details.
`consensus_args`	List of additional arguments controlling the GPA-consensus iteration when `target_method = "consensus"`. Recognised tuning parameters include the convergence tolerances `tol` and `loss_tol`, the iteration bounds `min_iter` and `max_iter`, the target-update damping `alpha`, the multi-start controls `multi_start` and `starts`, and `start`, which overrides the medoid imputation the iteration is otherwise started from.

procrustes_args
: List of efa_procrustes() algorithm controls for fixed-target alignment, for example oblique_maxit or oblique_random_starts. The loadings A, the alignment Target, the rotation family, and the cross-product S are derived from the imputations and cannot be set here.

rmsea_ci_level
: Numeric. Confidence level for the RMSEA CI.

rmsr_upper
: **[Deprecated]** Deprecated and ignored. efa_mi() now always computes RMSR the same way, from the unique off-diagonal residuals; SRMR is reported alongside it. Supplying it to efa_mi() signals a deprecation warning; the superseded EFA_POOLED() accepts it silently.

...
: Additional arguments passed to efa_fit() (e.g. estimator, rotation, se, n_factors, N). These select the estimator, rotation, standard-error method, and fit indices used for every imputation; see efa_fit() for the available options, their properties, and which combinations are valid. Two of them shape the pooled object rather than a single fit: seed sets the random state once for the whole efa_mi() call – every component bootstrap and every random-start rotation draws from it, so a seeded call is reproducible as a whole, and the caller's random stream is restored afterwards – and b_boot sets the number of bootstrap replicates drawn per imputation under se = "np-boot", which is what the pooled within-imputation variances are estimated from and is recorded in settings$b_boot. The estimate_control() and rotate_control() objects are accepted through ... as well, although they are not declared formals: pass them as estimate_control = / rotate_control = exactly as you would to efa_fit().

Value

The value of efa_mi(), normally a list of class c("efa_mi", "EFA_POOLED", "efa", "EFA"); see there for the components.

See Also

efa_mi()

---

efa_power *Power analysis for exploratory factor analysis*

Description

Analyses power for exploratory factor analysis, in one of two modes chosen with mode.

mode = "rmsea" (the default) gives the analytic power of the root mean square error of approximation (RMSEA) tests of close and not-close fit (MacCallum, Browne, & Sugawara, 1996). Give a sample size to get the power of the test, or give a target power to get the sample size needed to reach it.

mode = "simulation" runs a Monte-Carlo study: it draws n_datasets samples from a known population (via efa_simulate()), analyses each one, and reports how well the analysis recovers that population. See *Details* for what is reported.

Here the number of variables is p and the number of factors is k (elsewhere in the package: n_vars and n_factors).

Usage

```
efa_power(
  mode = c("rmsea", "simulation"),
  type = c("close", "notclose"),
  eps0 = NULL,
  eps1 = NULL,
  N = NULL,
  p = NULL,
  k = NULL,
  df = NULL,
  alpha = 0.05,
  power = NULL,
  group = 1,
  Lambda = NULL,
  Phi = NULL,
  Psi = NULL,
  R = NULL,
  n_datasets = 500,
  criteria = c("EKC", "MAP"),
  estimator = "PAF",
  rotation = NULL,
  recovery_threshold = 0.95,
  model_error = c("TKL", "CB", "WB", "none"),
  target_rmsea = NULL,
  target_cfi = NULL,
  seed = NULL
)
```

Arguments

mode	character. The kind of power analysis: "rmsea" (the default; analytic RMSEA power) or "simulation" (Monte-Carlo hit-rate and structure recovery). type, eps0, eps1, df, alpha, power, and group apply to RMSEA mode only; the arguments marked *Simulation mode* below apply to the other; N, p, and k are used in both.
type	character. The RMSEA test: "close" (test of close fit) or "notclose" (test of not-close fit). See *Details*.
eps0	numeric. The null-hypothesis RMSEA. Default is 0.05.
eps1	numeric. The alternative-hypothesis RMSEA (the true RMSEA power is evaluated at). Default is 0.08 for type = "close" and 0.01 for type = "notclose".
N	numeric. In "rmsea" mode, the total sample size across groups (the plain sample size when group is 1): give N to compute power, or leave it NULL to solve for the required N at a target power. In "simulation" mode N is required and is the size of each drawn sample, with no sample size solved for and no group division.

p — numeric. The number of observed variables. In "rmsea" mode, used with k to derive df when df is not given directly. In "simulation" mode it is read off the population, so leave it unset (or matching nrow(Lambda) / nrow(R)).

k — numeric. The number of factors. In "rmsea" mode, used with p to derive df when df is not given directly. In "simulation" mode it is the true number of factors: it is required with an R population and must be left unset (or match ncol(Lambda)) with a factor-model population.

df — numeric. The model degrees of freedom. Either supply df directly or supply both p and k, from which df = ((p - k)^2 - (p + k)) / 2. Must be positive.

alpha — numeric. The significance level. Default is 0.05.

power — numeric. The target power. Give power (or leave both power and N NULL, defaulting to 0.80) to solve for the required N; leave it NULL while giving N to compute power. Exactly one of N and power is solved for.

group — numeric. The number of groups. Default is 1. N is the total across all group groups, not the size of each one, and a solved N is a multiple of group. See *Details*.

Lambda — matrix. Simulation mode. A p by k_true population loading matrix. Supply this (optionally with Phi/Psi) to build a factor-model population; structure recovery is available only with this form. Passed to efa_simulate().

Phi — matrix. Simulation mode. The k_true by k_true population factor intercorrelations. Only used with Lambda; defaults to orthogonal factors. When rotation is unset, an oblique Phi selects a "promax" recovery fit and an orthogonal one a "varimax" fit.

Psi — numeric or matrix. Simulation mode. The population unique variances (a length-p vector or a p by p matrix). Only used with Lambda. Passed to efa_simulate().

R — matrix. Simulation mode. A p by p population correlation matrix to draw from directly, instead of a factor model. Structure recovery is not available for this form (there are no population loadings to recover), and k is required.

n_datasets — numeric. Simulation mode. The number of samples to draw and analyse. Default is 500.

criteria — character. Simulation mode. The factor-retention criteria to evaluate the hit-rate for, any of "CD", "EKC", "HULL", "KGC", "MAP", "NEST", "PARALLEL", and "SMT" (see efa_retain()). Default is c("EKC", "MAP"). Criteria that simulate internally ("CD", "HULL", "NEST", "PARALLEL") make each run substantially slower.

estimator — character. Simulation mode. The estimator ("PAF", "ML", or "ULS") used for the recovery fit and the retention criteria. Default is "PAF".

rotation — character. Simulation mode. The rotation for the recovery fit, passed to efa_fit(). Default is NULL, which matches the population: "varimax" for orthogonal factors and "promax" for oblique ones (a single factor is left unrotated). Recovery aligns the fitted loadings to the population pattern by permutation and sign only. A rotation that does not seek that structure – for example "none" with more than one factor – will understate recovery, so keep the default (or another structure-seeking rotation) for a meaningful recovery rate.

recovery_threshold
: numeric. Simulation mode. The matched-factor Tucker congruence a replicate must reach to count as recovered. Default is 0.95: Lorenzo-Seva and ten Berge (2006) treat congruence at or above this level as indicating the same factor.

model_error
: character. Simulation mode. The efa_simulate() method that perturbs the population with model error: "TKL" (Tucker-Koopman-Linn, the default here), "CB" (Cudeck-Browne), "WB" (Wu-Browne), or "none" for an exact population. It only takes effect when a target is supplied (target_rmsea and/or target_cfi), and only for a factor-model population; without a target the population stays exact whatever the method. "TKL" adds minor common factors, giving a realistically imperfect population but lowering both the hit-rate and structure recovery; "CB" and "WB" target the RMSEA only. "CB" keeps the population loadings as the exact minimizer of the perturbed population, so recovery stays close to perfect; "WB"'s loadings are not the minimizer, so they carry no such guarantee. Note that efa_simulate() itself defaults to "CB": the same target_rmsea passed to both functions gives an easier population there, unless model_error is also set explicitly here.

target_rmsea
: numeric. Simulation mode. The population RMSEA the model should have relative to the perturbed population, activating model error. Default is NULL. Passed to efa_simulate().

target_cfi
: numeric. Simulation mode. The population CFI target (only with model_error = "TKL"). Default is NULL. Passed to efa_simulate().

seed
: numeric. Simulation mode. Optional seed making the draws and analysis reproducible and worker-count independent; the caller's random-number stream is restored afterwards. Default is NULL.

Value

An object of class efa_power. For mode = "rmsea", a list containing:

power
: The power of the test at N (the achieved power, which for a solved sample size is at least the target).

N
: The total sample size across groups: the supplied N, or the solved required sample size (a multiple of group).

N_per_group
: The per-group sample size N / group, equal to N when group is 1. A whole number for a solved N; for a supplied N that is not a multiple of group it is the fraction that the noncentrality uses.

crit
: The critical chi-square value the fit statistic is compared against.

ncp
: The noncentrality parameters under the null (H0, from eps0) and the alternative (H1, from eps1).

solve_for
: "power" or "N", recording which quantity was solved for.

settings
: A list of the inputs: mode, type, eps0, eps1, df, p, k, alpha, group, and the target power (the value solved to when solve_for is "N", otherwise NULL).

For mode = "simulation", a list containing:

`hit_rate` A named numeric vector of the retention hit-rate per criterion (and, where a criterion has several variants, per variant); `NA` for a criterion that returned no suggestion on any replicate.

`hits` A data frame with one row per criterion (`criterion`) giving the number of replicates it returned a definite suggestion on (`n_valid`), the number of those that matched `k_true` (`hits`), and the `hit_rate` (`hits / n_valid`).

`recovery` For a factor-model population, a list with the structure-recovery rates (`min_rate`, `mean_rate`), the `threshold`, and the number of usable fits (`n_valid`); `NULL` for an `R` population. Rates are over every replicate whose fit returned loadings, including non-converged or Heywood solutions (their rates are reported separately in `convergence`).

`convergence` A list with the number of datasets (`n_datasets`), the number of fits that completed (`n_fit_ok`), how many of those converged (`n_converged`) and how many produced a Heywood case (`n_heywood`), and the corresponding rates: `fit_rate` (fits completed, over all datasets) and `convergence_rate` / `heywood_rate` (converged / Heywood, over the completed fits).

`replicates` The raw per-replicate values: the suggested factor counts (`n_hat`), the matched congruences (`rec_min`, `rec_mean`), the `converged`, `heywood`, and `fit_ok` flags, and `fit_error`, the message of the fit that did not complete (`NA` where it did).

`k_true` The true number of factors.

`model_error` The `efa_simulate()` model-error record, or `NULL`.

`settings` A list of the simulation inputs.

RMSEA mode

Power rises with a larger sample, a larger model (more degrees of freedom), and a bigger gap between the null and alternative RMSEA (MacCallum, Browne, & Sugawara, 1996).

Two tests are supported, chosen with `type` (never by the order of `eps0` and `eps1`):

`"close"` Tests *close fit* (MacCallum et al., 1996). The null hypothesis is that the fit is close (RMSEA $\leq$ `eps0`; conventionally 0.05). Power is the chance of detecting a worse alternative (`eps1`; conventionally 0.08, so `eps0` < `eps1`), in the upper tail.

`"notclose"` Tests *not-close fit*. The null hypothesis is that the fit is not close (RMSEA $\geq$ `eps0`). Power is the chance of detecting a better alternative (`eps1`; conventionally 0.01, so `eps0` > `eps1`), in the lower tail.

When `eps0` and `eps1` are in the wrong order for the chosen `type`, a message is shown but the requested test still runs. Equal `eps0` and `eps1` leave nothing to detect and are an error.

Power always increases with `N`, so the required sample size (the smallest `N` reaching `power`) is found by bisection. `N` is the **total** sample size across groups: with `group` > 1 the power calculation divides by `group` (the `1 / group` factor), so spreading a fixed total over more groups gives less power. The matching per-group sample size, `N / group`, is returned as `N_per_group`.

The `1 / group` factor makes all `group` groups the same size, so a required total is rounded up to the next multiple of `group`. A solved `N_per_group` is thus a whole number of persons, and the reported `power` is the power at a total that a study can collect. With `group = 2` and `df = 102`, for example, the required total is 260, or 130 per group. Bisection on the total alone gives 259, which asks for 129.5 persons in each group.

Simulation mode

The population is passed to `efa_simulate()`, which draws `n_datasets` samples of size `N` from it. The population's true number of factors, `k_true`, is `ncol(Lambda)` for a factor-model population, or `k` for a bare `R`. By default the population fits the factor model exactly, which overstates how well the criteria and the fit recover its structure; setting a misfit target makes the population more realistic (MacCallum, 2003).

Each replicate is analysed three ways:

Hit-rate The share of replicates where a criterion's suggested factor count (from `criteria`) matches `k_true`. A replicate where the criterion errored or gave no answer is left out of this count – it does not count as a miss.

Structure recovery (factor-model populations only) The `k_true`-factor model is fitted with `efa_fit()`, its loadings are matched to the population loadings, and the matched-factor Tucker congruences (Lorenzo-Seva & ten Berge, 2006) are compared with `recovery_threshold`. A replicate succeeds when its smallest (`min`) or average (`mean`) matched congruence reaches the threshold.

Convergence Among the replicates whose fit completed, the share that converged and the share that produced a Heywood case.

A replicate whose fit fails completely is not counted in any of the three measures above. If any fit fails, a warning reports how many failed and the cause of the first failure.

Replicates are analysed in parallel with **future.apply**; choose a parallel plan with `future::plan()`. Each replicate uses its own reproducible random-number stream, so with a fixed `seed` the result does not depend on the number of workers, and the caller's random-number state is left unchanged.

References

MacCallum, R. C., Browne, M. W., & Sugawara, H. M. (1996). Power analysis and determination of sample size for covariance structure modeling. *Psychological Methods, 1*(2), 130-149. doi:10.1037/1082989X.1.2.130

MacCallum, R. C. (2003). 2001 Presidential Address: Working with imperfect models. *Multivariate Behavioral Research, 38*(1), 113-139. doi:10.1207/S15327906MBR3801_5

Lorenzo-Seva, U., & ten Berge, J. M. F. (2006). Tucker's congruence coefficient as a meaningful index of factor similarity. *Methodology, 2*(2), 57-64. doi:10.1027/16142241.2.2.57

See Also

`efa_simulate()` draws the replicate datasets used in simulation mode. `efa_retain()` implements the retention criteria whose hit-rates simulation mode reports.

Other power analysis: `plot.efa_power()`, `print.efa_power()`

Examples

```
# Power of the test of close fit at N = 200 for a 100-df model
efa_power(df = 100, N = 200)

# Deriving df from the model dimensions instead of giving it directly
```

```
efa_power(p = 20, k = 3, N = 200)

# Required total sample size for 80% power
efa_power(df = 100, power = 0.80)

# Test of not-close fit
efa_power(df = 100, N = 200, type = "notclose")


# Simulation mode: retention hit-rate and structure recovery for a known
# three-factor population at N = 300
efa_power("simulation", Lambda = population_models$loadings$baseline,
          Phi = population_models$phis_3$moderate, N = 300,
          n_datasets = 50, criteria = c("EKC", "MAP"), seed = 42)
```

`efa_procrustes` *Rotate a loading matrix to a target using Procrustes alignment*

Description

`efa_procrustes()` aligns one loading matrix to a target loading matrix with the same dimensions. It is used internally by `efa_mi()`, but can also be used directly when factor columns must be brought into a common orientation before averaging or comparing solutions.

Usage

```
efa_procrustes(
  A,
  Target,
  rotation = c("orthogonal", "oblique"),
  S = NULL,
  T_init = NULL,
  oblique_eps = 1e-05,
  oblique_maxit = 1000,
  oblique_max_line_search = 10,
  oblique_step0 = 1,
  oblique_normalize = FALSE,
  oblique_random_starts = 0,
  oblique_screen_keep = 2,
  oblique_triage_maxit = 25,
  oblique_triage_improve_tol = 0
)
```

Arguments

`A`	Numeric loading matrix to be aligned.

`Target` Numeric target matrix with the same dimensions as `A`.

`rotation` Character string, either `"orthogonal"` or `"oblique"`.

`S` Optional `k x k` cross-product matrix `crossprod(A)`, kept for compatibility. It enters both the oblique criterion and its gradient, so any other matrix would minimize a different criterion: where `S` is used it is checked against `crossprod(A)` and must agree with it up to a relative tolerance of `1e-8`. That check forms `crossprod(A)` itself, so passing `S` no longer avoids any work: omitting it gives the same result for slightly less. `S` is used, and therefore checked, only on the oblique path with more than one factor and `oblique_normalize = FALSE`; if Kaiser normalization is requested, the cross-product must be recomputed on the normalized matrix and `S` is ignored.

`T_init` Optional `k x k` starting transformation matrix for the oblique solver. Its columns are normalized internally, and the normalized matrix must be well enough conditioned to define a proper factor correlation matrix: its smallest singular value must be at least `1e-4`, the same floor the solver applies to every candidate it evaluates. If `NULL` (the default), the oblique solver is warm-started from the closed-form orthogonal Procrustes solution.

`oblique_eps` Positive convergence tolerance for the projected-gradient norm in the oblique solver.

`oblique_maxit` Non-negative integer. Maximum number of projected-gradient updates in the full oblique solver.

`oblique_max_line_search`
Non-negative integer. Maximum number of step-halving attempts after the initial line-search step.

`oblique_step0` Positive initial step size for the oblique solver.

`oblique_normalize`
Logical; if `TRUE`, apply Kaiser row normalization to the loadings (only) in the oblique solver and back-transform the aligned loadings afterwards, leaving `Target` unnormalized (as in `GPArotation::targetQ(normalize = TRUE)`).

`oblique_random_starts`
Non-negative integer. Number of additional random starts used by the oblique solver.

`oblique_screen_keep`
Non-negative integer. Number of random starts retained after cheap objective screening and sent to triage optimization.

`oblique_triage_maxit`
Non-negative integer. Number of short optimization iterations used in the triage stage.

`oblique_triage_improve_tol`
Non-negative scalar. Relative improvement required for a triaged start to be promoted to full optimization.

Details

For `rotation = "orthogonal"`, the function solves the closed-form orthogonal Procrustes problem

$$\min_T \frac{1}{2}\|AT - B\|_F^2 \quad \text{subject to} \quad T'T = I,$$

where A is the loading matrix and B is Target.

For rotation = "oblique", the function calls the compiled .oblique_procrustes() optimizer. The oblique convention is the same as in GPArotation::targetQ():

$$L = AT^{-T}, \qquad \Phi = T'T, \qquad diag(\Phi) = 1.$$

By default the oblique solver is warm-started from the closed-form orthogonal Procrustes solution, which resolves the factor permutation and sign indeterminacy and avoids the poor local minima an identity start can fall into. Supply T_init to override this start. Random starts are only used for oblique alignment. For one-factor models, oblique and orthogonal alignment are equivalent, so the function uses the stable one-factor orthogonal solution instead of calling the oblique optimizer.

Value

A list. Every path returns the following components:

loadings	Aligned loading matrix.
T	Transformation matrix.
Phi	Factor intercorrelation matrix; the identity for orthogonal and one-factor alignment.
value	Target criterion at the returned solution.
convergence	Logical; TRUE for the closed-form orthogonal solution.
valid	Logical; whether the transformation defines an admissible Phi.
iterations	Number of solver iterations; 0 for the closed-form orthogonal solution.
kappa_T	Condition number of T; a constant 1 on the orthogonal path.
Table	Iteration history with columns iter, f, log10_s, and step; a single placeholder row on the orthogonal path.
method	"orthogonal_procrustes", "oblique_procrustes", or "single_factor_procrustes" for a one-factor oblique request.
line_search_failed	Logical line-search diagnostic.
best_start_index, all_start_indices, all_values, all_converged, all_iterations	Multi-start summary of the starts that were fully optimized; each has a single entry when no random starts were used.

The oblique solver additionally returns screen_start_indices and screen_values (the starts kept by cheap objective screening and their criterion values) together with the counts n_random_starts, n_screened, n_triaged, and n_fully_optimized. These six components are absent for rotation = "orthogonal" and for one-factor models, which are aligned with the orthogonal solution.

Row and column names are preserved where possible. When oblique_normalize = TRUE the returned loadings are back-transformed to the original scale, but value is the criterion on the Kaiser-normalized loadings, so it is not 0.5 * sum((loadings - Target)^2).

See Also

Other factor rotation: efa_schmid_leiman()

Examples

```
## Align an estimated loading matrix to a known target pattern: fit an
## unrotated three-factor model, then rotate its loadings toward the true
## population pattern.
efa_mod <- efa_fit(test_models$baseline$cormat, N = 500, n_factors = 3,
                   estimator = "PAF", rotation = "none")
target <- population_models$loadings$baseline

## Orthogonal target rotation (rigid rotation/reflection):
efa_procrustes(efa_mod$unrot_loadings, target, rotation = "orthogonal")

## Oblique target rotation (lets the aligned factors correlate):
efa_procrustes(efa_mod$unrot_loadings, target, rotation = "oblique")
```

---

efa_reliability *Reliability and common-variance coefficients for a factor solution*

Description

Computes model-based reliability coefficients for a factor solution: McDonald's omega (total, hierarchical, and subscale), standardized Cronbach's alpha, and the H index. For a bifactor solution, it also computes two common-variance indices, ECV and PUC, for the general factor. The result is a tidy, long-format table with one row per coefficient.

Usage

```
efa_reliability(
  model = NULL,
  coefficients = NULL,
  g_name = "g",
  group_names = NULL,
  factor_map = NULL,
  variance = c("correlation", "sums_load"),
  var_names = NULL,
  fac_names = NULL,
  g_load = NULL,
  s_load = NULL,
  u2 = NULL,
  cormat = NULL,
  pattern = NULL,
  Phi = NULL
)
```

Arguments

model	an efa_schmid_leiman(), schmid (psych::schmid()), efa_fit() (oblique), or lavaan object; a raw bifactor loading matrix (general factor first), the loading table of an efa_schmid_leiman() solution, or the pattern matrix of an oblique solution together with its Phi; or NULL to supply the components manually via g_load, s_load, u2, and var_names.
coefficients	character. An optional subset of the coefficients to return, any of "omega_total", "omega_hierarchical", "omega_subscale", "alpha", "H", "ECV", and "PUC". Default NULL returns all of them.
g_name	character. The name of the general factor in the lavaan solution. Only needed for a lavaan second-order or bifactor fit. A fit in which every variable loads on a single factor has no general factor, and reads none. No other input is affected by it. Default is "g".
group_names	character. An optional vector of group names for a lavaan multiple-group fit. Its length must match the number of groups. Not used for any other input – including a single-group lavaan fit, since every one of those is scored as a single ungrouped solution with no group label.
factor_map	matrix. A logical or 0/1 matrix indicating which variable belongs to which group factor, with the same dimensions as the group loading matrix (cross-loadings are allowed). Match its columns to the group factors by position – a map given in a different factor order than the solution still runs, but produces meaningless subscale coefficients. The function warns if a mapped item loads weakly on its assigned factor and more strongly on another one. If NULL (default), each variable is assigned to the group factor on which it loads most strongly. Not used for lavaan input.
variance	character. The total-variance denominator for the coefficients. "correlation" (default) takes each composite's variance from the correlation matrix, giving the observed-score omega. "sums_load" uses the model-implied composite variance from the loadings and the uniquenesses instead; it needs no correlation matrix, so it is the way to score a bare loading matrix or manual components given without one. lavaan input fixes the convention: its composite variances are always model-implied, and include any freed residual covariance as well as the residual variances. Neither convention changes the metric the coefficients are on – with polychoric or tetrachoric correlations, or an ordered lavaan fit, both describe the latent-response composite rather than the ordinal sum score (see Details).
var_names	character. Subtest names in the row order of the loadings. Only needed when model is NULL.
fac_names	character. An optional vector of group-factor names in the column order of the loadings. Taken from the input if NULL. A single-factor solution has no group factors; fac_names then labels its one factor instead. If neither fac_names nor the solution names that factor, it is labelled "F1". Not used for lavaan input, whose factor labels come from the model syntax.
g_load	numeric. General-factor loadings. Only needed when model is NULL.
s_load	matrix. Group-factor loadings. Only needed when model is NULL.

u2 — numeric. Uniquenesses. Only needed when `model` is `NULL`, or to override the communality-based default for a loading matrix. Under `variance = "correlation"`, the coefficients follow from the loadings and the correlation matrix alone, so `u2` only enters the check for an improper solution. Under `"sums_load"`, it is part of every composite's variance.

cormat — matrix or data.frame. A correlation matrix used when `variance = "correlation"`. It must hold the same variables as the solution: named variables in a different order are reordered to match; a different set of variables, or a different number of them, is an error. The matrix must use the solution's own variable names – for manually supplied components, that means the row names of `s_load`, not `var_names`, which only labels the output. Without row names, the variables are matched by position, so supply the matrix in the row order of the loadings. If `NULL`, it is taken from the input object, or reconstructed from `pattern` and `Phi` where possible. Supply the matrix on the same metric as the loadings: a polychoric or tetrachoric matrix keeps the coefficients on the latent-response metric, while a Pearson matrix given with loadings fitted to a polychoric one mixes the two (see Details).

pattern — matrix. Pattern coefficients from a separate oblique solution, used with `Phi` to reconstruct a correlation matrix when `model` is `NULL`. Supply it for a Schmid-Leiman input, whose `s_load` holds the orthogonalized group loadings rather than the oblique ones. It is an alternative to `cormat`, not a supplement: giving both is an error.

Phi — matrix. Factor intercorrelations. `NULL` (default) means uncorrelated group factors, as a Schmid-Leiman or bifactor solution has. It cannot be combined with a general factor: supplied together with a non-zero `g_load`, or with the loading table of an `efa_schmid_leiman()` solution, it is an error.

Supply it together with `s_load` (manually supplied components, with `g_load` zero throughout), or with a loading matrix of two or more factors in `model`, to score the input as a correlated-factors solution instead.

Without `Phi`, a loading matrix in `model` is read as a bifactor solution (general factor first). The one exception is a matrix that still carries the loading class `efa_fit()` gives its output – a class that subsetting a matrix or reordering its rows drops. Such a matrix is rejected instead; supply `Phi` to score it as a correlated-factors solution.

With `Phi`, a matrix in `model` that carries no loading class is read as a correlated-factors solution, and a warning names this reading (drop `Phi` to read it as a bifactor matrix instead).

Given together with `pattern` instead, see `pattern` above. With a fitted solution already in `model`, `Phi` is ignored, with a warning, since that solution carries its own factor intercorrelations.

Details

The function reads many kinds of input: a Schmid-Leiman solution (`efa_schmid_leiman()` or `psych::schmid()`), an oblique `efa_fit()` (correlated-factors) solution, a `lavaan` fit (single-factor, correlated-factors, second-order, or bifactor), a raw bifactor loading matrix, an oblique pattern matrix given with its factor intercorrelations, or manually supplied components.

Coefficients:

The reliability coefficients are McDonald's omegas (McDonald, 1978, 1985, 1999; Zinbarg et al., 2005, 2006, for omega hierarchical specifically), standardized Cronbach's alpha (Cronbach, 1951), and the H index (construct replicability; Hancock & Mueller, 2001).

The omegas give the share of true score variance in a unit-weighted composite. Omega total is the share due to all factors together. Omega hierarchical is the share due to the general factor only. Omega subscale is the share due to the group factors: for the whole scale, or for one specific group factor in a subscale composite.

Alpha is the standardized coefficient, computed from the correlation matrix of the items. Where no such matrix is available – for `lavaan` input, and for components supplied without one – alpha is computed from the model-implied correlation matrix instead, so it then reflects the fitted model rather than the raw data.

The H index is the reliability of an optimally weighted composite. A low value means the factor is not well defined by its indicators.

All of these coefficients describe the raw sum of the variables as fitted, without reverse-coding. If some items are keyed in the opposite direction – for example, a reverse-worded item that was not reverse-scored – they lower that sum's true-score variance. The coefficients then look poor even though the model fits well. The function warns when it detects this. Reverse-code such items before fitting the solution (Flora, 2020).

The sum these coefficients describe is not always the sum of the raw answers. Polychoric and tetrachoric correlations describe the continuous latent responses assumed to underlie ordinal answers; a `lavaan` fit that declares its indicators `ordered` does the same. In that case the loadings, the uniquenesses, the correlation matrix, and the coefficients are all on that latent-response metric. They give the reliability of the unit-weighted sum of the latent responses, which is not observed – not the reliability of the ordinal sum score the user actually computes from the answers. The two can differ substantially, especially where the answers use few categories or are strongly skewed. Green and Yang (2009) give an omega for the ordinal sum score itself, computed from the fitted model and its thresholds; this package does not compute it. Pearson correlations raise no such distinction, because their metric is the answers as scored.

Omega total is lower when a solution reproduces the observed correlations poorly, because residual covariance does not count as true score. `psych::omega()` computes the whole-scale omega total differently: residually, from the observed total-score variance. The two agree when the model reproduces the correlations exactly, and diverge otherwise.

The three coefficients are not generally additive. Omega total need not equal omega hierarchical plus omega subscale, except on the whole-scale row under `variance = "sums_load"`.

A single-factor solution is scored as such on every input route, and returns omega total, alpha, and the H index. Alpha assumes essentially tau-equivalent items, an assumption nested within a one-factor model – so a single factor is the case where reporting alpha is defensible, not merely possible.

The other coefficients are omitted because a single factor does not define them: omega subscale is the variance due to the group factors, and there are none; omega hierarchical would equal omega total, since the one factor accounts for all common variance; and ECV and PUC would both be 1 by construction, which reflects the number of factors in the model rather than evidence of unidimensionality.

Each coefficient answers a different question:

- Omega hierarchical: can the total score be read as a measure of one construct?
- Omega subscale: does a subscale score add anything beyond the general factor?

- The H index: is a factor well defined by its indicators?
- ECV together with PUC: is a unidimensional model defensible?

Alpha assumes essentially tau-equivalent items. Factor analysis instead yields congeneric solutions, for which alpha is only a lower bound. For a multidimensional scale, alpha is rarely the coefficient to report (Gignac, 2014).

Composite reliability and average variance extracted (AVE) are not among the coefficients this function computes. Use `semTools::compRelSEM()` and `semTools::AVE()` to compute them from a `lavaan` fit.

The common-variance indices ECV and PUC (Bonifay et al., 2015; Reise et al., 2013; Rodriguez et al., 2016a, 2016b) describe the general factor, so they are reported for the general factor only. ECV is the share of the common variance explained by the general factor. PUC is the proportion of correlations that reflect general-factor variance alone – correlations between indicators of different group factors. The higher the PUC, the more the general factor resembles the single factor of a unidimensional model.

Input:

`efa_reliability()` reads several kinds of input, illustrated in the examples below.

- **An oblique `efa_fit()` solution** is scored as the correlated-factors model it is. It has no general factor, so it omits the bifactor indices (omega hierarchical, ECV, and PUC). Its whole-scale row is labelled `"total"` rather than `"g"`.
- **An `efa_schmid_leiman()` solution** – or a `schmid` object from `psych::schmid()`, or a raw bifactor loading matrix with the general factor in its first column – is scored as a bifactor solution, with the whole-scale row labelled `"g"`. Indicator-to-factor correspondences come from `factor_map` if supplied (see `factor_map` below). Without one, an `efa_schmid_leiman()` or `psych::schmid()` solution is mapped by each variable's strongest group loading. A raw bifactor matrix instead defaults to its exact zero pattern – supply `factor_map` explicitly if that matrix has no exact zeros (e.g. an estimated rather than a target matrix).
- **A `lavaan` fit** – single-factor, correlated-factors, second-order, or bifactor – is scored per its structure. The structure is detected automatically, not taken from `g_name`. The general-factor coefficients need the latent factors to be uncorrelated: fit a bifactor model with `orthogonal = TRUE` (`lavaan`'s default leaves the factors correlated), and leave a second-order model's first-order factor covariances at zero. A fit whose factors correlate is rejected rather than scored as though they did not. `variance` is not used for `lavaan` input: its composite variances are always model-implied, computed separately per group for a multiple-group fit.
- **A pattern matrix with its `Phi`** – for example, the loadings and `Phi` of an oblique `efa_fit()` solution – is scored as the correlated-factors solution it represents. Uniquenesses are derived from the loadings under `Phi`, unless `u2` is given. Supply the pattern matrix, not the structure matrix – the structure matrix would be read as a pattern too, giving the wrong result. Unlike a fitted solution, this pair carries no correlation matrix. `variance = "correlation"` then needs one, supplied in `cormat`, and errors without it. `variance = "sums_load"` needs none.

A one-factor solution – a one-column loading matrix, a single-factor `lavaan` fit, a single-factor `efa_fit()` solution, or single-factor components – is scored the same way regardless of route (see Coefficients above for what it returns). Its row is labelled with the factor's own name, with `fac_names` if supplied, or with `"F1"` if neither names it; it is never labelled `"g"`.

Manually supplied components (`g_load`, `s_load`, `u2`, `var_names`, `Phi`) follow the same reading rules as the matrix routes above. Do not pass a correlation matrix as `s_load` (or as `model`, for the matrix routes) – supply it as `cormat` instead.

Value

An object of class `efa_reliability`: a long-format data frame with one row per computed coefficient, with columns

`coefficient`	the coefficient name (e.g. `"omega_total"`).
`level`	`"general"` for the general-factor row, `"total"` for the whole-scale row of a correlated-factors solution, and `"group"` for every other row. A single-factor solution has only the general-factor row.
`factor`	the factor label: `"g"` for the general factor of a solution with group factors. A single-factor solution's one factor takes its own name, or `"F1"` if neither the input nor `fac_names` names it. A correlated-factors solution has no general factor, so its first row is labelled `"total"` instead – it describes the composite of every variable, not a factor of the model.
`group`	the group label, or `NA` for a single ungrouped solution.
`value`	the coefficient value.

Structurally undefined cells (for example, ECV and PUC on a group factor) are omitted. The object also carries a `settings` attribute (the total-variance convention used, and whether the solution has a general factor) and a `kind` attribute tagging each coefficient as a reliability coefficient or a common-variance index. It has a `print.efa_reliability()` method.

Source

McDonald, R. P. (1978). Generalizability in factorable domains: Domain validity and generalizability. Educational and Psychological Measurement, 38, 75-79.

McDonald, R. P. (1985). Factor analysis and related methods. Hillsdale, NJ: Erlbaum.

McDonald, R. P. (1999). Test theory: A unified treatment. Mahwah, NJ: Erlbaum.

Cronbach, L. J. (1951). Coefficient alpha and the internal structure of tests. Psychometrika, 16, 297-334.

Gignac, G. E. (2014). On the inappropriateness of using items to calculate total scale score reliability via coefficient alpha for multidimensional scales. European Journal of Psychological Assessment, 30, 130-139.

Flora, D. B. (2020). Your coefficient alpha is probably wrong, but which coefficient omega is right? A tutorial on using R to obtain better reliability estimates. Advances in Methods and Practices in Psychological Science, 3, 484-501.

Green, S. B., & Yang, Y. (2009). Reliability of summed item scores using structural equation modeling: An alternative to coefficient alpha. Psychometrika, 74, 155-167.

Zinbarg, R. E., Revelle, W., Yovel, I., & Li, W. (2005). Cronbach's alpha, Revelle's beta, and McDonald's omega H: Their relations with each other and two alternative conceptualizations of reliability. Psychometrika, 70, 123-133.

Zinbarg, R. E., Yovel, I., Revelle, W., & McDonald, R. P. (2006). Estimating generalizability to a latent variable common to all of a scale's indicators: A comparison of estimators for omega H. Applied Psychological Measurement, 30, 121-144.

Hancock, G. R., & Mueller, R. O. (2001). Rethinking construct reliability within latent variable systems. In R. Cudeck, S. du Toit, & D. Sörbom (Eds.), Structural equation modeling: Present

and future - A Festschrift in honor of Karl Jöreskog (pp. 195-216). Lincolnwood, IL: Scientific Software International.

Bonifay, W. E., Reise, S. P., Scheines, R., & Meijer, R. R. (2015). When are multidimensional data unidimensional enough for structural equation modeling? An evaluation of the DETECT multidimensionality index. Structural Equation Modeling, 22, 504-516.

Reise, S. P., Scheines, R., Widaman, K. F., & Haviland, M. G. (2013). Multidimensionality and structural coefficient bias in structural equation modeling: A bifactor perspective. Educational and Psychological Measurement, 73, 5-26.

Rodriguez, A., Reise, S. P., & Haviland, M. G. (2016a). Applying bifactor statistical indices in the evaluation of psychological measures. Journal of Personality Assessment, 98, 223-237.

Rodriguez, A., Reise, S. P., & Haviland, M. G. (2016b). Evaluating bifactor models: Calculating and interpreting statistical indices. Psychological Methods, 21, 137-150.

See Also

`efa_fit()` for the solution these are computed from, and `OMEGA()`, the superseded function that returns these same coefficients in a wide, per-factor layout.

Other reliability coefficients: `efa_schmid_leiman()`, `print.efa_reliability()`

Examples

```
## From an oblique EFA (correlated-factors) solution. With no factor_map, each
## item is auto-assigned to its highest-loading factor.
efa_mod <- efa_fit(test_models$baseline$cormat, N = 500, n_factors = 3,
                   estimator = "PAF", rotation = "promax")
efa_reliability(efa_mod)

## From a Schmid-Leiman solution, with an explicit indicator-to-factor map.
sl_mod <- efa_schmid_leiman(efa_mod, estimator = "PAF")
fc <- sl_mod$sl[, c("F1", "F2", "F3")] >= .2
efa_reliability(sl_mod, factor_map = fc)

## Request a subset of the coefficients only.
efa_reliability(sl_mod, factor_map = fc,
                coefficients = c("omega_total", "alpha"))

## From an oblique pattern matrix and its factor intercorrelations. This is
## the same correlated-factors solution, and gives the same coefficients.
efa_reliability(efa_mod$rot_loadings, Phi = efa_mod$Phi,
                cormat = test_models$baseline$cormat)

## From lavaan fits: a bifactor solution, and a correlated-factors one.

if (requireNamespace("lavaan", quietly = TRUE)) {
mod_cf <- 'F1 =~ V1 + V2 + V3 + V4 + V5 + V6
           F2 =~ V7 + V8 + V9 + V10 + V11 + V12
           F3 =~ V13 + V14 + V15 + V16 + V17 + V18'
mod <- paste(mod_cf, 'g =~ V1 + V2 + V3 + V4 + V5 + V6 + V7 + V8 + V9 + V10 +
                          V11 + V12 + V13 + V14 + V15 + V16 + V17 + V18',
             sep = "\n")
```

```
fit <- lavaan::cfa(mod, sample.cov = test_models$baseline$cormat,
                   sample.nobs = 500, estimator = "ml", orthogonal = TRUE)
print(efa_reliability(fit, g_name = "g"))

## No general factor: omega hierarchical, ECV, and PUC are omitted.
fit_cf <- lavaan::cfa(mod_cf, sample.cov = test_models$baseline$cormat,
                      sample.nobs = 500, estimator = "ml")
efa_reliability(fit_cf)
}
```

`efa_retain` *Various factor retention criteria*

Description

Choosing the number of factors to retain is one of the most important decisions in an exploratory factor analysis (EFA). Many criteria exist to help with this choice. This function runs several of them together, and can also check whether the data are suitable for factor analysis.

Usage

```
efa_retain(
  x,
  criteria = c("CD", "EKC", "HULL", "MAP", "NEST", "PARALLEL"),
  suitability = TRUE,
  N = NA,
  use = c("pairwise.complete.obs", "all.obs", "complete.obs", "everything",
    "na.or.complete"),
  cor_method = c("pearson", "spearman", "kendall", "poly", "tetra"),
  n_factors_max = NA,
  N_pop = 10000,
  N_samples = 500,
  alpha = 0.3,
  ...,
  max_iter_CD = 50,
  n_fac_theor = NA,
  estimator = c("ML", "PAF", "ULS"),
  gof = c("CAF", "CFI", "RMSEA"),
  eigen_type_HULL = c("SMC", "PCA", "EFA"),
  eigen_type_other = c("SMC"),
  n_factors = 1,
  n_datasets = 1000,
  percent = 95,
  decision_rule = c("means", "percentile", "crawford"),
  ekc_type = lifecycle::deprecated(),
  n_datasets_nest = 1000,
```

```
  alpha_nest = 0.05,
  show_progress = FALSE,
  estimate_control = NULL
)
```

Arguments

x	data.frame or matrix. Raw data, or a correlation matrix. If "CD" is included as a criterion, x must be raw data.
criteria	character. Which factor retention methods to run: one or more of "CD", "EKC", "HULL", "KGC", "MAP", "NEST", "PARALLEL", "SCREE", and "SMT" (see details). The default runs a subset of commonly used, well-performing methods, listed in the details.
suitability	logical. Whether the data should be checked for suitability for factor analysis using Bartlett's test of sphericity and the Kaiser-Meyer-Olkin 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. Correlation computed from raw data: "pearson", "spearman", or "kendall" (passed to stats::cor()), or "poly" / "tetra" for polychoric / tetrachoric correlations (a two-step estimator). CD, PARALLEL, NEST, HULL, and SMT do not support "poly" / "tetra" and are skipped automatically if you request them together. Default is "pearson".
n_factors_max	numeric. Passed to efa_cd(). 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. Passed to efa_cd(). Size of finite populations of comparison data. Default is 10000.
N_samples	numeric. Passed to efa_cd(). Number of samples drawn from each population. Default is 500.
alpha	numeric. Passed to efa_cd(). The alpha level used to test the significance of the improvement added by an additional factor. Default is .30.
...	Further arguments passed to efa_fit() in efa_parallel(), efa_kgc(), efa_scree(), efa_nest(), and (through its parallel analysis and its own candidate fits) efa_hull(). An argument that efa_fit() does not recognize causes an error. The estimation tuning knobs are not passed here; they live in estimate_control. The standard-error arguments (se, b_boot, ci, seed) are not accepted, because the criterion fits are internal steps whose standard errors are not reported. Arguments listed after ... must be given by their full name (R matches an abbreviated name only against the arguments before ...), so a tuning knob such as max_iter cannot be mistaken for max_iter_CD.
max_iter_CD	numeric. Passed to efa_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 efa_hull(). Theoretical number of factors to retain. The Hull method uses one plus the larger of this number and the number of factors suggested by efa_parallel() as its upper bound.

estimator — character. Passed to efa_fit() in efa_hull(), efa_kgc(), efa_scree(), efa_parallel(), and efa_nest(). The estimator to use. One of "PAF", "ULS", or "ML", for principal axis factoring, unweighted least squares, and maximum likelihood, respectively. The default here is "ML". Some criteria default to something else when called on their own (for example, efa_hull() defaults to "PAF"), so results from efa_retain() can differ from calling that criterion directly unless you set estimator to match. In efa_kgc(), efa_scree(), and efa_parallel() it only takes effect when the respective eigen_type includes "EFA".

gof — character. Passed to efa_hull(). The goodness of fit index to use. Either "CAF", "CFI", or "RMSEA", or any combination of them. With the "PAF" estimator, 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 efa_parallel() in efa_hull(). What the eigenvalues in the parallel analysis are based on. One of "SMC", "PCA", or "EFA" – different ways of estimating how much variance each indicator shares with the others before the eigenvalues are computed. "SMC" (default) uses each indicator's squared multiple correlation with the others (its diagonal value in the correlation matrix). "PCA" leaves the diagonal at 1, so each indicator's total variance – not just the shared part – feeds into the eigenvalues. "EFA" uses the communalities from a fitted EFA solution instead.

eigen_type_other — character. Passed to efa_kgc(), efa_scree(), and efa_parallel(). The same as eigen_type_HULL, but multiple inputs are possible here (any combination of "PCA", "SMC", and "EFA"). Default is "SMC".

n_factors — numeric. Passed to efa_parallel() (also within efa_hull()), efa_kgc(), and efa_scree(). Number of factors to extract if "EFA" is included in eigen_type_HULL or eigen_type_other. Default is 1.

n_datasets — numeric. Passed to efa_parallel() (also within efa_hull()). The number of datasets to simulate. Default is 1000.

percent — numeric. Passed to efa_parallel() (also within efa_hull()). The percentile to take from the simulated eigenvalues. Default is 95.

decision_rule — character. Passed to efa_parallel() (also within efa_hull()). Which rule to use to determine the number of factors to retain. Default is "means", which uses 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).

ekc_type — **[Deprecated]** Accepted and ignored. It used to select between two ways to compute the efa_ekc() reference values. The "AM2019" reference values do not depend on the observed eigenvalues. They therefore skip the empirical correction that defines the criterion, so they are no longer computed.

`n_datasets_nest`
: numeric. Passed to `efa_nest()`. The number of datasets to simulate. Default is 1000.

`alpha_nest`
: numeric. Passed to `efa_nest()`. The alpha level to use. The reference values are the eigenvalues at the (1 - alpha_nest) percentile. Default is .05.

`show_progress`
: logical. Whether a progress bar should be shown in the console. Default is FALSE.

`estimate_control`
: an `estimate_control()` object with the estimation settings for the `efa_fit()` fits run by the criteria that fit a model (`efa_hull()`, `efa_kgc()`, `efa_scree()`, `efa_parallel()`, `efa_nest()`, and `efa_smt()`). `NULL` (default) uses the `efa_fit()` defaults. It only applies to criteria that fit a model, and only to the parts of that fit each criterion actually runs. `efa_cd()`, `efa_ekc()`, and `efa_map()` fit no model. `efa_kgc()`, `efa_scree()`, and `efa_parallel()` only fit one when their `eigen_type` includes `"EFA"`. `efa_smt()` fits with maximum likelihood by definition, so only `start_method` takes effect there. All fits are unrotated, so no rotation settings apply.

Details

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

- Bartlett's test of sphericity (see `efa_bartlett()`)
- Kaiser-Meyer-Olkin criterion (see `efa_kmo()`)

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

- Comparison data (see `efa_cd()`)
- Empirical Kaiser criterion (see `efa_ekc()`)
- Hull method (see `efa_hull()`)
- Kaiser-Guttman criterion (see `efa_kgc()`)
- Velicer's minimum average partial, MAP (see `efa_map()`)
- Next Eigenvalue Sufficiency Test, NEST (see `efa_nest()`)
- Parallel analysis (see `efa_parallel()`)
- Scree plot (see `efa_scree()`)
- Sequential chi-square model tests, RMSEA lower bound, and AIC (see `efa_smt()`)

The default `criteria` are comparison data, the empirical Kaiser criterion, the Hull method, MAP, NEST, and parallel analysis. No single criterion is the most accurate in all conditions. `efa_retain()` therefore runs several criteria together, and the printed summary gives the range of their suggestions and the most common one. Auerswald and Moshagen (2019) compare the criteria and give guidance on the selection.

The comparison data, parallel analysis, and NEST criteria compare the data against simulated reference data, so their suggested numbers of factors vary slightly from run to run. The Hull method also varies, because it calls `efa_parallel()` to set its upper bound. Call `base::set.seed()` before `efa_retain()` to make the results reproducible.

Value

A list of class `c("efa_retain", "N_FACTORS")`, the trailing class keeping `inherits(x, "N_FACTORS")` working for code written against the superseded name. It contains

`suitability` A list with the results from `efa_bartlett()` and `efa_kmo()` (`bartlett` and `kmo`), or `NULL` if `suitability = FALSE`.

`outputs` A named list with one `efa_retention` object per factor retention criterion that was run (see, e.g., `efa_ekc()`).

`n_factors` A named numeric vector with the suggested number of factors per criterion and, where a criterion has several variants, per variant (e.g. `EKC_BvA2017` or `PARALLEL_SMC`). Criteria without a numeric suggestion (the scree plot) are not included. The printed summary's "most common" value is based on each criterion's own most frequent (modal) suggestion among its variants, not a plain tally of this vector, so counting entries here by hand can give a different answer.

`not_run` A named character vector with the criteria that were skipped or failed and the reason, or `NULL` if all requested criteria ran.

`settings` A list of the settings used. Its `criteria` element records the requested criteria, in the order they were given, while `outputs` and `n_factors` are in the order in which the criteria were run. `gof` records the requested Hull goodness-of-fit indices, and `gof_used` records the ones the Hull method actually computed (it reduces them to `"CAF"` for the PAF estimator). `gof_used` is `NA` when HULL was not requested, was skipped, or failed.

Source

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

See Also

`efa_screen()` for data screening before retention, and `efa_fit()` to extract the chosen number of factors.

Other factor retention criteria: `efa_cd()`, `efa_ekc()`, `efa_hull()`, `efa_kgc()`, `efa_map()`, `efa_nest()`, `efa_parallel()`, `efa_scree()`, `efa_smt()`

Examples

```
# Default criteria, with correlation matrix and estimator "ML" (where needed)
# This will throw a warning for CD, as no raw data were specified
# The simulation-based criteria are seeded to make the run reproducible
set.seed(42)
nfac_all <- efa_retain(test_models$baseline$cormat, N = 500, estimator = "ML",
                       n_datasets = 100, n_datasets_nest = 100)

# The same as above, but without "CD"
nfac_wo_CD <- efa_retain(test_models$baseline$cormat, criteria = c("EKC",
                         "HULL", "PARALLEL", "NEST"), N = 500,
```

```
                        estimator = "ML", n_datasets = 100,
                        n_datasets_nest = 100)

# Use PAF instead of ML (this will take longer). PAF only supports "CAF" as
# gof for the Hull method, so set it explicitly to avoid the automatic message.
nfac_PAF <- efa_retain(test_models$baseline$cormat, criteria = c("EKC",
                       "HULL", "PARALLEL", "NEST"), N = 500,
                       estimator = "PAF", gof = "CAF", n_datasets = 100,
                       n_datasets_nest = 100)

# Back to the default ML estimator (unlike above), with only "PCA" type eigenvalues
nfac_PCA <- efa_retain(test_models$baseline$cormat, criteria = c("EKC",
                       "HULL", "PARALLEL", "NEST"), N = 500,
                       estimator = "ML", eigen_type_other = "PCA",
                       n_datasets = 100, n_datasets_nest = 100)

# Use raw data, such that CD can also be performed
nfac_raw <- efa_retain(GRiPS_raw, estimator = "ML", N_pop = 500,
                        N_samples = 20, n_datasets = 100,
                        n_datasets_nest = 100)
```

`efa_schmid_leiman` *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_fit()` 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. The group factors of the returned solution are sorted and relabelled, so their column order can differ from the input solution's (see Details).

Usage

```
efa_schmid_leiman(
  x,
  Phi = NULL,
  estimator = c("PAF", "ML", "ULS", "MINRES"),
  g_name = "g",
  estimate_control = NULL,
  ...
)
```

Arguments

x	object of class efa_fit(), class psych::fa(), class lavaan::lavaan(), a matrix, or an efa_loadings/loadings object. If class efa_fit() or class psych::fa(), pattern coefficients and factor intercorrelations are taken from this object. If class lavaan::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).
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.
estimator	character. One of "PAF", "ML", or "ULS" to use principal axis factoring, maximum likelihood, or unweighted least squares, respectively, used in efa_fit() to find the second-order loadings. "MINRES" is accepted as a synonym for "ULS" (the same estimator).
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".
estimate_control	an estimate_control() object with the estimation settings for the second-order efa_fit() fit, including the type preset. NULL (default) uses the efa_fit() defaults. The second-order fit is unrotated, so no rotation settings apply.
...	Arguments to be passed to efa_fit(). The estimation tuning knobs are not passed here; they live in estimate_control, and the standard-error arguments (se, b_boot, ci, seed) are not accepted because the second-order fit is an internal step run against a placeholder sample size and only its loadings are kept.

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 factor analyzed to extract a single second-order (general) factor, yielding a two-level hierarchical 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 efa_reliability()).

Where the first-order factors come from a loading matrix – an efa_fit() or a psych::fa() solution, or a pattern matrix supplied with Phi – they are sorted by the number in their column labels, so that "F10" follows "F2" rather than "F1". The sort needs a number in every column label; columns that carry no labels, or a label without a number, keep the order they arrive in. A second-order lavaan solution is not sorted at all: its first-order factors keep the order the model declares them in.

The columns are then labelled "F1" to "Fk" by position, on every route, and the input solution's own factor names are not carried over. A factor a lavaan model calls "F3" can therefore come back as "F1". Where the sort applies it is independent of how the input orders its factors, so the group factors of the returned sl matrix can also be in a different order from the columns they came from. A psych::fa() solution shows this most readily: it orders its columns by their sums of squared loadings, but keeps each factor's own number in its label, so those numbers arrive out of order. A

solution whose columns are "PA2", "PA3", "PA1" comes back with those same three factors sorted as PA1, PA2, PA3 and labelled "F1", "F2", "F3". The first group factor of the result is then the third column of the input. The same holds against psych::schmid(), whose columns keep the input order: the two solutions agree column for column only after one of them is permuted to the other's order. An efa_fit() solution already labels its factors "F1" to "Fk" in that order, so nothing moves for one; the reordering shows itself for a psych::fa() solution, and for a pattern matrix supplied with labels of its own.

Read the group factors from the returned matrix, therefore, rather than from the input. An indicator-to-factor map is matched to the group factors by position, so one built in the input solution's column order lines up only where the columns did not move; where they did, efa_reliability() or OMEGA() scores each composite against the wrong factor. Build such a map from the "F1" to "Fk" columns of the returned sl matrix instead, which is right on every route. The fac_names of efa_reliability() are matched by position in the same way, so names given in the input solution's order label the wrong subscales, and do so without any sign.

Value

A list of class c("efa_schmid_leiman", "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.
convergence	Integer convergence code of the second-order EFA (0 = converged); NA for a lavaan input. See efa_fit().
settings	list. The settings (arguments) used in EFA to get the second-order loadings.

Source

Schmid, J. & Leiman, J. M. (1957). The development of hierarchical factor solutions. Psychometrika, 22(1), 53–61. doi:10.1007/BF02289209

Wolff, H.-G., & Preising, K. (2005). Exploring item and higher order factor structure with the Schmid-Leiman solution: Syntax codes for SPSS and SAS. Behavior Research Methods, 37 , 48–58. doi:10.3758/BF03206397

See Also

Other factor rotation: efa_procrustes()

Other reliability coefficients: efa_reliability(), print.efa_reliability()

Examples

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

```
SL_EFAtools <- efa_schmid_leiman(EFA_mod, estimator = "PAF",
                                 estimate_control = estimate_control(type = "EFAtools"))


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


## 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
SL_flex <- efa_schmid_leiman(EFA_mod$rot_loadings, Phi = EFA_mod$Phi, estimator = "PAF",
                             estimate_control = estimate_control(type = "EFAtools"))


## Use with a lavaan second-order CFA output
if (requireNamespace("lavaan", quietly = TRUE)) {

# 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 <- efa_schmid_leiman(fit, g_name = "g")

}
```

`efa_scores` *Estimate factor scores and score-quality diagnostics for an EFA model*

Description

Computes factor-score weights, and (from raw data) the factor scores themselves, for an `efa_fit()` solution or a directly supplied loading matrix. It also returns score-quality diagnostics: the score intercorrelations, the determinacy (validity) and univocality of each score, and Guttman's indeterminacy index. Factor scores are returned only when raw data are supplied; a correlation matrix yields the weights and diagnostics alone.

Usage

```
efa_scores(
  x,
  f,
  Phi = NULL,
  rho = NULL,
 method = c("regression", "Bartlett", "Anderson", "tenBerge", "Harman", "components")
)
```

Arguments

x	data.frame or matrix. Raw data (needed to obtain factor scores) or a correlation matrix (yields weights and diagnostics only). When f is a directly supplied loading matrix, a correlation-matrix x also supplies the correlations the weights are derived from; when f is an efa_fit() object, its own fitted correlations are used instead (supply rho to derive the weights from another matrix). x describes the model variables either way. When raw data carry column names, they are matched to the model variables by name (any extra columns are ignored, and a model variable missing from x is an error). A named correlation matrix is likewise matched to the loading rows by name; its row and column names must use the same order, and it must carry one row and column for each model variable. Unnamed input is matched by position. Raw data are scored as supplied: no imputation is performed, so a case with a missing value on any model variable receives NA scores (and is reported as not scored). A model variable that carries no usable spread in x – constant, infinite, or observed fewer than twice – is an error.
f	object of class efa_fit(), an efa_loadings object, or a matrix of factor loadings.
Phi	matrix. Factor intercorrelations. Only used when a loading matrix is supplied directly in f; taken from the efa object otherwise, in which case a supplied Phi is ignored with a warning. Named rows and columns are matched to the loading columns and must use the same order. Default is NULL, in which case the factors are assumed uncorrelated.
rho	matrix. Correlation matrix used to derive the scoring weights. Defaults to NULL, in which case f$orig_R is used for an efa object; for a directly supplied loading matrix, x itself when it is a correlation matrix, and cor(x, use = "pairwise") otherwise. Pass a matrix here to score against a correlation other than the one implied by f/x. Named rows and columns are matched to the loading rows; row and column names must use the same order.
method	character. The factor-score method: one of "regression" (default), "Bartlett", "Anderson", "tenBerge", "Harman", or "components".

Details

Each method combines the loadings with some or all of the factor correlations and the scoring correlation matrix into weights in a different way:

"regression" Thurstone's (1935) regression scores.

`"Bartlett"` Bartlett's (1937) scores.

`"Anderson"` Anderson & Rubin's (1956) scores.

`"tenBerge"` ten Berge, Krijnen, Wansbeek & Shapiro's (1999) scores.

`"Harman"` Harman's (1976) scores, based on an idealized variable (a hypothetical variable that would correlate perfectly with the factor).

`"components"` Component scores. These are formed from the raw, uncentered data (`X %*% W`) rather than the standardized data, so unlike the other methods they are on the scale of the input variables. The diagnostics below describe the standardized combination `scale(X) %*% W`, and therefore differ from the realized correlations of the returned scores whenever the variables have unequal variances.

The determinacy (validity) of a score is its correlation with the factor it estimates, computed from the returned weights; for regression scores it is the multiple correlation between the factor and the observed variables (Guttman, 1955; Grice, 2001). The off-diagonal score-factor correlations give the univocality (the correlation of a score with the *other* factors). Guttman's (1955) indeterminacy index, `2 rho^2 - 1`, is the minimum correlation between two equally valid sets of scores. For a `method` other than `"regression"` both quantities are specific to those scores: the determinacy is the method's own score-factor correlation (never larger than the regression value), and the reported `guttman` follows from it.

Determinacies close to 1 mean the scores stand in for the factor with little loss; Grice (2001) regards values of about .90 and above as the level required before scores are interpreted for individual cases, and treats lower values as usable only for group-level research. The Guttman index makes the same point more sharply, because a factor score is never the factor: at `rho = .90` two equally valid sets of scores can still correlate as low as .62, and at `rho = .80` as low as .28, so the rank order of cases is not unique.

Which `method` to prefer follows from what the scores are for. Regression scores correlate most highly with the factor, but they are biased towards it and correlate across factors even when the model is orthogonal. Bartlett scores are conditionally unbiased, which makes them the choice when the scores stand in for the factor in a later model. `"tenBerge"` reproduces the factor intercorrelations `Phi`, so it is the choice when the scores will be correlated with each other or with external variables. `"Anderson"` forces the scores to be uncorrelated with unit variance and is appropriate only when the factors themselves are orthogonal. `"components"` is a weighted sum of the observed variables rather than an estimate of a common factor.

Value

An object of class `efa_scores`, a list containing:

`weights`	The `p` by `m` factor-score weight matrix.
`scores`	The factor scores (`n` by `m`), or `NULL` when a correlation matrix was supplied. A case with a missing value on any model variable is not scored and keeps `NA` in every column.
`r.scores`	The `m` by `m` correlations of the factor-score estimates (see Details for the `"components"`-method scale caveat).
`score_cor`	The `m` by `m` score-factor correlation matrix; its diagonal is the determinacy (validity) of each score and its off-diagonals the univocality.

`determinacy` A data frame with, per factor, the determinacy `rho`, the squared determinacy `rho2`, and Guttman's indeterminacy index `guttman`.

`settings` A list of the settings used, including the number of supplied observations `n_obs` and the number of them that could be scored `n_scored`.

Source

Thurstone, L. L. (1935). *The vectors of mind*. University of Chicago Press.

Bartlett, M. S. (1937). The statistical conception of mental factors. *British Journal of Psychology, 28*, 97-104.

Anderson, T. W., & Rubin, H. (1956). Statistical inference in factor analysis. In *Proceedings of the Third Berkeley Symposium on Mathematical Statistics and Probability* (Vol. 5, pp. 111-150). University of California Press.

Guttman, L. (1955). The determinacy of factor score matrices with implications for five other basic problems of common-factor theory. *British Journal of Statistical Psychology, 8*, 65-81.

ten Berge, J. M. F., Krijnen, W. P., Wansbeek, T., & Shapiro, A. (1999). Some new results on correlation-preserving factor scores prediction methods. *Linear Algebra and its Applications, 289*, 311-318.

Grice, J. W. (2001). Computing and evaluating factor scores. *Psychological Methods, 6*, 430-450.

See Also

`efa_fit()` for the solution these are computed from.

Other factor scoring: `print.efa_scores()`

Examples

```
# Weights and score diagnostics from an EFA on a correlation matrix
efa <- efa_fit(test_models$baseline$cormat, n_factors = 3, N = 500,
               estimator = "PAF", rotation = "oblimin")
fs <- efa_scores(test_models$baseline$cormat, f = efa)
fs
summary(fs)

# Factor scores from raw data (Bartlett method)

efa_raw <- efa_fit(GRiPS_raw, n_factors = 1, estimator = "PAF")
efa_scores(GRiPS_raw, f = efa_raw, method = "Bartlett")


# Loadings supplied directly, with the factor intercorrelations
efa_scores(test_models$baseline$cormat, f = efa$rot_loadings, Phi = efa$Phi)
```

`efa_scree` *Scree plot*

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

```
efa_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", "poly", "tetra"),
  n_factors = 1,
  estimate_control = NULL,
  ...
)
```

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. Default is `c("PCA", "SMC", "EFA")`, i.e. all three; `"EFA"` is the only one that fits a model.
`use`	character. Passed to `stats::cor()` if raw data is given as input. Default is "pairwise.complete.obs".
`cor_method`	character. Correlation computed from raw data: `"pearson"`, `"spearman"`, or `"kendall"` (passed to `stats::cor()`), or `"poly"` / `"tetra"` for polychoric / tetrachoric correlations of ordinal / binary data (a two-step estimator). Default is "pearson".
`n_factors`	numeric. Number of factors to extract if "EFA" is included in `eigen_type`. Default is 1.

estimate_control
: an estimate_control() object with the estimation settings for the efa_fit() fit that provides the communalities when "EFA" is included in eigen_type. NULL (default) uses the efa_fit() defaults. The fit is unrotated, so no rotation settings apply.

...
: Additional arguments passed to efa_fit(). For example, estimator, to change the estimator (PAF is default). The estimation tuning knobs are not passed here; they live in estimate_control, and the standard-error arguments (se, b_boot, ci, seed) are not accepted because the fit is an internal step that keeps only its communalities.

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 efa_scree function can also be called together with other factor retention criteria in the efa_retain() function.

Value

An object of class efa_retention (see print.efa_retention() and plot.efa_retention() for the print and plot methods). The scree plot is a visual criterion, so it returns no numeric suggestion. Its main fields are:

results
: A list with one record per requested eigenvalue type, each holding the eigenvalues used for the scree plot.

settings
: A list of the settings used.

Source

Cattell, R. B. (1966). The scree test for the number of factors. Multivariate Behavioral Research, 1(2), 245–276. https://doi.org/10.1207/s15327906mbr0102_10

Zwick, W. R., & Velicer, W. F. (1986). Comparison of five rules for determining the number of components to retain. Psychological Bulletin, 99, 432–442. https://doi.org/10.1037/0033-2909.99.3.432

See Also

efa_retain() as a wrapper function for this and the other factor retention criteria.

Other factor retention criteria: efa_cd(), efa_ekc(), efa_hull(), efa_kgc(), efa_map(), efa_nest(), efa_parallel(), efa_retain(), efa_smt()

Examples

```
efa_scree(test_models$baseline$cormat, eigen_type = c("PCA", "SMC"))
```

efa_screen *Screen data for exploratory factor analysis*

Description

Checks whether your data are suitable for exploratory factor analysis (EFA). From a correlation matrix or raw data, it reports the Kaiser-Meyer-Olkin (KMO) measure of sampling adequacy, Bartlett's test of sphericity, the determinant and condition number of the correlation matrix, and each variable's squared multiple correlation (SMC). When you supply raw data, it also reports each variable's variance and percentage of missing values, category counts for categorical variables, tests of multivariate normality, and multivariate outliers.

Usage

```
efa_screen(
  x,
  N = NA,
  use = c("pairwise.complete.obs", "all.obs", "complete.obs", "everything",
    "na.or.complete"),
  cor_method = c("pearson", "spearman", "kendall", "poly", "tetra"),
  mcd_alpha = 0.5,
  outlier_cutoff = 0.975,
  seed = NULL
)
```

Arguments

x	data.frame or matrix. Raw data, or a correlation matrix. Needs at least three variables, none of which is a perfect linear combination of the others.
N	numeric. The number of observations. Set this only when you supply a correlation matrix; it is needed for Bartlett's test of sphericity and is taken from the data automatically when you supply raw data. Default is NA.
use	character. How to handle missing values in raw data. For cor_method = "pearson", "spearman", or "kendall" this is passed to stats::cor(). For "poly" or "tetra" the same rule is applied to the raw data before the correlations are estimated; "all.obs" and "everything" then stop with an error on any missing value, instead of returning NA correlations. Default is "pairwise.complete.obs".
cor_method	character. How to compute correlations from raw data: "pearson", "spearman", or "kendall" (via stats::cor()), or "poly" / "tetra" for polychoric / tetrachoric correlations of ordinal or binary data. A Spearman or Kendall correlation matrix is screened on its own scale, not converted to look like a Pearson correlation matrix; Kendall's tau in particular measures something different from a Pearson correlation, not just a rescaled version of it. Default is "pearson".
mcd_alpha	numeric. The proportion of cases used to build the robust outlier estimate, between 0.5 and 1. The default, 0.5, is the most robust choice; a larger value uses more of the data but resists outliers less well. Used only with raw data.

`outlier_cutoff` numeric. The probability used to set the cutoff for flagging a multivariate outlier, between 0.5 and 0.9999. Default is `0.975`. Used only with raw data.

`seed` integer. A seed for the random subsets used by the outlier detection, so the result is reproducible. Does not affect your random-number generator elsewhere. Default is `NULL`. Used only with raw data.

Details

The diagnostics are computed from the analysis correlation matrix R:

KMO The Kaiser-Meyer-Olkin measure of sampling adequacy (Kaiser, 1970; Kaiser & Rice, 1974), overall and for each variable; see `efa_kmo()`. It shows how much common variance your variables share. Higher values are better; a common rule of thumb treats values below .50 as unacceptable.

Bartlett Bartlett's (1951) test of sphericity: the likelihood-ratio test of whether the correlation matrix is an identity matrix, i.e., whether your variables correlate with each other at all; see `efa_bartlett()`. A significant result supports doing a factor analysis. The test needs the sample size `N`; without it, this diagnostic is skipped with a warning and `$bartlett` is `NULL`. If `N` is too small relative to the number of variables, the statistic is `NA`, also with a warning.

Determinant The determinant of R, reported as a number only. It falls as you add variables even when the variables are not collinear, so a fixed cut-off on it (such as the 0.00001 often quoted from Field, 2018) says more about how many variables you have than about your data. Use the condition number instead.

Condition number The ratio of the largest to the smallest eigenvalue of R. Its square root, the condition index, is the collinearity diagnostic of Belsley, Kuh & Welsch (1980); it drives the printed report and its recommendation. An index of 10 or less is rarely of interest. An index above 30 flags a near linear dependency: two or more variables that together carry much the same information. An index between the two is not negligible, but it stays below the value that flags a dependency. Belsley (1991) gives 30 as one example value and calls the choice of a cut-off "somewhat of an art form", so the report grades an index above 30 by its position on the scale 1, 3, 10, 30, 100, 300, 1000: moderate (30 to 100), strong (100 to 300), or very strong (above 300). These values come from regression diagnostics on data that are not centred, but a correlation matrix is centred, so use them as a guide and not as a test.

SMC The squared multiple correlation of each variable with all the others. A low value flags a variable that has little in common with the rest of your set.

Variance and missing data For raw data: each variable's variance (over its available values) and percentage of missing values, computed from every row you supplied. These missing-value percentages explain why the correlation matrix's sample size (`N`) can be smaller than the number of rows in your data. Ordered-factor columns are recoded to integer levels first, so `variance` reflects those codes.

Categories For raw data: for each variable with fewer than ten distinct values (treated as categorical), the count of responses in each category. A category with fewer than five responses is flagged as sparse, and an unused category between the smallest and largest response is flagged as empty. As a rule of thumb, items with fewer than five response categories are better analysed with `cor_method = "poly"` or `"tetra"` than with Pearson correlations (Rhemtulla et al., 2012).

Multivariate normality For raw data, using only complete cases: two tests of multivariate normality, Mardia's (1970) test of skewness and kurtosis and the Henze-Zirkler (1990) test. A small p-value on either test suggests your data depart from a multivariate normal distribution, a reason to prefer a robust or ordinal method over normal-theory maximum likelihood. In a very small sample the kurtosis statistic is NA. The Henze-Zirkler p-value is not available with more than about 50 to 60 variables; its test statistic is still reported.

Outliers For raw data, using only complete cases: multivariate outliers, found from a robust estimate of each case's distance from the centre of your data (the minimum covariance determinant method; Rousseeuw & Van Driessen, 1999). A flagged case is unusually far from the rest of your sample. When there are too few complete cases, the variables are too collinear, or too many cases share identical answers, a plain (non-robust) distance is used instead, with a warning explaining why.

Value

An object of class `efa_screen`, a list containing:

`kmo`	A list with the overall KMO (`KMO`) and the per-variable KMO (`KMO_i`).
`bartlett`	A list with Bartlett's chi-square statistic (`chisq`), its `p_value`, and its degrees of freedom (`df`); `chisq` and `p_value` are `NA` when `N` was too small for the correction. `NULL` when `N` is unavailable.
`determinant`	The determinant of the correlation matrix.
`condition`	The condition number of the correlation matrix (largest eigenvalue over smallest).
`smc`	The per-variable squared multiple correlations.
`per_item`	A data frame with one row per variable (row names are the variable names): `variance`, `missing` (percentage), `smc`, `kmo_i`, and `flags` (any sparse/empty-category issues). `NULL` when a correlation matrix is supplied instead of raw data.
`normality`	A list with `mardia` (skewness `skewness`, `skewness_df`, `skewness_p`, kurtosis `kurtosis` and `kurtosis_p`, and the underlying `b1p`/`b2p`), `hz` (the Henze-Zirkler `statistic` and its `p_value`), and `n_complete` (the number of complete cases used). `NULL` without raw data, or a note explaining why when the complete-case data cannot support the tests.
`outliers`	A list with `distances` (each complete case's robust distance, named by its row number), `cutoff` (the flagging threshold, on the same scale as `distances`), `flagged` (the row numbers exceeding `cutoff`), `center` and `cov` (the robust location and scatter), `method` (`"mcd"` or the `"classical"` fallback), `fallback_reason` (why the robust estimate was unavailable, when `method` is `"classical"`), and `n_complete`. `NULL` without raw data, or a note explaining why when no covariance can be formed.
`categories`	A named list with the response-category counts for each categorical variable (in category order); `NA` for a variable treated as continuous. `NULL` without raw data.
`note`	Explains why the raw-data diagnostics (`per_item`, `normality`, `outliers`, `categories`) are missing, when a correlation matrix is supplied instead of raw data. `NULL` when raw data are supplied.
`settings`	The settings used: `N`, `n_obs` (rows in the raw data supplied, `NA` for a correlation-matrix input), `use`, `cor_method`, `mcd_alpha`, `outlier_cutoff`, and `seed`.

Source

Bartlett, M. S. (1951). The effect of standardization on a Chi-square approximation in factor analysis. Biometrika, 38, 337-344.

Belsley, D. A. (1991). A guide to using the collinearity diagnostics. Computer Science in Economics and Management, 4, 33-50.

Belsley, D. A., Kuh, E. & Welsch, R. E. (1980). Regression diagnostics: Identifying influential data and sources of collinearity. Wiley.

Cochran, W. G. (1954). Some methods for strengthening the common χ^2 tests. Biometrics, 10, 417-451.

Croux, C. & Haesbroeck, G. (1999). Influence function and efficiency of the minimum covariance determinant scatter matrix estimator. Journal of Multivariate Analysis, 71, 161-190.

Field, A. (2018). Discovering statistics using IBM SPSS statistics (5th ed.). Sage.

Henze, N. & Zirkler, B. (1990). A class of invariant consistent tests for multivariate normality. Communications in Statistics - Theory and Methods, 19, 3595-3617.

Kaiser, H. F. (1970). A second generation little jiffy. Psychometrika, 35, 401-415.

Kaiser, H. F. & Rice, J. (1974). Little jiffy, mark IV. Educational and Psychological Measurement, 34, 111-117.

Mardia, K. V. (1970). Measures of multivariate skewness and kurtosis with applications. Biometrika, 57, 519-530.

Mardia, K. V. (1974). Applications of some measures of multivariate skewness and kurtosis in testing normality and robustness studies. Sankhya B, 36, 115-128.

Pison, G., Van Aelst, S. & Willems, G. (2002). Small sample corrections for LTS and MCD. Metrika, 55, 111-123.

Rhemtulla, M., Brosseau-Liard, P. E. & Savalei, V. (2012). When can categorical variables be treated as continuous? A comparison of robust continuous and categorical SEM estimation methods under suboptimal conditions. Psychological Methods, 17, 354-373.

Rousseeuw, P. J. & Van Driessen, K. (1999). A fast algorithm for the minimum covariance determinant estimator. Technometrics, 41, 212-223.

See Also

`efa_kmo()` and `efa_bartlett()` for the individual suitability measures, and `efa_retain()` for factor retention criteria.

Other factor analysis suitability: `efa_bartlett()`, `efa_kmo()`, `print.efa_screen()`

Examples

```
# From a correlation matrix (supply N for Bartlett's test of sphericity)
efa_screen(test_models$baseline$cormat, N = 500)

# From raw data (N is taken from the data; the seed makes the outlier
# diagnostics reproducible)
efa_screen(GRiPS_raw, seed = 1)
```

efa_simulate *Simulate data from a common-factor population model*

Description

Draws data from a population correlation matrix, given either directly or built from a factor model. The population correlation is either supplied in `R`, or assembled from a loading matrix `Lambda`, the factor intercorrelations `Phi`, and the unique variances `Psi` as $R = Lambda\,Phi\,Lambda' + Psi$, standardized to a correlation matrix. `marginals` chooses the marginal distribution of the drawn cases, `categories` discretizes them into ordered categories, `missing` imposes a missing-data mechanism, and a misfit target perturbs the population with model error; see *Details*.

Usage

```
efa_simulate(
  N = NULL,
  Lambda = NULL,
  Phi = NULL,
  Psi = NULL,
  R = NULL,
  model_error = c("CB", "TKL", "WB", "none"),
  target_rmsea = NULL,
  target_cfi = NULL,
  marginals = c("normal", "empirical", "VM", "IG"),
  marginal_data = NULL,
  n_factors = NULL,
  skewness = NULL,
  kurtosis = NULL,
  force_pd = FALSE,
  categories = NULL,
  match = NULL,
  missing = c("none", "MCAR", "MAR", "MNAR"),
  missing_prop = NULL,
  missing_strength = NULL,
  missing_predictor = NULL,
  missing_vars = NULL,
  n_datasets = 1L,
  seed = NULL,
  return_pop = FALSE
)
```

Arguments

`N`	numeric. Number of cases (rows) to draw per dataset. Required unless `return_pop = TRUE`.
`Lambda`	matrix. A `p` by `m` matrix of factor loadings. Supply this (optionally with `Phi` and `Psi`) instead of `R` to build the population from a factor model.

`Phi` matrix. The `m` by `m` factor intercorrelation matrix. Only used with `Lambda`. Default is `NULL`, in which case the factors are orthogonal (an identity matrix).

`Psi` numeric vector or matrix. The unique variances: either a length-`p` vector or a `p` by `p` matrix (added as the residual covariance). Only used with `Lambda`. Default is `NULL`, in which case the unique variances that standardize the population to a correlation matrix are used.

`R` matrix. A `p` by `p` population correlation matrix to draw from directly. Supply this instead of `Lambda`/`Phi`/`Psi`. It must have a unit diagonal; standardize a covariance matrix with `stats::cov2cor()` first.

`model_error` character. The method used to perturb the population so the factor model fits it imperfectly ("model error"): one of `"CB"` (Cudeck-Browne, the default), `"TKL"` (Tucker-Koopman-Linn), `"WB"` (Wu-Browne), or `"none"`. Model error is only applied when a target is supplied in `target_rmsea` or `target_cfi`; without one the population is exact, whatever `model_error`. Only used with a factor-model population (`Lambda`). Note that `efa_power()` defaults to `"TKL"` instead, because its minor common factors degrade factor recovery as well as the fit, so passing the same target to the two functions perturbs the population differently unless `model_error` is set explicitly.

`target_rmsea` numeric. The population RMSEA the factor model should have relative to the perturbed population, a single number strictly in `(0, 1)`. Supplying it activates model error. Simulating from an exact population overstates recovery, so a realistic value (around `0.05`) is recommended for simulation studies (MacCallum, 2003). Default is `NULL` (an exact population; do not pass `0`). Required for `"CB"` and `"WB"`; optional for `"TKL"`.

`target_cfi` numeric. Only used with `model_error = "TKL"`: the population CFI to target, a single number strictly in `(0, 1)`, on its own or together with `target_rmsea` (TKL then trades the two off). Default is `NULL` (do not pass `1`). `"CB"` and `"WB"` target the RMSEA only.

`marginals` character. The marginal distribution of the drawn data: one of `"normal"` (the default), which draws normal marginals; `"empirical"`, which reproduces the population correlation while preserving the empirical marginals supplied in `marginal_data`; or `"VM"` (Vale-Maurelli) and `"IG"` (independent generator), which draw non-normal marginals with the target `skewness` and `kurtosis`.

`marginal_data` matrix or data frame. Only used with `marginals = "empirical"`, where it is required: a data set with one numeric column per variable (`p` columns), each with at least two distinct values, whose per-column distributions are resampled as the marginals of the drawn data. Its correlations are ignored. Default is `NULL`.

`n_factors` numeric. Only used with `marginals = "empirical"`: the number of factors the rank-matching reproduction fits. Default is `NULL`, in which case it is the number of columns of `Lambda` when the population is built from a factor model; it must be given when the population is supplied via `R`.

`skewness` numeric. Only used with `marginals = "VM"` or `"IG"`: the target marginal skewness, as a single value applied to every variable or a length-`p` vector. Default is `NULL` (0, a symmetric marginal). At least one of `skewness` or `kurtosis` must be given for these marginals.

kurtosis — numeric. Only used with marginals = "VM" or "IG": the target marginal excess kurtosis (0 for a normal marginal), as a single value applied to every variable or a length-p vector. Default is NULL (0).

force_pd — logical. Used with marginals = "VM" and with Cudeck-Browne model error (model_error = "CB"), where FALSE (the default) rejects a population or intermediate matrix that is not positive definite. What TRUE accepts instead differs by path, in both cases with a warning: for marginals = "VM" the intermediate correlation matrix is projected to the nearest correlation matrix (via psych::cor.smooth()), and the returned population becomes the one the projected draw attains; for model_error = "CB" there is no such projection – the closest positive-definite perturbation is kept instead, whose achieved RMSEA is **below** the target. Has no effect for the "TKL" or "WB" methods.

categories — numeric or list. Requests ordinal output by discretizing each variable into ordered categories. Either a count of equally probable categories (a single value applied to every variable or a length-p vector), or a length-p list of numeric vectors giving the marginal category proportions per variable (each strictly positive and summing to 1). Default is NULL, which returns the continuous data.

match — character. Only used with categories: an assertion about how the categorization relates to the population correlation. With a normal latent, cutting at the normal-scale thresholds already leaves the population polychoric correlation of the categorized data equal to the target correlation, so both values compute the same thresholds and produce identical data whenever both are legal. "thresholds" (the default) also cuts the "VM" and "IG" draws, whose ordinal Pearson and polychoric correlations then both depart from the population; "polychoric" states that the polychoric match is required and therefore rejects non-normal marginals. Not available with marginals = "empirical". Default is NULL ("thresholds" when categories is set).

missing — character. An optional missing-data mechanism to impose on the drawn data: one of "none" (the default, complete data), "MCAR" (missing completely at random), "MAR" (missing at random, depending on another variable), or "MNAR" (missing not at random, depending on the variable's own value). Introduced values become NA. By default every variable is holed, so under "MAR" each variable's predictor is itself subject to missingness and the mechanism is MAR given the *complete* data rather than ignorably MAR; missing_vars and missing_predictor together give an ignorable design (see Details).

missing_prop — numeric. Only used when missing is not "none", where it is required: the target marginal proportion of missing values per variable, a single number strictly between 0 and 1. This is the expected rate; the realized rate of a given draw varies around it.

missing_strength — numeric. Only used with missing = "MAR" or "MNAR": the slope of the logistic missingness model, setting how strongly the missing probability depends on the predictor. Default is NULL (1, a moderate dependence); 0 removes the dependence (equivalent to MCAR at the same rate) and large magnitudes make missingness nearly deterministic.

missing_predictor — integer or character. Only used with missing = "MAR": which variable drives each holed variable's missingness, as one column index or variable name per

variable in missing_vars (so one per variable by default), in that order. A variable cannot be its own predictor; if the same predictor is reused for several missing_vars, it must not itself be one of them. Default is NULL, in which case each variable's missingness is driven by the next variable cyclically (so variable order matters; supply this explicitly when the order is arbitrary). Predictors outside missing_vars are fully observed (see Details for why this matters).

missing_vars — integer or character. Only used when missing is not "none": which variables carry missing values, as column indices or variable names. Default is NULL, in which case every variable is holed. Restricting it, together with pointing missing_predictor at variables outside that set, is how an ignorably MAR design is obtained (see Details).

n_datasets — numeric. The number of datasets to draw. Default is 1. With more than one, a list of datasets is returned.

seed — numeric. Optional seed for reproducible draws. When supplied, the caller's random-number stream is saved and restored, so the call leaves the global RNG state unchanged. Default is NULL (no seeding).

return_pop — logical. If TRUE, return only the population correlation matrix and draw no data. Default is FALSE. Because no data are drawn, the Vale-Maurelli intermediate correlation matrix is never formed, so its positive-definiteness (and any force_pd projection of it) is neither checked nor reflected in the returned population.

Details

Provide the population either as a ready correlation matrix in R, or through the model components Lambda, Phi, and Psi; the two ways are mutually exclusive. When the model components are used, Phi defaults to the identity matrix (orthogonal factors) and Psi defaults to the unique variances that make the population a correlation matrix ($1 - \mathrm{diag}(Lambda\, Phi\, Lambda')$); the assembled covariance is standardized with stats::cov2cor() so a non-standardized Psi still yields a correlation matrix. With the default Psi, a factor model whose implied communalities exceed 1 (a Heywood case) leaves no unique variance and is rejected; a Psi you supply is instead only required to give positive variances and a positive-semidefinite (a mathematically valid, internally consistent correlation/covariance structure) population. Cases with normal marginals (the default) are drawn through a matrix square root of the population correlation – a Cholesky factor, or a symmetric eigen square root when it is singular (e.g. a communality of exactly 1).

With marginals = "empirical", the iterative rank-matching algorithm of Ruscio and Kaczetow (2008) reproduces the population correlation while each variable takes the empirical marginal distribution of the matching column of marginal_data (resampled with replacement). Only the marginals of marginal_data are used; its own correlations are ignored, and the drawn columns follow the population's variables, not those of marginal_data.

With marginals = "VM" (Vale-Maurelli, 1983) or "IG" (the independent-generator method; Foldnes & Olsson, 2016), the cases reproduce the population correlation while carrying non-normal marginals with the target skewness and (excess) kurtosis. The Vale-Maurelli family does not span every valid non-normal distribution (Foldnes & Grønneberg, 2015); "IG" covers distributions "VM" cannot. Not every (skewness, kurtosis) pair is attainable – every distribution needs excess kurtosis of at least skewness^2 - 2, and either method covers a smaller region still – so an unreachable request

is rejected, as is a "VM" intermediate correlation matrix that is not positive definite unless force_pd allows it.

With categories, the drawn data are discretized into ordered categories (an integer code 1 to K) at the thresholds that reproduce the requested category proportions (Olsson, 1979): the standard-normal quantiles for marginals = "normal", and for marginals = "VM" those quantiles mapped through the same Fleishman (1978) cubic the draw uses, so the requested proportions are reproduced on the non-normal scale too. Under marginals = "IG" the thresholds stay on the standard-normal scale while the data do not, so the achieved proportions depart from the request systematically rather than by sampling noise, and only the *number* of categories is guaranteed; the same holds for a "VM" variable whose Fleishman cubic is not increasing over its own thresholds and the tails beyond them, which keeps the normal-scale thresholds and is reported with a warning. This is more likely with strong skewness/kurtosis or very unequal category proportions; the warning names the affected variable. Because categorization attenuates product-moment correlations, the categorized data's Pearson correlation is smaller in magnitude than the population correlation; under non-normal marginals its polychoric correlation departs from the population as well. Ordinal output is not available with marginals = "empirical". Empty categories left by a draw are reported with a warning, as they destabilize the polychoric correlation and the factor analysis.

With missing, missing values are introduced into the drawn data under a chosen mechanism (Rubin, 1976), each variable holed at a target expected rate missing_prop. "MCAR" draws an independent mask, so missingness is unrelated to the data. "MAR" and "MNAR" set each case's missing probability by a logistic model of a standardized predictor: another variable for "MAR" (chosen by missing_predictor) or the variable's own value for "MNAR", with slope missing_strength. The mechanism acts on the drawn (latent) values, so when categories also discretizes the data the missingness is keyed on the underlying value, not the category code. For "MAR" the predictor is evaluated on the complete drawn values, so whether the mechanism is *ignorably* MAR depends on which variables carry missing values. By default every variable is holed, which leaves a variable's MAR predictor itself missing for roughly a missing_prop fraction of the cases whose missingness it drove. This breaks ignorability: the mechanism is then MAR conditional on the *complete* data but **not** ignorable for an analyst who sees only the observed data. As a result, estimators that are consistent under ignorable MAR – cor_method = "fiml" in efa_fit(), or the multiple imputation behind efa_mi() – keep a residual bias that grows with missing_prop and missing_strength. Restricting the holed variables with missing_vars and pointing missing_predictor at variables outside that set makes every predictor fully observed, which is ignorably MAR and recovers the unbiasedness those estimators are advertised with. The returned matrix carries the NAs, which the correlation estimators handle downstream.

With model_error, the population is perturbed away from the exact factor structure so the q-factor model (q = ncol(Lambda)) fits it only approximately, at a prescribed misfit; exact factor structures are unrealistic (see target_rmsea). The perturbation is applied once to the population, and the achieved misfit of the specified generating model is computed with the same fit-index formulas efa_fit() uses and returned in the model_error element. It needs a factor-model population with residual degrees of freedom and an exact factor structure (a diagonal Psi), and is orthogonal to the marginal, ordinal, and missing-data options. Three methods are available. "CB" (Cudeck & Browne, 1992) matches the target RMSEA to numerical precision and keeps the q-factor model the exact minimizer (the CFI follows as a derived quantity). "TKL" (Tucker, Koopman & Linn, 1969) adds minor common factors tuned so the achieved RMSEA – and, optionally, CFI – match the target(s); with a single target the match is close, with both it is a compromise, reported with a warning when the two cannot be reconciled. "WB" (Wu & Browne, 2015) draws the population from an inverse-Wishart distribution around the model-implied correlation; its calibration applies

to the *best-fitting* model, so the reported misfit of the *generating* model is systematically larger than the target – about 1.4 times for a typical 12-variable, 3-factor model. Use `"CB"` when the reported RMSEA must equal the target.

Replicated draws (`n_datasets` > 1) are generated in parallel across replicates with **future.apply**; a parallel plan can be selected with `future::plan()` (the default plan runs sequentially). Each replicate is assigned its own reproducible random-number stream, so with a fixed `seed` the output is identical regardless of the number of workers.

Value

An object of class `efa_simulated`, a list containing:

`data`	The simulated data: an `N` by `p` numeric matrix, an integer matrix of category codes when `categories` is set, or a length-`n_datasets` list of these when `n_datasets` > 1. `NULL` when `return_pop` = `TRUE`.
`population`	The `p` by `p` population correlation matrix drawn from, model-error-perturbed when requested. When `force_pd` = `TRUE` causes `marginals` = `"VM"`'s intermediate matrix to be projected, `population` is the population the projected draw actually reaches, differing from the target by the drift the warning reports.
`model_error`	`NULL`, or, when model error was applied, a list of the method, the target and achieved RMSEA/CFI, the model degrees of freedom `df`, and the method's tuning parameter: `kappa` for `"CB"`, `v` and `eps` for `"TKL"`, `m` for `"WB"`.
`settings`	The call's key arguments (`N`, `n_datasets`, `marginals`, `categories`, `match`, `missing`, `seed`), kept for reference and used when printing.

Printing the object shows a compact summary.

References

Cudeck, R., & Browne, M. W. (1992). Constructing a covariance matrix that yields a specified minimizer and a specified minimum discrepancy function value. Psychometrika, 57(3), 357-369. doi:10.1007/BF02295424

Fleishman, A. I. (1978). A method for simulating non-normal distributions. Psychometrika, 43(4), 521-532. doi:10.1007/BF02293811

Foldnes, N., & Grønneberg, S. (2015). How general is the Vale-Maurelli simulation approach? Psychometrika, 80(4), 1066-1083. doi:10.1007/s1133601494140

Foldnes, N., & Olsson, U. H. (2016). A simple simulation technique for nonnormal data with prespecified skewness, kurtosis, and covariance matrix. Multivariate Behavioral Research, 51(2-3), 207-219. doi:10.1080/00273171.2015.1133274

MacCallum, R. C. (2003). 2001 Presidential Address: Working with imperfect models. Multivariate Behavioral Research, 38(1), 113-139. doi:10.1207/S15327906MBR3801_5

Olsson, U. (1979). Maximum likelihood estimation of the polychoric correlation coefficient. Psychometrika, 44(4), 443-460. doi:10.1007/BF02296207

Rubin, D. B. (1976). Inference and missing data. Biometrika, 63(3), 581-592. doi:10.1093/biomet/63.3.581

Ruscio, J., & Kaczetow, W. (2008). Simulating multivariate nonnormal data using an iterative algorithm. Multivariate Behavioral Research, 43(3), 355-381. doi:10.1080/00273170802285693

Tucker, L. R., Koopman, R. F., & Linn, R. L. (1969). Evaluation of factor analytic research procedures by means of simulated correlation matrices. Psychometrika, 34(4), 421-459. doi:10.1007/BF02290601

Vale, C. D., & Maurelli, V. A. (1983). Simulating multivariate nonnormal distributions. Psychometrika, 48(3), 465-471. doi:10.1007/BF02293687

Wu, H., & Browne, M. W. (2015). Quantifying adventitious error in a covariance structure as a random effect. Psychometrika, 80(3), 571-600. doi:10.1007/s1133601594513

See Also

`efa_power()`, whose simulation mode draws its replicate data sets with this function, and `efa_fit()` for analysing the simulated data.

Other data simulation: `print.efa_simulated()`

Examples

```
# Build a population from a shipped loading pattern and factor correlations
Lambda <- population_models$loadings$baseline
Phi <- population_models$phis_3$moderate

# Draw one normal dataset of 500 cases (the data live in $data)
sim <- efa_simulate(N = 500, Lambda = Lambda, Phi = Phi, seed = 42)
dim(sim$data)

# Return only the population correlation matrix
R_pop <- efa_simulate(Lambda = Lambda, Phi = Phi, return_pop = TRUE)$population

# Draw several datasets at once from a supplied correlation matrix
sims <- efa_simulate(N = 500, R = R_pop, n_datasets = 3, seed = 42)
length(sims$data)

# Reproduce the population correlation but with skewed, empirical marginals
# (here from a chi-squared source with one column per variable)
src <- matrix(rchisq(200 * nrow(Lambda), df = 3), ncol = nrow(Lambda))
dat_emp <- efa_simulate(N = 500, Lambda = Lambda, Phi = Phi,
                        marginals = "empirical", marginal_data = src, seed = 42)

# Draw skewed, leptokurtic data with the Vale-Maurelli method
dat_vm <- efa_simulate(N = 500, Lambda = Lambda, Phi = Phi, marginals = "VM",
                       skewness = 1.5, kurtosis = 4, seed = 42)

# Draw five-category ordinal data whose polychoric correlation matches R
dat_ord <- efa_simulate(N = 500, Lambda = Lambda, Phi = Phi,
                        categories = 5, match = "polychoric", seed = 42)

# Draw data with 15% missing at random, driven by a neighbouring item
dat_mar <- efa_simulate(N = 500, Lambda = Lambda, Phi = Phi, missing = "MAR",
                        missing_prop = 0.15, seed = 42)
colMeans(is.na(dat_mar$data))

# An ignorably MAR design: only the first nine items are holed, each driven by one of
```

```
# the last nine, which stay complete
dat_ign <- efa_simulate(N = 500, Lambda = Lambda, Phi = Phi, missing = "MAR",
                        missing_prop = 0.15, missing_vars = 1:9,
                        missing_predictor = 10:18, seed = 42)
colMeans(is.na(dat_ign$data))

# Add realistic model error: a population the model fits with RMSEA of about .05
# (Cudeck-Browne, the default method; the achieved fit is reported)
sim_me <- efa_simulate(N = 500, Lambda = Lambda, Phi = Phi,
                       target_rmsea = 0.05, seed = 42)
sim_me$model_error$rmsea
```

`efa_smt` *Sequential chi square model tests, RMSEA lower bound, and AIC*

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

```
efa_smt(
  x,
  N = NA,
  use = c("pairwise.complete.obs", "all.obs", "complete.obs", "everything",
    "na.or.complete"),
  cor_method = c("pearson", "spearman", "kendall", "poly", "tetra"),
  estimate_control = NULL
)
```

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. Must be larger than the number of variables.
`use`	character. Passed to `stats::cor()` if raw data is given as input. Default is "pairwise.complete.obs".

`cor_method` character. One of `"pearson"`, `"spearman"`, or `"kendall"`, passed to `stats::cor()`. `"poly"` and `"tetra"` are not supported because `SMT` rests on a normal-theory chi-square test that is not valid for polychoric / tetrachoric correlations. Default is `"pearson"`.

`estimate_control` an `estimate_control()` object with the estimation settings for the sequential `efa_fit()` fits. `NULL` (default) uses the `efa_fit()` defaults. The sequential models are fitted with maximum likelihood (the chi-square, RMSEA, and AIC the SMT is built on are defined for it), so of the estimation knobs only `start_method` takes effect; the ones governing principal axis factoring do not apply. The models are unrotated, so no rotation settings apply either.

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.

The sequential models are fitted without inequality constraints, so a solution can be inadmissible (a Heywood case, or a fit that did not converge). Only the models the three rules actually select are checked for this; if one of them is inadmissible a warning is raised and the corresponding suggestion should be interpreted with caution.

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).

Value

An object of class `efa_retention` (see `print.efa_retention()` for the print method). SMT has no plot; `plot.efa_retention()` returns `NULL` with a message for it. Its main fields are:

`n_factors` A named numeric vector (`"chi"`, `"RMSEA"`, `"AIC"`) with the suggested number of factors from the sequential chi-square model tests, the RMSEA lower bound, and the AIC.

`results` A list with one record per criterion, each holding the criterion values for the null model (zero factors) through the maximum number of factors.

`settings` A list of the settings used.

Source

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

Browne, M.W., & Cudeck, R. (1992). Alternative ways of assessing model fit. Sociological Methods and Research, 21, 230–258.

Preacher, K. J., Zhang G., Kim, C., & Mels, G. (2013). Choosing the Optimal Number of Factors in Exploratory Factor Analysis: A Model Selection Perspective, Multivariate Behavioral Research, 48(1), 28-56, doi:10.1080/00273171.2012.710386

Steiger, J. H., & Lind, J. C. (1980, May). Statistically based tests for the number of common factors. Paper presented at the annual meeting of the Psychometric Society, Iowa City, IA.

See Also

`efa_retain()` as a wrapper function for this and the other factor retention criteria.

Other factor retention criteria: `efa_cd()`, `efa_ekc()`, `efa_hull()`, `efa_kgc()`, `efa_map()`, `efa_nest()`, `efa_parallel()`, `efa_retain()`, `efa_scree()`

Examples

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

EKC *Empirical Kaiser criterion*

Description

[Superseded]

`EKC()` has been superseded by `efa_ekc()`, which is the recommended interface going forward. It remains available and unchanged so existing code keeps working.

Usage

```
EKC(
  x,
  N = NA,
  use = c("pairwise.complete.obs", "all.obs", "complete.obs", "everything",
    "na.or.complete"),
  cor_method = c("pearson", "spearman", "kendall", "poly", "tetra"),
  type = lifecycle::deprecated()
)
```

Arguments

x	data.frame or matrix. Dataframe or matrix of raw data or matrix with correlations.
N	numeric. The number of observations. Only needed if x is a correlation matrix. Must be larger than the number of variables.
use	character. Passed to stats::cor() if raw data is given as input. Default is "pairwise.complete.obs".
cor_method	character. Correlation computed from raw data: "pearson", "spearman", or "kendall" (passed to stats::cor()), or "poly" / "tetra" for polychoric / tetrachoric correlations of ordinal / binary data (a two-step estimator). Default is "pearson". Note that the EKC reference values rest on the Marchenko-Pastur law for the eigenvalues of a sample correlation matrix of independent variables, which assumes the sampling behaviour of product-moment correlations; with "poly" / "tetra" (and, to a lesser degree, the rank-based methods) the reference series is therefore an approximation.
type	**[Deprecated]** Accepted and ignored. It selected between two ways to compute the reference values. The "AM2019" reference values do not depend on the observed eigenvalues, so they do not apply the empirical correction that defines the criterion, and they are no longer computed.

Value

An object of class efa_retention, identical to the value of efa_ekc(); see there for the components.

See Also

efa_ekc()

---

estimate_control *Control objects for estimation and rotation settings*

Description

estimate_control() and rotate_control() collect the estimation and rotation *tuning* arguments of a factor analysis into two small, validated objects. They are a declarative surface over the same settings resolved internally by the package's estimation and rotation engines, so that a fit's many tuning knobs can be prepared, inspected, and reused as a single value instead of being passed one by one.

Usage

```
estimate_control(
  type = c("EFAtools", "psych", "SPSS", "none"),
  init_comm = NA,
  criterion = NA,
```

```
  criterion_type = NA,
  max_iter = NA,
  abs_eigen = NA,
  start_method = "psych",
  fiml_max_iter = 500,
  fiml_tol = 1e-05
)

rotate_control(
  type = c("EFAtools", "psych", "SPSS", "none"),
  normalize = TRUE,
  precision = 1e-05,
  order_type = NA,
  varimax_type = NA,
  p_type = NA,
  k = NA,
  random_starts = 100,
  ...
)
```

Arguments

type	character. One of "EFAtools" (default), "psych", "SPSS", or "none". Selects the preset that fills the NA-defaulted knobs below when the control is used to fit a model.
init_comm	character. Method for the initial communalities in principal axis factoring: "smc" (squared multiple correlations), "mac" (maximum absolute correlations), or "unity". NA (default) resolves from type.
criterion	numeric. The convergence criterion for principal axis factoring: iteration stops once the change in communalities falls below it. A single number greater than 0 and smaller than 1; NA (default) resolves from type.
criterion_type	character. The convergence criterion type for principal axis factoring: "max_individual" (the largest change in any communality, as in SPSS) or "sum" (the change in the summed communalities, as in psych::fa()). NA (default) resolves from type.
max_iter	numeric. The maximum number of principal-axis-factoring iterations before the procedure is halted with a warning. A single whole number of at least 1; NA (default) resolves from type.
abs_eigen	logical. Which algorithm the principal-axis-factoring iterations use: FALSE computes the loadings from the eigenvalues (as in psych::fa()); TRUE uses the absolute eigenvalues (as in SPSS). NA (default) resolves from type.
start_method	character. Starting values for the maximum-likelihood optimiser: "psych" (default, the psych::fa() starts) or "factanal" (the stats::factanal() starts); abbreviations are matched. Not governed by type. Only maximum likelihood uses it, so NA leaves it unset and is rejected only by a fit that is actually run with estimator = "ML".
fiml_max_iter	numeric. The maximum number of EM iterations used to estimate the two-stage full-information maximum-likelihood moments from raw data with missing val-

ues (`cor_method = "fiml"`); the last iterate is returned, with a warning, if the cap is reached. A single whole number of at least 1; default `500`. Not governed by `type`, and unused by every other correlation method. The EM converges linearly and needs more iterations the larger the fraction of missing information, so raise it when a fit reports that the moments did not converge.

`fiml_tol` numeric. The convergence tolerance of that EM: iteration stops once the largest change in the standardized moments (the standardized means, log-variances, and correlations) falls below it, so it does not depend on the variables' measurement scale. A single number greater than 0 and smaller than 1 (at or above 1 the criterion is met immediately and the starting moments would be returned as converged); default `1e-5`. Not governed by `type`.

`normalize` logical. If `TRUE` (default), a Kaiser normalization is performed before the rotation. The one knob that is always on unless you turn it off with `FALSE`.

`precision` numeric. The convergence tolerance of the rotation procedure. A single number greater than 0 and at most 1; default `1e-5`. Each rotation stage monitors its own quantity, so the same number is not the same tolerance everywhere: `varimax_type = "kaiser"` stops on the *absolute* change in the varimax simplicity criterion, which is an average over variables (and so does not scale with how many there are) but rises with the number of factors, roughly toward `1 - 1 / n_factors`, so a fixed value is a relatively weaker tolerance the more factors are extracted; `varimax_type = "svd"` stops on the *relative* change in the singular values (as in `stats::varimax()`); and the criterion rotations fitted by gradient projection stop when the *projected-gradient norm* falls below it. Promax inherits whichever of the two varimax tests its `varimax_type` selects, because it rotates a varimax base.

`order_type` character. How the factors are ordered: `"eigen"` (by descending sum of squared loadings, as in `psych::fa()`) or `"ss_factors"` (by descending unweighted sum of squared loadings). `NA` (default) resolves from `type`.

`varimax_type` character. The varimax variant used (for the varimax and promax rotations): `"svd"` (as in `stats::varimax()`) or `"kaiser"` (the SPSS / Kaiser (1958) procedure). `NA` (default) resolves from `type`.

`p_type` character. How the promax target matrix is computed: `"unnorm"` (the unnormalized target of Hendrickson & White (1964), also used by psych and stats) or `"norm"` (the normalized target used by SPSS). `NA` (default) resolves from `type`.

`k` numeric. The promax power (for the target matrix) or the number of near-zero loadings for simplimax. A single number greater than 0; `NA` (default) leaves it to the fit (the `type`-dependent promax value, or `nrow(loadings)` for simplimax). Simplimax counts loadings, so a fit using it additionally requires a whole number no larger than the number of loadings in the solution; promax's power has no such restriction.

`random_starts` numeric. The number of random starts used by the criterion-based rotations to guard against local minima. A single whole number of at least 0, where `0` runs the rotation from its warm start only; default `100`. The default suffices for the smooth criteria; `simplimax` remains materially start-dependent at it, so raise it there (see the *Rotations* section of `efa_fit()`).

... Additional arguments forwarded to the rotation engine. Only the names a rotation engine can consume are accepted: `maxit` (a whole number of at least 0 bounding a *single* gradient-projection optimization – the multi-start search runs several of them and each is bounded separately, so it is not a budget for the run as a whole; varimax and promax have no such stage and take only `precision`), and the criterion parameters `gam` (oblimin; `gam = 0` is the recommended default, and larger values increasingly reward correlated factors and can drive the solution toward factor collapse, so inspect `Phi` before interpreting a fit with `gam > 0`) and `delta` (geomin; a positive number, default `0.01`); anything else is rejected as a misspelling. They are stored in `extra_args` and passed on to the rotation engine when the control is used to fit a model; an extra a given fit's rotation does not consume is ignored by that fit, so one control can serve fits with different rotations. An estimation knob (which belongs in `estimate_control()`) or one of the former spellings `P_type` and `randomStarts` is likewise rejected here, because the fit would silently drop it.

Details

Each argument that governs a `type` preset defaults to `NA`, meaning "leave this knob to the preset". Setting `type` to one of `"EFAtools"`, `"psych"`, or `"SPSS"` fills those knobs from the corresponding preset when the fit is run; setting `type = "none"` requires the relevant knobs to be supplied explicitly. The control object only records the chosen `type` and the knobs you set: the preset is resolved (and any "argument set alongside `type`" warning issued) when the object is used to fit a model, exactly as it is today, because which preset applies depends on the estimator and rotation.

Value

`estimate_control()` returns a list of class `efa_estimate_control` with the components `type`, `init_comm`, `criterion`, `criterion_type`, `max_iter`, `abs_eigen`, `start_method`, `fiml_max_iter`, and `fiml_tol`. `rotate_control()` returns a list of class `efa_rotate_control` with the components `type`, `normalize`, `precision`, `order_type`, `varimax_type`, `p_type`, `k`, `random_starts`, and `extra_args` (a named list of any additional arguments forwarded to the rotation engine).

See Also

`efa_fit()`, which takes both controls; `efa_retain()`, the retention criteria, and `efa_schmid_leiman()`, which take an `estimate_control` for the fits they run.

Other Control functions: `print.efa_control`

Examples

```
# Estimation knobs taken entirely from a preset:
estimate_control(type = "SPSS")

# A preset with one knob pinned to a non-preset value:
estimate_control(type = "EFAtools", max_iter = 500)

# Every knob supplied explicitly (type = "none"):
estimate_control(type = "none", init_comm = "smc", criterion = 1e-3,
                 criterion_type = "sum", max_iter = 300, abs_eigen = TRUE)
```

```
# Rotation knobs taken from a preset:
rotate_control(type = "psych")

# A criterion-specific extra argument, forwarded to the rotation engine:
rotate_control(type = "EFAtools", k = 3, gam = 0.5)
```

FACTOR_SCORES *Estimate factor scores for an EFA model*

Description

[Superseded]

`FACTOR_SCORES()` has been superseded by `efa_scores()`, which is the recommended interface going forward. It remains available so existing code keeps working. Note that `R2` is now the squared factor-score determinacy of the *requested* `method`: the squared correlation between a factor and the scores that method produces. For `method = "Thurstone"` this is each factor's squared multiple correlation with the observed variables (the value `psych::factor.scores()` returns with `Grice = TRUE`); for every other method it is smaller, because no estimator correlates more highly with the factor than the regression estimator does. Earlier versions returned psych's default `Grice = FALSE` validity coefficient, so the slot is not comparable across versions.

A convenience wrapper around `efa_scores()` that returns factor scores and weights in a compact list. Factor scores are calculated according to the specified method if raw data are provided, and only factor weights if a correlation matrix is provided.

Usage

```
FACTOR_SCORES(
  x,
  f,
  Phi = NULL,
  rho = NULL,
 method = c("Thurstone", "tenBerge", "Anderson", "Bartlett", "Harman", "components")
)
```

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_fit()` or matrix.
`Phi`	matrix. A matrix of factor intercorrelations. Only needs to be specified if a factor loadings matrix is entered directly into `f`; for an `efa_fit()` object the intercorrelations are taken from the object, and a supplied `Phi` is ignored with a warning. Default is `NULL`, in which case the intercorrelations of a directly supplied loading matrix are assumed to be zero.

rho — matrix. Correlation matrix used to derive the scoring weights. Defaults to NULL, in which case the matrix the EFA in f was fit on (f$orig_R) is used, so the weights stay consistent with the loadings even for a non-Pearson correlation (e.g. polychoric); for a directly supplied loading matrix, x itself is used when it is a correlation matrix, otherwise the Pearson correlation of x. Pass a matrix here to score against a different correlation.

method — character. The method used to calculate factor scores. One of "Thurstone" (regression-based; default), "tenBerge", "Anderson", "Bartlett", "Harman", or "components".

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 — Whether the raw data contained missing values (only if raw data are provided).

R2 — The squared factor-score determinacy for each factor: the squared correlation between a factor and the score the requested method produces. For method = "Thurstone" this equals the squared multiple correlation between the factor and the observed variables; for every other method it is specific to those scores and smaller. See efa_scores() for the underlying score-quality diagnostics.

settings — A list of the settings used.

See Also

efa_scores() for the factor-score weights together with the full set of score-quality diagnostics (determinacy, univocality, and Guttman indeterminacy index) and a print/summary method.

Examples

```
# Example with raw data with method "Bartlett"
EFA_raw <- efa_fit(DOSPERT_raw, n_factors = 10, estimator = "PAF",
                   rotation = "oblimin",
                   rotate_control = rotate_control(random_starts = 1))
fac_scores_raw <- FACTOR_SCORES(DOSPERT_raw, f = EFA_raw, method = "Bartlett")

# Same as above, but with raw data AND a correlation matrix
cor_pearson <- cor(DOSPERT_raw)
EFA_cor_pearson <- efa_fit(cor_pearson, n_factors = 10, N = nrow(DOSPERT_raw),
                           estimator = "PAF", rotation = "oblimin",
                           rotate_control = rotate_control(random_starts = 1))
fac_scores_cor_pearson <- FACTOR_SCORES(DOSPERT_raw, f = EFA_cor_pearson,
                                        rho = cor_pearson,
                                        method = "Bartlett")

# Scores between two alternatives above are identical
isTRUE(all.equal(fac_scores_raw$scores, fac_scores_cor_pearson$scores,
```

```
                  check.attributes = FALSE))

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

`format.efa_retain` *Format method for efa_retain objects*

Description

Format method for efa_retain objects

Usage

```
## S3 method for class 'efa_retain'
format(x, ...)
```

Arguments

`x`	an object of class efa_retain, returned by `efa_retain()`.
`...`	not used.

Value

A character vector with the report lines (styled to the active console theme; plain when colours are disabled).

Examples

```
nf <- efa_retain(test_models$baseline$cormat, criteria = c("EKC", "SMT"),
                 N = 500)
writeLines(format(nf))
```

format.efa_retention *Format method for efa_retention objects*

Description

Format method for efa_retention objects

Usage

```
## S3 method for class 'efa_retention'
format(x, ...)
```

Arguments

x	an object of class efa_retention, returned by a factor-retention criterion (e.g. efa_ekc() or efa_hull()).
...	not used.

Value

A character vector with the report lines (styled to the active console theme; plain when colours are disabled).

Examples

```
writeLines(format(efa_ekc(test_models$baseline$cormat, N = 500)))
```

GRiPS_raw *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/.

Usage

```
GRiPS_raw
```

Format

A data.frame with 810 rows (participants) and 8 columns, one per GRiPS item. Each item is a self-report indicator of general risk propensity, labelled by a keyword from the item:

fun (numeric) - Risk-taking makes life more fun.

friends (numeric) - Friends would describe the respondent as a risk taker.

enjoy (numeric) - Enjoyment of taking risks.

hurt (numeric) - Willingness to take a risk even if it might hurt.

part (numeric) - Risk-taking as an important part of life.

commonly (numeric) - Commonly takes risks.

chances (numeric) - Belief in taking chances.

attracted (numeric) - Attracted, rather than scared, by risk.

Source

Zhang, D. C., Highhouse, S., & Nye, C. D. (2019). Development and validation of the general risk propensity scale (GRiPS). Journal of Behavioral Decision Making, 32, 152–167. doi: 10.1002/bdm.2102

Steiner, M., & Frey, R. (2020). Representative design in psychological assessment: A case study using the Balloon Analogue Risk Task (BART). PsyArXiv Preprint. doi:10.31234/osf.io/dg4ks

---

HULL *Hull method*

Description

[Superseded]

`HULL()` has been superseded by `efa_hull()`, which is the recommended interface going forward. It remains available and unchanged so existing code keeps working.

Usage

```
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", "poly", "tetra"),
  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 efa_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. One plus the larger of this number and the number of factors suggested by efa_parallel is used as the upper bound J of factors to extract in the Hull method.
method	character. The estimator to use; passed to `efa_hull()` as its `estimator` argument. One of `"PAF"`, `"ULS"`, or `"ML"`.
gof	character. The goodness of fit index to use. Either `"CAF"`, `"CFI"`, or `"RMSEA"`, or any combination of them. With the `"PAF"` estimator, 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 `efa_parallel()`.
use	character. Passed to `stats::cor()` if raw data is given as input. Default is `"pairwise.complete.obs"`.
cor_method	character. One of `"pearson"`, `"spearman"`, or `"kendall"`, passed to `stats::cor()`. `"poly"` and `"tetra"` are not supported because `HULL` derives its factor-search bound from an internal parallel analysis against continuous reference data. Default is `"pearson"`.
n_datasets	numeric. The number of datasets to simulate. Must be at least 1. Default is 1000. This is passed to `efa_parallel()`.
percent	numeric. The percentile to take from the simulated eigenvalues. Default is 95. This is passed to `efa_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 `efa_parallel()`.
n_factors	numeric. Number of factors to extract if `"EFA"` is included in `eigen_type`. Default is 1. This is passed to `efa_parallel()`.

... Further arguments passed on to the efa_fit() fits, including the estimation tuning knobs (type, init_comm, criterion, criterion_type, max_iter, abs_eigen, start_method), which are repacked into an estimate_control() object so that they tune the fits exactly as they always did. The estimator is selected with method.

Value

An object of class efa_retention, identical to the value of efa_hull(); see there for the components.

See Also

efa_hull()

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.

CB (numeric) - Crossing out boxes.

NL (numeric) - Numbers / letters.

NLM (numeric) - Numbers / letter mixed.

GF (numeric) - Geometric figures.

RGF (numeric) - Rotated geometric figures.

CM (numeric) - Completing matrices.

EP (numeric) - Excluding pictures.

CA (numeric) - Categories.

OP (numeric) - Opposites.

RS (numeric) - Retelling a story.

DP (numeric) - Describing pictures.

Source

Grieder, S., & Grob, A. (2019). Exploratory factor analyses of the intelligence and development scales–2: Implications for theory and practice. Assessment. Advance online publication. doi:10.1177/1073191119845051

Grob, A., & Hagmann-von Arx, P. (2018). Intelligence and Development Scales–2 (IDS-2). Intelligenz- und Entwicklungsskalen für Kinder und Jugendliche. [Intelligence and Development Scales for Children and Adolescents.]. Bern, Switzerland: Hogrefe.

`KGC` *Kaiser-Guttman criterion*

Description

[Superseded]

`KGC()` has been superseded by `efa_kgc()`, which is the recommended interface going forward. It remains available and unchanged so existing code keeps working.

Usage

```
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", "poly", "tetra"),
  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. Default is `c("PCA", "SMC", "EFA")`, i.e. all three; `"EFA"` is the only one that fits a model.
`use`	character. Passed to `stats::cor()` if raw data is given as input. Default is "pairwise.complete.obs".

cor_method	character. Correlation computed from raw data: "pearson", "spearman", or "kendall" (passed to stats::cor()), or "poly" / "tetra" for polychoric / tetrachoric correlations of ordinal / binary data (a two-step estimator). Default is "pearson".
n_factors	numeric. Number of factors to extract if "EFA" is included in eigen_type. Default is 1.
...	Further arguments passed on to the efa_fit() fit. For example, estimator, to change the estimator (PAF is default), or one of the estimation tuning knobs (type, init_comm, criterion, criterion_type, max_iter, abs_eigen, start_method), which are repacked into an estimate_control() object so that they tune the fit exactly as they always did.

Value

An object of class efa_retention, identical to the value of efa_kgc(); see there for the components.

See Also

efa_kgc()

KMO *Kaiser-Meyer-Olkin criterion*

Description

[Superseded]

KMO() has been superseded by efa_kmo(), which is the recommended interface going forward. It remains available and unchanged so existing code keeps working.

Usage

```
KMO(
  x,
  use = c("pairwise.complete.obs", "all.obs", "complete.obs", "everything",
    "na.or.complete"),
  cor_method = c("pearson", "spearman", "kendall", "poly", "tetra")
)
```

Arguments

x	data.frame or matrix. Dataframe or matrix of raw data or matrix with correlations.
use	character. The missing-data policy for raw data. Passed to stats::cor() for "pearson", "spearman", and "kendall"; for "poly" / "tetra" the same policies are applied to the raw data before the polychoric estimation, where "all.obs" and "everything" abort on a missing value instead of returning NA correlations. Default is "pairwise.complete.obs".

cor_method — character. Correlation computed from raw data: "pearson", "spearman", or "kendall" (passed to stats::cor()), or "poly" / "tetra" for polychoric / tetrachoric correlations of ordinal / binary data (a two-step estimator). Default is "pearson".

Value

A list of class c("efa_kmo", "KMO"), identical to the value of efa_kmo(); see there for the components.

See Also

efa_kmo()

MAP *Minimum average partial*

Description

[Superseded]

MAP() has been superseded by efa_map(), which is the recommended interface going forward. It remains available and unchanged so existing code keeps working.

Usage

```
MAP(
  x,
  use = c("pairwise.complete.obs", "all.obs", "complete.obs", "everything",
    "na.or.complete"),
  cor_method = c("pearson", "spearman", "kendall", "poly", "tetra")
)
```

Arguments

x — A numeric matrix or data.frame. Can be either (a) a correlation matrix, or (b) raw data (rows = observations, columns = variables) from which correlations are computed.

use — Character string specifying the treatment of missing values when computing correlations. Passed to stats::cor(). Defaults to "pairwise.complete.obs".

cor_method — Character string specifying the correlation coefficient to be computed if raw data are supplied. One of "pearson", "spearman", or "kendall" (passed to stats::cor()), or "poly" / "tetra" for polychoric / tetrachoric correlations of ordinal / binary data (a two-step estimator). Defaults to "pearson".

Value

An object of class efa_retention, identical to the value of efa_map(); see there for the components.

See Also

efa_map()

N_FACTORS *Various factor retention criteria*

Description

[Superseded]

N_FACTORS() has been superseded by efa_retain(), which is the recommended interface going forward. It remains available and unchanged so existing code keeps working.

Usage

```
N_FACTORS(
  x,
  criteria = c("CD", "EKC", "HULL", "MAP", "NEST", "PARALLEL"),
  suitability = TRUE,
  N = NA,
  use = c("pairwise.complete.obs", "all.obs", "complete.obs", "everything",
    "na.or.complete"),
  cor_method = c("pearson", "spearman", "kendall", "poly", "tetra"),
  n_factors_max = NA,
  N_pop = 10000,
  N_samples = 500,
  alpha = 0.3,
  max_iter_CD = 50,
  n_fac_theor = NA,
  method = c("ML", "PAF", "ULS"),
  gof = c("CAF", "CFI", "RMSEA"),
  eigen_type_HULL = c("SMC", "PCA", "EFA"),
  eigen_type_other = c("SMC"),
  n_factors = 1,
  n_datasets = 1000,
  percent = 95,
  decision_rule = c("means", "percentile", "crawford"),
  ekc_type = lifecycle::deprecated(),
  n_datasets_nest = 1000,
  alpha_nest = 0.05,
  show_progress = FALSE,
  ...
)
```

Arguments

x	data.frame or matrix. Raw data, or a correlation matrix. 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", "MAP", "NEST", "PARALLEL", "SCREE", and "SMT" (see the details in efa_retain()). By default, a subset of often used, well-performing methods are performed.
suitability	logical. Whether the data should be checked for suitability for factor analysis using Bartlett's test of sphericity and the Kaiser-Meyer-Olkin 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. Correlation computed from raw data: "pearson", "spearman", or "kendall" (passed to stats::cor()), or "poly" / "tetra" for polychoric / tetrachoric correlations (a two-step estimator). CD, PARALLEL, NEST, HULL, and SMT do not support "poly" / "tetra" and are skipped automatically if you request them together. Default is "pearson".
n_factors_max	numeric. Passed to efa_cd(). 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. Passed to efa_cd(). Size of finite populations of comparison data. Default is 10000.
N_samples	numeric. Passed to efa_cd(). Number of samples drawn from each population. Default is 500.
alpha	numeric. Passed to efa_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 efa_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 efa_hull(). Theoretical number of factors to retain. The Hull method uses one plus the larger of this number and the number of factors suggested by efa_parallel() as its upper bound.
method	character. The estimator to use in the criteria that fit EFA models; passed to efa_retain() as its estimator argument. One of "ML", "PAF", or "ULS".
gof	character. Passed to efa_hull(). The goodness of fit index to use. Either "CAF", "CFI", or "RMSEA", or any combination of them. With the "PAF" estimator, 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 efa_parallel() in efa_hull(). What the eigenvalues in the parallel analysis are based on. One of "SMC", "PCA", or "EFA" – different ways of estimating how much variance each indicator shares with the others before the eigenvalues are computed. "SMC" (default) uses each indicator's squared

multiple correlation with the others (its diagonal value in the correlation matrix). "PCA" leaves the diagonal at 1, so each indicator's total variance – not just the shared part – feeds into the eigenvalues. "EFA" uses the communalities from a fitted EFA solution instead.

eigen_type_other
: character. Passed to efa_kgc(), efa_scree(), and efa_parallel(). The same as eigen_type_HULL, but multiple inputs are possible here (any combination of "PCA", "SMC", and "EFA"). Default is "SMC".

n_factors
: numeric. Passed to efa_parallel() (also within efa_hull()), efa_kgc(), and efa_scree(). Number of factors to extract if "EFA" is included in eigen_type_HULL or eigen_type_other. Default is 1.

n_datasets
: numeric. Passed to efa_parallel() (also within efa_hull()). The number of datasets to simulate. Default is 1000.

percent
: numeric. Passed to efa_parallel() (also within efa_hull()). The percentile to take from the simulated eigenvalues. Default is 95.

decision_rule
: character. Passed to efa_parallel() (also within efa_hull()). Which rule to use to determine the number of factors to retain. Default is "means", which uses 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).

ekc_type
: **[Deprecated]** Accepted and ignored. It used to select between two ways to compute the efa_ekc() reference values. The "AM2019" reference values do not depend on the observed eigenvalues. They therefore skip the empirical correction that defines the criterion, so they are no longer computed.

n_datasets_nest
: numeric. Passed to efa_nest(). The number of datasets to simulate. Default is 1000.

alpha_nest
: numeric. Passed to efa_nest(). The alpha level to use. The reference values are the eigenvalues at the (1 - alpha_nest) percentile. Default is .05.

show_progress
: logical. Whether a progress bar should be shown in the console. Default is FALSE.

...
: Further arguments passed on to the efa_fit() fits, including the estimation tuning knobs (type, init_comm, criterion, criterion_type, abs_eigen, start_method), which are repacked into an estimate_control() object so that they tune the fits exactly as they always did. The estimator is selected with method; max_iter is taken by the max_iter_CD argument (R matches an abbreviated name against the arguments before ...) and so does not reach the fits.

Value

A list of class c("efa_retain", "N_FACTORS"), identical to the value of efa_retain(); see there for the components.

See Also

efa_retain()

NEST *Next eigenvalue sufficiency test*

Description

[Superseded]

`NEST()` has been superseded by `efa_nest()`, which is the recommended interface going forward. It remains available and unchanged so existing code keeps working.

Usage

```
NEST(
  x,
  N = NA,
  alpha = 0.05,
  use = c("pairwise.complete.obs", "all.obs", "complete.obs", "everything",
    "na.or.complete"),
  cor_method = c("pearson", "spearman", "kendall", "poly", "tetra"),
  n_datasets = 1000,
  ...
)
```

Arguments

`x`	data.frame or matrix. Dataframe or matrix of raw data or matrix with correlations.
`N`	numeric. The number of observations. Only needed if x is a correlation matrix. Must be larger than the number of variables.
`alpha`	numeric. The alpha level to use (i.e., 1-alpha percentile of eigenvalues is used for reference values).
`use`	character. Passed to `stats::cor()` if raw data is given as input. Default is "pairwise.complete.obs".
`cor_method`	character. One of `"pearson"`, `"spearman"`, or `"kendall"`, passed to `stats::cor()`. `"poly"` and `"tetra"` are not supported because `NEST` compares the data against simulated continuous reference data. Default is `"pearson"`.
`n_datasets`	numeric. The number of datasets to simulate. Default is 1000.
`...`	Further arguments passed on to the `efa_fit()` fits. For example, `estimator`, to change the estimator (PAF is default), or one of the estimation tuning knobs (`type`, `init_comm`, `criterion`, `criterion_type`, `max_iter`, `abs_eigen`, `start_method`), which are repacked into an `estimate_control()` object so that they tune the fits exactly as they always did.

Value

An object of class `efa_retention`, identical to the value of `efa_nest()`; see there for the components.

See Also

efa_nest()

OMEGA *McDonald's omega*

Description

[Superseded]

OMEGA() has been superseded by efa_reliability(), which is the recommended interface going forward. It remains available and unchanged so existing code keeps working.

This function finds omega total, hierarchical, and subscale, as well as additional model-based indices of interpretive relevance (H index, ECV, PUC) 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(), efa_schmid_leiman(), or, in a more flexible way, by leaving model = NULL and specifying additional arguments. The type argument selects how variables are assigned to group factors, and can reproduce the assignment psych::omega() makes.

Usage

```
OMEGA(
  model = NULL,
  type = c("EFAtools", "psych"),
  g_name = "g",
  group_names = NULL,
  add_ind = TRUE,
  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 efa_schmid_leiman(), class schmid, or class lavaan object. That is, an output object from efa_schmid_leiman() 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 additionally. If of class efa_schmid_leiman() 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.

`add_ind` logical. Whether additional indices (H index, ECV, PUC) should be calculated or not (see details for these indices). If FALSE, only omegas are returned. Default is `TRUE`.

`factor_corres` matrix. A logical matrix or a numeric matrix containing 0's and 1's that indicates 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 `efa_schmid_leiman()` output is entered in `model`, the correlation matrix is taken from the output. If left `NULL` and a `psych::schmid()` output is entered, the correlation 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

What this function does:

All types of McDonald's omegas (total, hierarchical, and subscale; McDonald, 1978, 1985, 1999) are calculated for the general factor as well as for the subscales / group factors (see, e.g., Gignac, 2014; Rodriguez et al., 2016a, 2016b). Omegas refer to the correlation between a factor and a unit-weighted composite score and thus the true score variance in a unit-weighted composite based on the respective indicators. Omega total is the total true score variance in a composite. Omega hierarchical is the true score variance in a composite that is attributable to the general factor, and omega subscale is the true score variance in a composite attributable to all subscales / group factors (for the whole scale) or to the specific subscale / group factor (for subscale composites).

Accordingly, on a subscale row the `hier` column reports the share of that subscale's composite variance due to the general factor and the `sub` column the share due to the subscale-specific factor; the latter corresponds to the omega hierarchical subscale of Rodriguez et al. (2016a, 2016b).

The H index (also construct reliability or replicability index) is the correlation between an optimally-weighted composite score and a factor (Hancock & Mueller, 2001; Rodriguez et al., 2016a, 2016b). It, too, can be calculated for the whole scale / general factor as well as for the subscales / group factors. Low values indicate that a latent variable is not well defined by its indicators.

The ECV (Rodriguez et al., 2016a, 2016b) is the ratio of the variance explained by the general factor and the variance explained by the general factor and the group factors.

The PUC (Bonifay et al., 2015; Reise et al., 2013, Rodriguez et al., 2016a, 2016b) refers to the proportion of correlations in the underlying correlation matrix that is not contaminated by variance of both the general factor and the group factors (i.e., correlations between indicators from different group factors, which reflect only general factor variance). The higher the PUC, the more similar a general factor from a multidimensional model will be to the single factor from a unidimensional model.

How to use this function:

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 with one second-order factor assumed) or a breadth factor (bifactor model assumed). Please note that this function only works for second-order models if they contain no more than one second-order factor. 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 `efa_schmid_leiman()` 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 indices are computed for each group. For `lavaan` input the composite variances entering the omegas are model-implied: they are computed from the fitted loadings and the fitted residual covariance matrix, and count any freed residual covariance as well as the residual variances. The coefficients thus coincide with the observed-score versions when the model fits perfectly. The omegas split a composite's variance into a general part and one part per group factor, which needs uncorrelated latent variables: fit a bifactor model with `orthogonal = TRUE` (not `lavaan`'s default) and leave the covariances between a second-order model's first-order factors at zero. A fit whose factors correlate is rejected rather than scored as though they did not. The type argument is not evaluated if `model` is of class `lavaan`.

If `model` is of class `efa_schmid_leiman()` 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" derives them as psych::omega() does, 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.

Both settings score a composite by the true score variance the model attributes to it, counting every factor its variables load on; they differ only in the variance that is divided into. variance = "correlation" uses the composite's observed variance, giving the observed-score form of omega; "sums_load" uses its model-implied variance, which partitions exactly into omega hierarchical plus omega subscale on the whole-scale row. The two settings agree up to model misfit, and differ mainly in the whole-scale omega subscale, which counts all group-factor variance under "sums_load" but only the assigned subscale composites under "correlation".

Value

If found for an SL or lavaan second-order or bifactor solution without multiple groups: A matrix with omegas for the whole scale and for the subscales and (only if add_ind = TRUE) with the H index, ECV, and PUC.

tot	Omega total.
hier	Omega hierarchical.
sub	Omega subscale.
H	H index.
ECV	Explained common variance.
PUC	Percent of uncontaminated correlations.

If found for a lavaan single factor solution without multiple groups: A (named) vector with omega total and (if add_ind = TRUE) the H index 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

McDonald, R. P. (1978). Generalizability in factorable domains: "Domain validity and generalizability". Educational and Psychological Measurement, 38, 75–79.

McDonald, R. P. (1985). Factor analysis and related methods. Hillsdale, NJ: Erlbaum.

McDonald, R. P. (1999). Test theory: A unified treatment. Mahwah, NJ: Erlbaum.

Rodriguez, A., Reise, S. P., & Haviland, M. G. (2016a). Applying bifactor statistical indices in the evaluation of psychological measures. Journal of Personality Assessment, 98, 223-237.

Rodriguez, A., Reise, S. P., & Haviland, M. G. (2016b). Evaluating bifactor models: Calculating and interpreting statistical indices. Psychological Methods, 21, 137-150.

Hancock, G. R., & Mueller, R. O. (2001). Rethinking construct reliability within latent variable systems. In R. Cudeck, S. du Toit, & D. Sörbom (Eds.), Structural equation modeling: Present and future—A Festschrift in honor of Karl Jöreskog (pp. 195–216). Lincolnwood, IL: Scientific Software International.

Reise, S. P., Scheines, R., Widaman, K. F., & Haviland, M. G. (2013). Multidimensionality and structural coefficient bias in structural equation modeling: A bifactor perspective. Educational and Psychological Measurement, 73, 5–26.

Bonifay, W. E., Reise, S. P., Scheines, R., & Meijer, R. R. (2015). When are multidimensional data unidimensional enough for structural equation modeling?: An evaluation of the DETECT multidimensionality index. Structural Equation Modeling, 22, 504—516.

Gignac, G. E. (2014). On the Inappropriateness of Using Items to Calculate Total Scale Score Reliability via Coefficient Alpha for Multidimensional Scales. European Journal of Psychological Assessment, 30, 130-139.

See Also

`efa_reliability()` for the same coefficients in a tidy, long-format result.

Examples

```
## Use with lavaan outputs
if (requireNamespace("lavaan", quietly = TRUE)) {

# 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 omegas and additional indices for bifactor solution
OMEGA(fit_bi, g_name = "g")

# Compute only omegas
OMEGA(fit_bi, g_name = "g", add_ind = FALSE)

# 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 omegas and additional indices for second-order solution
OMEGA(fit_ho, g_name = "g")
```

```
}


## Use with an output from the SL function, with type EFAtools
efa_mod <- efa_fit(test_models$baseline$cormat, N = 500, n_factors = 3,
                   estimator = "PAF", rotation = "promax")
sl_mod <- efa_schmid_leiman(efa_mod, estimator = "PAF")

# Indicator-to-factor correspondences from a salience threshold (here: .20):
factor_corres_1 <- sl_mod$sl[, c("F1", "F2", "F3")] >= .2

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

## Use with an output from the psych::schmid function, with type psych for
## OMEGA
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
OMEGA(schmid_mod, type = "psych")
# Use specified correlation matrix
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.

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)

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)
```

PARALLEL *Parallel analysis*

Description

[Superseded]

PARALLEL() has been superseded by efa_parallel(), which is the recommended interface going forward. It remains available and unchanged so existing code keeps working.

Usage

```
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", "poly", "tetra"),
  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. Must be larger than the number of variables.
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. Must be at least 1. 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. Default is c("PCA", "SMC", "EFA"), i.e. all three, which costs roughly six times a single non-EFA type: "EFA" fits an EFA to every simulated dataset and dominates that total. Pass a single type if the run is time-critical.
use	character. Passed to stats::cor() if raw data is given as input. Default is "pairwise.complete.obs".
cor_method	character. One of "pearson", "spearman", or "kendall", passed to stats::cor(). "poly" and "tetra" are not supported because PARALLEL compares the data against simulated continuous reference data. 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). All three rules retain the factors up to the first observed eigenvalue that fails to exceed its reference value; an eigenvalue further down the series that rises above

its own reference again therefore adds no factor. Because the average simulated eigenvalue is a lower reference than the percentile, `"means"` tends to retain more factors than the more conservative `"percentile"` rule (Glorfeld, 1995).

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

`...` Further arguments passed on to the `efa_fit()` fits. For example, `estimator`, to change the estimator (default is "PAF"; PAF is more robust, but it will take longer compared to "ML" and "ULS"), or one of the estimation tuning knobs (`type`, `init_comm`, `criterion`, `criterion_type`, `max_iter`, `abs_eigen`, `start_method`), which are repacked into an `estimate_control()` object so that they tune the fits exactly as they always did.

Value

An object of class `efa_retention`, identical to the value of `efa_parallel()`; see there for the components.

See Also

`efa_parallel()`

`plot.efa_average` *Plot efa_average object*

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.

Value

A ggplot object showing, for each indicator and factor, the minimum, maximum, and average (mean or median) loading across the averaged solutions. Each panel carries a point at the average, a bar spanning the minimum to the maximum with a tick at each endpoint, and a grey band marking the loadings that fall below the salience threshold; the caption names the four marks.

Examples

```
EFA_aver <- efa_average(test_models$baseline$cormat, n_factors = 3, N = 500)
plot(EFA_aver)
```

`plot.efa_compare` *Plot efa_compare object*

Description

Plot method for the `efa_compare()` function showing the distribution of the absolute differences between the two compared objects as a violin plot with jittered points. Differences above the threshold are highlighted.

Usage

```
## S3 method for class 'efa_compare'
plot(x, plot_red = NULL, ...)
```

Arguments

`x`	list. An object of class `efa_compare` (output from the `efa_compare()` function).
`plot_red`	numeric or `NULL`. Threshold above which to draw the absolute differences in red, documented in `efa_compare()`. `NULL` (default) uses the value recorded in `x$settings`; supplying one overrides it for this plot only, so the comparison need not be recomputed to redraw it at another threshold.
`...`	not used.

Value

A ggplot object showing the absolute differences, with differences above `plot_red` highlighted in red.

Examples

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

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

# compare the two and plot the differences
```

```
comp <- efa_compare(EFA_SPSS_5$unrot_loadings, EFA_psych_5$unrot_loadings,
                    x_labels = c("SPSS", "psych"))
plot(comp)
```

`plot.efa_group` *Plot a multigroup factor analysis*

Description

Two views of an `efa_group()` result, selected by `type`:

Usage

```
## S3 method for class 'efa_group'
plot(x, type = c("congruence", "differences"), ...)
```

Arguments

`x`	An object of class `efa_group` (output from `efa_group()`).
`type`	character. Which plot to draw: `"congruence"` (per-factor congruence with confidence intervals) or `"differences"` (a per-item loading-difference heatmap).
`...`	Not used; for consistency with the generic.

Details

- `"congruence"` (the default) plots the matched Tucker congruence of each factor between every group pair, with a percentile bootstrap confidence interval when one was computed (`b_boot > 0`). The Lorenzo-Seva and ten Berge (2006) reference bands (`.95` "equal", `.85` "fair") are drawn so a factor's cross-group similarity can be read against them at a glance.
- `"differences"` draws a heatmap of the signed cross-group loading differences (item by factor, one panel per group pair). Cells whose absolute difference reaches the salience threshold `delta` are outlined.

Value

A ggplot2::ggplot object.

References

Lorenzo-Seva, U., and ten Berge, J. M. F. (2006). Tucker's congruence coefficient as a meaningful index of factor similarity. *Methodology*, 2, 57-64. doi: 10.1027/1614-2241.2.2.57

See Also

Other factor analysis: `efa_average()`, `efa_fit()`, `efa_group()`, `efa_mi()`, `print.efa_group()`

Examples

```
g <- rep(c("g1", "g2"), length.out = nrow(GRiPS_raw))
mg <- efa_group(GRiPS_raw, groups = g, n_factors = 1)

# Per-factor congruence against the Lorenzo-Seva & ten Berge bands
plot(mg)

# Per-item cross-group loading-difference heatmap
plot(mg, type = "differences")
```

`plot.efa_power` *Plot the RMSEA power curve*

Description

Draws the analytic RMSEA power (MacCallum, Browne, & Sugawara, 1996) of an `efa_power()` result as a function of the total sample size, mirroring `semTools::plotRMSEApower()` but returning a ggplot2::ggplot object rather than drawing to the active device. The test, its null and alternative RMSEA, the significance level, and the number of groups are taken from the object; only the sample-size axis is swept, with an optional sweep of the degrees of freedom or the alternative RMSEA to overlay several curves.

Usage

```
## S3 method for class 'efa_power'
plot(x, n = NULL, df = NULL, eps1 = NULL, ...)
```

Arguments

`x`	An object of class `efa_power` (output from `efa_power()`).
`n`	numeric. The total sample sizes to evaluate. If `NULL` (the default) a sequence bracketing the object's sample size is chosen automatically.
`df`	numeric. The model degrees of freedom (must be positive). Defaults to the object's `df`; a vector of length greater than one draws one curve per value.
`eps1`	numeric. The alternative-hypothesis RMSEA (must differ from the null `eps0`). Defaults to the object's `eps1`; a vector of length greater than one draws one curve per value. At most one of `df` and `eps1` may be a vector.
`...`	Not used; for consistency with the generic.

Details

When the plotted curve is the object's own – a single curve with neither `df` nor `eps1` overridden – it is annotated with the object's result: a dashed vertical line at its sample size `x$N`, a dashed horizontal line at the reference power (the target power when a sample size was solved for, otherwise the power achieved at `x$N`), and a point at `x$N` and the achieved power. Overriding `df` or `eps1`, sweeping either as a vector, or supplying an `n` that does not span `x$N` moves that point off the drawn curve, so the marks are then omitted.

Value

A ggplot2::ggplot object.

References

MacCallum, R. C., Browne, M. W., & Sugawara, H. M. (1996). Power analysis and determination of sample size for covariance structure modeling. *Psychological Methods, 1*(2), 130-149. doi:10.1037/1082989X.1.2.130

See Also

Other power analysis: `efa_power()`, `print.efa_power()`

Examples

```
pw <- efa_power(df = 100, N = 200)

# Power curve for the test of close fit, marking the object's own N
plot(pw)

# Overlay several models by sweeping the degrees of freedom
plot(pw, df = c(50, 100, 200))

# Sweep the alternative RMSEA instead
plot(pw, eps1 = c(0.06, 0.08, 0.10))
```

---

plot.efa_retain *Plot method for efa_retain objects*

Description

Plots every factor-retention criterion in the `efa_retain()` result that has a plottable outcome (see `plot.efa_retention()`). Criteria without a plot (e.g. `efa_map()` or `efa_smt()`) are skipped.

Usage

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

Arguments

x	an object of class efa_retain, returned by `efa_retain()`.
...	not used.

Value

A named list of ggplot2::ggplot objects, one per criterion with a plottable result, or invisibly `NULL` if there is none.

Examples

```
nf <- efa_retain(test_models$baseline$cormat, criteria = c("EKC", "SMT"),
                 N = 500)
plot(nf)
```

`plot.efa_retention` *Plot method for efa_retention objects*

Description

Plots the result of a factor-retention criterion. Eigenvalue-based criteria (e.g. `efa_ekc()`) are shown as an eigenvalue plot, the Hull method (`efa_hull()`) as a convex-hull plot. Criteria with more than one sub-variant are faceted.

Usage

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

Arguments

`x`	an object of class efa_retention, returned by a factor-retention criterion (e.g. `efa_ekc()` or `efa_hull()`).
`...`	not used.

Value

A ggplot2::ggplot object, or invisibly `NULL` if the criterion has no plottable result.

Examples

```
plot(efa_ekc(test_models$baseline$cormat, N = 500))
```

`population_models` *population_models*

Description

Population factor models, some of which (baseline to case_11e) used for the simulation analyses reported in Grieder and Steiner (2022). 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).

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, and .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). `phis_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). `phis_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

Grieder, S., & Steiner, M. D. (2022). Algorithmic jingle jungle: A comparison of implementations of principal axis factoring and promax rotation in R and SPSS. Behavior Research Methods, 54, 54–74. doi: 10.3758/s13428-021-01581-x

de Winter, J.C.F., & Dodou, D. (2012). Factor recovery by principal axis factoring and maximum likelihood factor analysis as a function of factor pattern and sample size. Journal of Applied Statistics. 39.

`print.efa` *Print and summarise an efa object*

Description

`print()` shows a concise overview of an `efa_fit()` or `efa_mi()` solution: a model header, the loading matrix (with the factor intercorrelations for oblique solutions), the variances accounted for, and the model fit. `summary()` returns a `summary.efa` object whose print method adds the full diagnostics: model and simple-structure diagnostics, confidence-interval tables, the structure matrix, multiple-imputation uncertainty (for pooled objects), and residual diagnostics. `format()` assembles the same report and returns it as a character vector; `print()` is `cat(format(x), sep = "\n")`. The lines follow the active console theme, so they are plain when colours are disabled (for example when captured into a file or stripped with `cli::ansi_strip()`).

Usage

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

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

## S3 method for class 'efa'
format(
  x,
  cutoff = 0.3,
  digits = 3,
  max_name_length = 10,
  sort_loadings = c("none", "primary", "clustered"),
  show_loading_legend = TRUE,
  max_factors_per_block = NULL,
  ...
)

## S3 method for class 'efa_mi'
format(x, ...)

## S3 method for class 'efa'
```

```
summary(
  object,
  cutoff = 0.3,
  digits = 3,
  max_name_length = 10,
  ci = c("auto", "none", "separate"),
  ci_filter = c("salient", "all", "nonzero"),
  diagnostics_top_n = 10,
  residual_cutoff = 0.1,
  residual_top_n = 10,
  show_structure = TRUE,
  sort_loadings = c("none", "primary", "clustered"),
  show_loading_legend = TRUE,
  cross_loading_cutoff = cutoff,
  min_primary_gap = 0.2,
  min_salient_per_factor = 3,
  max_factors_per_block = NULL,
  show_mi_diagnostics = NULL,
  ...
)

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

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

## S3 method for class 'summary.efa'
format(x, ...)
```

Arguments

x, object	An object of class efa (from efa_fit()) or efa_mi (from efa_mi()); for the summary.efa methods, the object returned by summary().
...	Further arguments passed to print.efa_loadings().
cutoff	numeric. The absolute value at or above which loadings are emphasised in the loading table. Default is .3.
digits	numeric. Number of decimal places for the printed tables. Default is 3.
max_name_length	numeric. Maximum length of the variable names to display; longer names are cut from the right, or abbreviated where cutting would give two variables the same label. Applies to every table that names variables. name_style (see print.efa_loadings()) can be passed through ... to choose the shortening explicitly, but it reaches the loading table only; the confidence-interval and simple-structure tables always shorten by cutting.
sort_loadings	character. Optional row sorting for the loading table. See print.efa_loadings().

show_loading_legend
: logical. Whether to print a short legend for the loading-table styling. Default is TRUE. The legend is only printed when that styling is actually rendered (a colour-capable console); in plain output it is omitted and this argument has no effect.

max_factors_per_block
: numeric or NULL. Maximum number of factor columns per loading-table block. If NULL, chosen from the console width.

ci
: character. Which confidence intervals summary() shows, if available. "auto" and "separate" print CI sections when CIs were computed; "none" suppresses them. Default is "auto".

ci_filter
: character. Which loading CIs summary() prints: "salient" (default), "all", or "nonzero"; see Details.

diagnostics_top_n
: numeric. Maximum number of item-level entries summary() prints per simple-structure diagnostic.

residual_cutoff
: numeric. Absolute residual cutoff for the residual diagnostics in summary(). Default is .1.

residual_top_n
: numeric. Maximum number of residuals summary() prints. Use Inf to print all residuals above residual_cutoff.

show_structure
: logical. Whether summary() prints the structure matrix for oblique solutions when available. Default is TRUE.

cross_loading_cutoff
: numeric. Cutoff for counting cross-loadings in the summary() diagnostics. Defaults to cutoff.

min_primary_gap
: numeric. Minimum desired absolute difference between the largest and second-largest absolute loading of an item, used in the summary() diagnostics.

min_salient_per_factor
: numeric. Minimum number of salient indicators per factor used in the summary() diagnostics. Default is 3.

show_mi_diagnostics
: logical or NULL. Whether summary() prints a multiple-imputation uncertainty summary for pooled EFAs. NULL shows it for pooled objects.

Details

The methods are shared by single-imputation efa objects and pooled efa_mi objects. For efa_mi objects the header reports the number of imputations and the alignment/pooling settings; confidence intervals and a multiple-imputation uncertainty summary are shown by summary() when the pooled object carries bootstrap/MI quantities.

In summary(), ci_filter controls which loading intervals are shown: "salient" reports intervals for loadings whose absolute point estimate is at least cutoff, "nonzero" reports intervals excluding zero, and "all" reports every finite interval.

Value

`print()` and the print method for `summary.efa` objects return their argument invisibly. `format()` returns a character vector with the report lines. `summary()` returns an object of class `summary.efa`.

Examples

```
mod <- efa_fit(test_models$baseline$cormat, n_factors = 3, N = 500,
               estimator = "PAF", rotation = "promax")
mod

# The full diagnostics, CI tables, and residual diagnostics:
summary(mod)

# format() returns the same lines as a character vector, e.g. for a report file:
writeLines(format(mod))
```

`print.efa_average` *Print and format an efa_average object*

Description

`print()` shows a summarised output of the `efa_average()` function: the averaging settings, the error/convergence/Heywood/admissibility rates, the indicator-to-factor correspondences, the averaged loadings (and, for oblique solutions, the factor intercorrelations), the variances accounted for, and the model fit. `format()` assembles the same report and returns it as a character vector; by default (`plot = FALSE`) `print()` is `cat(format(x), sep = "\n")`. With `plot = TRUE` it additionally draws the loading plot, which is the one thing `format()` cannot return: the printed lines are the same, but the call has a side effect beyond them. The lines follow the active console theme, so they are plain when colours are disabled (for example when captured into a file or stripped with `cli::ansi_strip()`).

Usage

```
## S3 method for class 'efa_average'
print(x, stat = c("average", "range"), plot = FALSE, ...)

## S3 method for class 'efa_average'
format(x, stat = c("average", "range"), ...)
```

Arguments

`x`	An object of class `efa_average` (output from `efa_average()`).
`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 FALSE. If more than 10 factors are extracted, no plot is created. Only used by print(); plot.efa_average() draws the same plot and returns it, so it is the route to take when the plot object itself is wanted.
...	Not used; for consistency with the generic.

Value

print() returns its argument x invisibly. format() returns a character vector with the report lines.

Examples

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

# format() returns the same lines as a character vector:
writeLines(format(EFA_aver))
```

---

print.efa_bartlett *Print and format an efa_bartlett object*

Description

print() reports the outcome of efa_bartlett()'s test of sphericity: a verdict on whether the test was significant (and what that implies for the suitability of the data for factor analysis), followed by the chi-square statistic, its degrees of freedom, and the p-value. format() assembles the same report and returns it as a character vector; print() is cat(format(x), sep = "\n"). The lines follow the active console theme, so they are plain when colours are disabled (for example when captured into a file or stripped with cli::ansi_strip()).

Usage

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

## S3 method for class 'efa_bartlett'
format(x, ...)
```

Arguments

x	An object of class efa_bartlett (output from efa_bartlett()).
...	Not used; for consistency with the generic.

Value

print() returns its argument x invisibly. format() returns a character vector with the report lines.

Examples

```
bart <- efa_bartlett(test_models$baseline$cormat, N = 500)
bart

# format() returns the same lines as a character vector:
writeLines(format(bart))
```

`print.efa_compare` *Print and format an efa_compare object*

Description

`print()` shows a summarised output of the `efa_compare()` function: the mean (with its range), median, and root mean squared distance (RMSE) of the differences, the number of decimals to which all numbers agree, the minimum number of decimals provided, and (for matrices) the number of differing indicator-to-factor correspondences, followed (optionally) by the table of elementwise differences. `format()` assembles the same report and returns it as a character vector; `print()` is `cat(format(x), sep = "\n")`. The lines follow the active console theme, so they are plain when colours are disabled (for example when captured into a file or stripped with `cli::ansi_strip()`).

Usage

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

## S3 method for class 'efa_compare'
format(
  x,
  digits = NULL,
  m_red = NULL,
  range_red = NULL,
  round_red = NULL,
  print_diff = NULL,
  ...
)
```

Arguments

`x`	An object of class `efa_compare` (output from `efa_compare()`).
`...`	Passed from `print()` to `format()`; not otherwise used.
`digits, m_red, range_red, round_red, print_diff`	Display controls, documented in `efa_compare()`. Each defaults to `NULL`, meaning the value `efa_compare()` recorded in `x$settings` is used; supplying one overrides it for this call only, so the comparison need not be recomputed to change the printed report.

Details

The line reporting the minimum number of decimals provided is shown only when it carries information: two ordinary double matrices carry the full double precision, for which the count is uninformative and the line is omitted.

The summary statistics are absolute differences, so they carry no direction. The elementwise differences are signed, and the table is headed by the direction of the subtraction, named with the `x_labels` recorded by `efa_compare()` (`"x"` and `"y"` by default): a negative cell means the first solution is the lower of the two there.

Value

`print()` returns its argument `x` invisibly. `format()` returns a character vector with the report lines.

Examples

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

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

# compare the two
comp <- efa_compare(EFA_SPSS_5$unrot_loadings, EFA_psych_5$unrot_loadings,
                    x_labels = c("SPSS", "psych"))
comp

# format() returns the same lines as a character vector:
writeLines(format(comp))

# the display settings can be changed without recomputing the comparison:
print(comp, digits = 2, print_diff = FALSE)
```

`print.efa_control` *Print and format a control object*

Description

`print()` shows the chosen `type` and each tuning knob, with an unset (`NA`) preset-driven knob marked as resolved from the `type` preset. `format()` assembles the same report and returns it as a character vector; `print()` is `cat(format(x), sep = "\n")`. The lines follow the active console theme, so they are plain when colours are disabled.

Usage

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

## S3 method for class 'efa_estimate_control'
format(x, ...)

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

## S3 method for class 'efa_rotate_control'
format(x, ...)
```

Arguments

x	A control object from estimate_control() or rotate_control().
...	Not used; for consistency with the generic.

Value

print() returns its argument x invisibly. format() returns a character vector with the report lines.

See Also

estimate_control(), rotate_control()

Other Control functions: estimate_control()

Examples

```
est <- estimate_control(type = "SPSS")
est
writeLines(format(est))
```

---

print.efa_group *Print and format a multigroup factor analysis*

Description

print() turns an efa_group() result into a sectioned report: a header recapping the groups, the common number of factors, the estimator, the rotation, and the alignment; a group-pair table of the matched Tucker congruences between the aligned loadings; a per-pair summary of the cross-group loading differences (with the salient and, when a bootstrap was run, the confidence-interval flags); and, when invariance = TRUE, a group-pair by factor grid of the approximate-invariance verdicts. format() assembles the same report and returns it as a character vector; print() is cat(format(x), sep = "\n"). The lines follow the active console theme, so they are plain when colours are disabled (for example when captured into a file or stripped with cli::ansi_strip()). print() does not draw a plot; use plot.efa_group().

Usage

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

## S3 method for class 'efa_group'
format(x, digits = 3, ...)
```

Arguments

x	An object of class efa_group (output from efa_group()).
digits	Integer. The number of decimal places the reported values are rounded to. Default is 3.
...	Not used; for consistency with the generic.

Value

print() returns its argument x invisibly. format() returns a character vector with the report lines.

See Also

Other factor analysis: efa_average(), efa_fit(), efa_group(), efa_mi(), plot.efa_group()

Examples

```
g <- rep(c("g1", "g2"), length.out = nrow(GRiPS_raw))
mg <- efa_group(GRiPS_raw, groups = g, n_factors = 1)
mg

# format() returns the same lines as a character vector:
writeLines(format(mg))
```

---

print.efa_kmo *Print and format an efa_kmo object*

Description

print() shows the Kaiser-Meyer-Olkin (KMO) criterion computed by efa_kmo(): a titled section with a verdict on the overall KMO value (and what it implies for the suitability of the data for factor analysis), the overall value, and the per-variable KMO values. format() assembles the same report and returns it as a character vector; print() is cat(format(x), sep = "\n"). The lines follow the active console theme, so they are plain when colours are disabled (for example when captured into a file or stripped with cli::ansi_strip()).

Usage

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

## S3 method for class 'efa_kmo'
format(x, ...)
```

Arguments

x	An object of class efa_kmo (output from efa_kmo()).
...	Not used; for consistency with the generic.

Value

print() returns its argument x invisibly. format() returns a character vector with the report lines.

Examples

```
KMO_base <- efa_kmo(test_models$baseline$cormat)
KMO_base

# format() returns the same lines as a character vector:
writeLines(format(KMO_base))
```

---

print.efa_loadings *Print a loading matrix*

Description

Print a loading matrix

Usage

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

## S3 method for class 'efa_loadings'
format(
  x,
  cutoff = 0.3,
  digits = 3,
  max_name_length = 10,
  h2 = NULL,
  color = TRUE,
  name_style = c("truncate", "abbreviate", "full"),
  max_factor_name_length = NULL,
```

```
  max_factors_per_block = NULL,
  sort_loadings = c("none", "primary", "clustered"),
  legend = FALSE,
  ...
)
```

Arguments

x	a loading matrix of class efa_loadings (or the legacy LOADINGS).
...	additional arguments passed to print or format
cutoff	numeric. The value at or above which loadings are emphasized; default is .3.
digits	numeric. Passed to round. Number of digits to round the loadings to (default is 3).
max_name_length	numeric. The maximum length of the variable names to display. Everything beyond this will be cut from the right unless name_style = "abbreviate" or name_style = "full" is used. Cutting never leaves two variables sharing a row label: if it would (as for items with a long common prefix), the names are abbreviated instead, and numbered if needed.
h2	numeric. Vector of communalities to print. If named and x has row names, names are used to align communalities to rows.
color	logical. Whether to apply console styling using **cli**. Default is TRUE.
name_style	character. How to shorten variable names longer than max_name_length. "truncate" cuts names from the right, "abbreviate" uses base::abbreviate(), and "full" prints full names.
max_factor_name_length	numeric or NULL. Optional maximum length of factor names. If NULL, factor names are not shortened.
max_factors_per_block	numeric or NULL. Maximum number of factor columns to print per block. If NULL, the number is chosen from the console width.
sort_loadings	character. Optional row sorting. "none" preserves the input order, "primary" groups rows by the factor with the largest absolute loading, and "clustered" additionally sorts within each factor by the size of the primary loading.
legend	logical. Whether to append a short explanation of the styling. Default is FALSE for standalone loading matrices. The legend is only printed when the styling it describes is actually rendered (color = TRUE and a colour-capable console); in plain output it is omitted and this argument has no effect.

Details

The method prints a loading matrix in a compact, console-oriented table. Loadings with absolute value greater than or equal to cutoff are emphasized, smaller loadings are de-emphasized, and Heywood-relevant communality/ uniqueness values are marked when h2 is supplied. Long variable names can be truncated, abbreviated, or printed in full. If the matrix has many factor columns, the table is split into column blocks so that the output remains readable in narrower consoles.

If h2 is named and x has row names, h2 is matched to the row names of x before any optional row sorting is applied. If x has no row names, a named h2 vector is used in the supplied order.

Value

print() returns its argument x invisibly; it is cat(format(x, ...), sep = "\n") followed by a blank line for console spacing. format() returns a character vector with the table lines (styled to the active console theme; plain when colours are disabled).

Examples

```
EFAtools_PAF <- efa_fit(test_models$baseline$cormat, n_factors = 3, N = 500,
                        estimator = "PAF", rotation = "promax")
EFAtools_PAF

# format() returns the same lines as a character vector:
writeLines(format(EFAtools_PAF$rot_loadings))
```

---

print.efa_power *Print and format an efa_power object*

Description

print() turns an efa_power() result into a short report, and format() builds the same report as a character vector (print() is cat(format(x), sep = "\n")).

Usage

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

## S3 method for class 'efa_power'
format(x, digits = 3, ...)
```

Arguments

x	An object of class efa_power (output from efa_power()).
digits	Integer. The number of decimal places the reported values are rounded to. Default is 3.
...	Not used; for consistency with the generic.

Details

For an RMSEA-mode object, the report has a header naming the test, the null and alternative hypotheses with the significance level and degrees of freedom, the headline result (the power at the sample size, or the required sample size for the target power), and the critical value and noncentrality parameters.

For a simulation-mode object, the report instead has the population and design, the retention hit-rate per criterion, the structure-recovery rate, and the convergence and Heywood-case rate.

The lines follow the active console theme, so they print as plain text when colours are disabled – for example when captured into a file, or stripped with `cli::ansi_strip()`.

Value

`print()` returns its argument `x` invisibly. `format()` returns a character vector with the report lines.

See Also

Other power analysis: `efa_power()`, `plot.efa_power()`

Examples

```
pw <- efa_power(df = 100, N = 200)
pw

# format() returns the same lines as a character vector:
writeLines(format(pw))
```

`print.efa_reliability` *Print and format a reliability object*

Description

`print()` shows the reliability coefficients for the general factor and the group factors, for a single group or for each group: the reliability coefficients (omega total, omega hierarchical, and omega subscale, standardized Cronbach's alpha, and the H index) and the common-variance indices (the explained common variance, ECV, and the percent of uncontaminated correlations, PUC). `format()` assembles the same report and returns it as a character vector; `print()` is `cat(format(x), sep = "\n")`. The lines follow the active console theme, so they are plain when colours are disabled (for example when captured into a file or stripped with `cli::ansi_strip()`).

Usage

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

## S3 method for class 'efa_reliability'
format(x, digits = 3, ...)
```

Arguments

x	An object of class efa_reliability.
digits	Integer. The number of decimal places the coefficients are rounded to. Default is 3.
...	Not used; for consistency with the generic.

Value

print() returns its argument x invisibly. format() returns a character vector with the report lines.

See Also

Other reliability coefficients: efa_reliability(), efa_schmid_leiman()

Examples

```
efa_mod <- efa_fit(test_models$baseline$cormat, N = 500, n_factors = 3,
                   estimator = "PAF", rotation = "promax")
rel <- efa_reliability(efa_mod)
rel

# format() returns the same lines as a character vector:
writeLines(format(rel))
```

---

print.efa_retain *Print method for efa_retain objects*

Description

Print method for efa_retain objects

Usage

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

Arguments

x	an object of class efa_retain, returned by efa_retain().
...	not used.

Value

print() returns its argument x invisibly; it is cat(format(x), sep = "\n").

Examples

```
efa_retain(test_models$baseline$cormat, criteria = c("EKC", "SMT"), N = 500)
```

---

print.efa_retention *Print method for efa_retention objects*

Description

Print method for efa_retention objects

Usage

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

Arguments

x	an object of class efa_retention, returned by a factor-retention criterion (e.g. efa_ekc() or efa_hull()).
...	not used.

Value

print() returns its argument x invisibly; it is cat(format(x), sep = "\n").

Examples

```
efa_ekc(test_models$baseline$cormat, N = 500)
```

---

print.efa_schmid_leiman

Print and format an efa_schmid_leiman object

Description

print() shows a summarised output of the efa_schmid_leiman() function: a model header (when the settings are available), the Schmid-Leiman loading matrix, and the variances accounted for. format() assembles the same report and returns it as a character vector; print() is cat(format(x), sep = "\n"). The lines follow the active console theme, so they are plain when colours are disabled (for example when captured into a file or stripped with cli::ansi_strip()).

Usage

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

## S3 method for class 'efa_schmid_leiman'
format(x, ...)
```

Arguments

x	An object of class efa_schmid_leiman (output from efa_schmid_leiman()).
...	Not used; for consistency with the generic.

Value

print() returns its argument x invisibly. format() returns a character vector with the report lines.

Examples

```
EFA_mod <- efa_fit(test_models$baseline$cormat, N = 500, n_factors = 3,
                   estimator = "PAF", rotation = "promax")
sl_mod <- efa_schmid_leiman(EFA_mod, estimator = "PAF")
sl_mod

# format() returns the same lines as a character vector:
writeLines(format(sl_mod))
```

---

print.efa_scores *Print and format an efa_scores object*

Description

print() shows a concise overview of an efa_scores() result: a header naming the method and whether factor scores were computed, and the per-factor determinacy table (determinacy, squared determinacy, and Guttman index). summary() returns a summary.efa_scores object whose print method adds the full factor-weight matrix, the score validity/univocality matrix, and the score inter-correlations. format() assembles the same report and returns it as a character vector; print() is cat(format(x), sep = "\n"). The lines follow the active console theme, so they are plain when colours are disabled (for example when captured into a file or stripped with cli::ansi_strip()).

Usage

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

## S3 method for class 'efa_scores'
format(x, digits = 3, ...)
```

```
## S3 method for class 'efa_scores'
summary(object, digits = 3, ...)

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

## S3 method for class 'summary.efa_scores'
format(x, digits = x$opts$digits, ...)
```

Arguments

x, object	An object of class efa_scores; for the summary.efa_scores methods, the object returned by summary().
digits	numeric. Number of decimal places for the printed tables. Default is 3.
...	Not used; for consistency with the generics.

Value

print() and the print method for summary.efa_scores objects return their argument invisibly. format() returns a character vector with the report lines. summary() returns an object of class summary.efa_scores.

See Also

Other factor scoring: efa_scores()

Examples

```
efa <- efa_fit(test_models$baseline$cormat, n_factors = 3, N = 500,
               estimator = "PAF", rotation = "oblimin")
fs <- efa_scores(test_models$baseline$cormat, f = efa)
fs
summary(fs)

# format() returns the same lines as a character vector:
writeLines(format(fs))
```

---

print.efa_screen *Print and format an efa_screen object*

Description

`print()` turns the factor-analysis screening diagnostics computed by `efa_screen()` into a sectioned report with banded, colour-coded verdicts: sampling adequacy and sphericity (the Kaiser-Meyer-Olkin measure and Bartlett's test of sphericity), multicollinearity (the determinant and condition number of the correlation matrix), the per-variable diagnostics, and, when raw data were supplied, multivariate normality and multivariate outliers. It closes with a consolidated list of actionable recommendations (for example, which items to consider dropping, whether to prefer an ordinal or a robust estimator, and a caveat that keeps an over-powered Bartlett's test from being over-trusted). `format()` assembles the same report and returns it as a character vector; `print()` is `cat(format(x), sep = "\n")`. The lines follow the active console theme, so they are plain when colours are disabled (for example when captured into a file or stripped with `cli::ansi_strip()`). `print()` does not draw a plot.

Usage

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

## S3 method for class 'efa_screen'
format(x, digits = 3, ...)
```

Arguments

`x`	An object of class `efa_screen` (output from `efa_screen()`).
`digits`	Integer. The number of decimal places the reported values are rounded to. Default is 3.
`...`	Not used; for consistency with the generic.

Value

`print()` returns its argument `x` invisibly. `format()` returns a character vector with the report lines.

See Also

Other factor analysis suitability: `efa_bartlett()`, `efa_kmo()`, `efa_screen()`

Examples

```
# From raw data
efa_screen(iris[, 1:4])

# From a correlation matrix (supply N for Bartlett's test of sphericity)
efa_screen(test_models$baseline$cormat, N = 500)

# format() returns the same lines as a character vector:
writeLines(format(efa_screen(test_models$baseline$cormat, N = 500)))
```

`print.efa_simulated` *Print and format an efa_simulated object*

Description

`print()` shows a compact summary of the data simulated by `efa_simulate()`: how many datasets were drawn and their dimensions, the marginal distribution, whether the data were discretized into ordered categories or given missing values, and – when model error was injected – the method with the target and achieved RMSEA and CFI. The simulated data themselves live in the `data` element and the population correlation matrix in `population`. `format()` returns the same summary as a character vector; `print()` is `cat(format(x), sep = "\n")`. The lines follow the active console theme, so they are plain when colours are disabled.

Usage

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

## S3 method for class 'efa_simulated'
format(x, digits = 3, ...)
```

Arguments

`x`	An object of class `efa_simulated` (output from `efa_simulate()`).
`digits`	Integer. The number of decimal places the reported fit values are rounded to. Default is 3.
`...`	Not used; for consistency with the generic.

Value

`print()` returns its argument `x` invisibly. `format()` returns a character vector with the summary lines.

See Also

Other data simulation: `efa_simulate()`

Examples

```
Lambda <- population_models$loadings$baseline
Phi <- population_models$phis_3$moderate
efa_simulate(N = 500, Lambda = Lambda, Phi = Phi, target_rmsea = 0.05, seed = 42)
```

`print.efa_sl_loadings` *Print an efa_sl_loadings object*

Description

Print an efa_sl_loadings object

Usage

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

## S3 method for class 'efa_sl_loadings'
format(
  x,
  cutoff = 0.2,
  digits = 3,
  max_name_length = 10,
  color = TRUE,
  name_style = c("truncate", "abbreviate", "full"),
  max_factors_per_block = NULL,
  sort_loadings = c("none", "primary", "clustered"),
  ...
)
```

Arguments

`x`	class efa_sl_loadings matrix.
`...`	additional arguments passed to print or format.
`cutoff`	numeric. The value at or above which loadings are emphasized (default is .2). The default is lower than the .3 of an ordinary loading table (`print.efa_loadings()`): the group-factor loadings are residualized, that is, they carry only the variance left once the general factor has been partialled out, and are therefore smaller than the corresponding first-order loadings.
`digits`	numeric. Passed to `round`. Number of digits to round the loadings to (default is 3).
`max_name_length`	numeric. The maximum length of the variable names to display; see `print.efa_loadings()`.
`color`	logical. Whether to apply console styling using **cli**. Default is `TRUE`.
`name_style`	character. How to shorten variable names longer than `max_name_length`; see `print.efa_loadings()`.
`max_factors_per_block`	numeric or `NULL`. Maximum number of factor columns to print per block. If `NULL`, the number is chosen from the console width.

`sort_loadings` character. Optional row sorting; see `print.efa_loadings()`. The default `"none"` keeps the input order. When sorting is requested, rows are grouped by their largest *group*-factor loading: the general factor is left out of the comparison, since it is the largest loading of almost every item and sorting on it would leave the order untouched.

Details

Prints a Schmid-Leiman loading matrix (general factor, group factors, and the communality/uniqueness columns) as a styled, decimal-aligned table. Loadings with absolute value greater than or equal to `cutoff` are emphasised, smaller loadings are de-emphasised, and Heywood-relevant cells (a loading or communality above 1, or a negative uniqueness) are highlighted. If the matrix has many columns or the console is narrow, the table is split into stacked column blocks so the output stays readable.

Value

`print()` returns its argument `x` invisibly; it is `cat(format(x, ...), sep = "\n")` followed by a blank line for console spacing. `format()` returns a character vector with the table lines (styled to the active console theme; plain when colours are disabled).

Examples

```
EFA_mod <- efa_fit(test_models$baseline$cormat, N = 500, n_factors = 3,
                   estimator = "PAF", rotation = "promax")
efa_schmid_leiman(EFA_mod, estimator = "PAF")
```

---

print.OMEGA *Print and format an OMEGA object*

Description

`print()` shows the omega coefficients computed by `OMEGA()`: omega total (and, for multi-factor solutions, omega hierarchical, omega subscale, the H index, the explained common variance, and the percent of uncontaminated correlations) for the general factor and the group factors, for a single group or for each group. `format()` assembles the same report and returns it as a character vector; `print()` is `cat(format(x), sep = "\n")`. The lines follow the active console theme, so they are plain when colours are disabled (for example when captured into a file or stripped with `cli::ansi_strip()`).

Usage

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

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

Arguments

x	An object of class OMEGA (output from OMEGA()).
digits	Integer. The number of decimal places the coefficients are rounded to (passed to base::round()). Default is 3.
...	Not used; for consistency with the generic.

Value

print() returns its argument x invisibly. format() returns a character vector with the report lines.

Examples

```
efa_mod <- efa_fit(test_models$baseline$cormat, N = 500, n_factors = 3,
                   estimator = "PAF", rotation = "promax")
sl_mod <- efa_schmid_leiman(efa_mod, estimator = "PAF")

om <- OMEGA(sl_mod, type = "EFAtools",
            factor_corres = sl_mod$sl[, c("F1", "F2", "F3")] >= .2)
om

# format() returns the same lines as a character vector:
writeLines(format(om))
```

PROCRUSTES *Rotate a loading matrix to a target using Procrustes alignment*

Description

[Superseded]

PROCRUSTES() has been superseded by efa_procrustes(), which is the recommended interface going forward. It remains available and unchanged so existing code keeps working.

Usage

```
PROCRUSTES(
  A,
  Target,
  rotation = c("orthogonal", "oblique"),
  S = NULL,
  T_init = NULL,
  oblique_eps = 1e-05,
  oblique_maxit = 1000,
  oblique_max_line_search = 10,
  oblique_step0 = 1,
  oblique_normalize = FALSE,
  oblique_random_starts = 0,
```

```
  oblique_screen_keep = 2,
  oblique_triage_maxit = 25,
  oblique_triage_improve_tol = 0
)
```

Arguments

A	Numeric loading matrix to be aligned.
Target	Numeric target matrix with the same dimensions as A.
rotation	Character string, either "orthogonal" or "oblique".
S	Optional k x k cross-product matrix crossprod(A), kept for compatibility. It enters both the oblique criterion and its gradient, so any other matrix would minimize a different criterion: where S is used it is checked against crossprod(A) and must agree with it up to a relative tolerance of 1e-8. That check forms crossprod(A) itself, so passing S no longer avoids any work: omitting it gives the same result for slightly less. S is used, and therefore checked, only on the oblique path with more than one factor and oblique_normalize = FALSE; if Kaiser normalization is requested, the cross-product must be recomputed on the normalized matrix and S is ignored.
T_init	Optional k x k starting transformation matrix for the oblique solver. Its columns are normalized internally, and the normalized matrix must be well enough conditioned to define a proper factor correlation matrix: its smallest singular value must be at least 1e-4, the same floor the solver applies to every candidate it evaluates. If NULL (the default), the oblique solver is warm-started from the closed-form orthogonal Procrustes solution.
oblique_eps	Positive convergence tolerance for the projected-gradient norm in the oblique solver.
oblique_maxit	Non-negative integer. Maximum number of projected-gradient updates in the full oblique solver.
oblique_max_line_search	Non-negative integer. Maximum number of step-halving attempts after the initial line-search step.
oblique_step0	Positive initial step size for the oblique solver.
oblique_normalize	Logical; if TRUE, apply Kaiser row normalization to the loadings (only) in the oblique solver and back-transform the aligned loadings afterwards, leaving Target unnormalized (as in GPArotation::targetQ(normalize = TRUE)).
oblique_random_starts	Non-negative integer. Number of additional random starts used by the oblique solver.
oblique_screen_keep	Non-negative integer. Number of random starts retained after cheap objective screening and sent to triage optimization.
oblique_triage_maxit	Non-negative integer. Number of short optimization iterations used in the triage stage.

oblique_triage_improve_tol
: Non-negative scalar. Relative improvement required for a triaged start to be promoted to full optimization.

Value

A list identical to the value of `efa_procrustes()`; see there for the components.

See Also

`efa_procrustes()`

---

residuals.efa *Extract residuals from an efa object*

Description

Returns the residual correlation matrix of an `efa_fit()` or `efa_mi()` solution. Residuals are a pure extractor here; their diagnostics and a formatted display are part of `summary.efa()`.

Usage

```
## S3 method for class 'efa'
residuals(object, type = c("raw", "standardized"), ...)
```

Arguments

`object`	a list of class `efa`. Output from `efa_fit()` or `efa_mi()`.
`type`	character. Which residuals to return. `"raw"` (default) returns `orig_R - model_implied_R`; `"standardized"` returns the standardized residuals (residuals divided by their standard errors), available only when the object was fitted with bootstrap standard errors.
`...`	Further arguments (currently unused).

Value

A numeric matrix of residual correlations.

Examples

```
efa <- efa_fit(test_models$baseline$cormat, n_factors = 3, N = 500)
residuals(efa)
```

RiskDimensions *RiskDimensions*

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

A list of 2 with elements "cormat" (9 x 9 matrix of bivariate correlations) and "N" (scalar). The correlation matrix contains the following risk dimensions:

Voluntariness (numeric) - Voluntariness of exposure to the risk.

Immediacy (numeric) - Immediacy of the risk's effect.

Known to exposed (numeric) - How well the risk is known to those exposed to it.

Known to science (numeric) - How well the risk is known to science.

Controllability (numeric) - Controllability of the risk.

Newness (numeric) - Newness of the risk.

Chronic (numeric) - Whether the risk is chronic rather than catastrophic.

Common (numeric) - Whether the risk is common rather than dreaded.

Severity of consequences (numeric) - Severity of the consequences.

Source

Fischhoff, B, Slovic, P, Lichtenstein, S, Read, S, and Combs, B. (1978). How safe is safe enough? A psychometric study of attitudes towards technological risks and benefits. Policy Sciences, 9, 127-152. doi: 10.1007/BF00143739

SCREE *Scree plot*

Description

[Superseded]

`SCREE()` has been superseded by `efa_scree()`, which is the recommended interface going forward. It remains available and unchanged so existing code keeps working.

Usage

```
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", "poly", "tetra"),
  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. Default is `c("PCA", "SMC", "EFA")`, i.e. all three; `"EFA"` is the only one that fits a model.
`use`	character. Passed to `stats::cor()` if raw data is given as input. Default is "pairwise.complete.obs".
`cor_method`	character. Correlation computed from raw data: `"pearson"`, `"spearman"`, or `"kendall"` (passed to `stats::cor()`), or `"poly"` / `"tetra"` for polychoric / tetrachoric correlations of ordinal / binary data (a two-step estimator). Default is "pearson".
`n_factors`	numeric. Number of factors to extract if "EFA" is included in `eigen_type`. Default is 1.
`...`	Further arguments passed on to the `efa_fit()` fit. For example, `estimator`, to change the estimator (PAF is default), or one of the estimation tuning knobs (`type`, `init_comm`, `criterion`, `criterion_type`, `max_iter`, `abs_eigen`, `start_method`),

which are repacked into an estimate_control() object so that they tune the fit exactly as they always did.

Value

An object of class efa_retention, identical to the value of efa_scree(); see there for the components.

See Also

efa_scree()

SL *Schmid-Leiman transformation*

Description

[Superseded]

SL() has been superseded by efa_schmid_leiman(), which is the recommended interface going forward. It remains available and unchanged so existing code keeps working.

Usage

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

Arguments

x	object of class efa_fit(), class psych::fa(), class lavaan::lavaan(), a matrix, or an efa_loadings/loadings object. If class efa_fit() or class psych::fa(), pattern coefficients and factor intercorrelations are taken from this object. If class lavaan::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).
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. In efa_schmid_leiman() it is set through the type of the estimate_control() object.

method	character. The estimator for the second-order factor analysis; passed to efa_schmid_leiman() as its estimator argument. One of "PAF", "ML", "ULS", or "MINRES".
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".
...	Further arguments passed on to the second-order efa_fit(), including the estimation tuning knobs (init_comm, criterion, criterion_type, max_iter, abs_eigen, start_method), which are repacked, together with type, into an estimate_control() object so that they tune that fit exactly as they always did. The estimator is selected with method.

Value

A list of class c("efa_schmid_leiman", "SL"), identical to the value of efa_schmid_leiman(); see there for the components.

See Also

efa_schmid_leiman()

SMT *Sequential model tests*

Description

[Superseded]

SMT() has been superseded by efa_smt(), which is the recommended interface going forward. It remains available and unchanged so existing code keeps working.

Usage

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

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. Must be larger than the number of variables.
use	character. Passed to stats::cor() if raw data is given as input. Default is "pairwise.complete.obs".

cor_method — character. One of "pearson", "spearman", or "kendall", passed to stats::cor(). "poly" and "tetra" are not supported because SMT rests on a normal-theory chi-square test that is not valid for polychoric / tetrachoric correlations. Default is "pearson".

Value

An object of class efa_retention, identical to the value of efa_smt(); see there for the components.

See Also

efa_smt()

SPSS_23 *Various outputs from SPSS (version 23) FACTOR*

Description

Various outputs from SPSS (version 23) FACTOR for the IDS-2 (Grob & Hagmann-von Arx, 2018), the WJIV (3 to 5 and 20 to 39 years; McGrew, LaForte, & 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 (2022).

Usage

SPSS_23

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, 6, 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 or item names, and generic variable labels (V1 to VN) for the simulated datasets.

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 or item names, and generic variable labels (V1 to VN) for the simulated datasets.

pro_load (matrix) - F1 to FN = promax rotated factor loadings obtained with the FACTOR algorithm with PAF. Rownames are the abbreviated subtest or item names, and generic variable labels (V1 to VN) for the simulated datasets.

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 or numeric) - 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, and the DOSPERT contain all the above entries. The NEO-PI-R contains all of them except L2 and sl, while the four simulated datasets contain only paf_load, var_load, pro_load, and pro_phi.

The principal axis factoring was run with the iteration limit raised above SPSS's own default of 25, so reproducing these solutions requires the same: `case_1a` needs 60 iterations and `case_11b` needs 33, and at `max_iter = 25` both stop short of convergence and differ from the stored loadings in the second decimal. Use `estimate_control(type = "SPSS", max_iter = 500)` when checking a preset against these references; the other two simulated cases converge in six iterations and are unaffected.

Source

Grieder, S., & Steiner, M. D. (2022). Algorithmic jingle jungle: A comparison of implementations of principal axis factoring and promax rotation in R and SPSS. Behavior Research Methods, 54, 54–74. doi: 10.3758/s13428-021-01581-x

Wolff, H.G., & Preising, K. (2005). Exploring item and higher order factor structure with the Schmid-Leiman solution: Syntax codes for SPSS and SAS. Behavior Research Methods, 37, 48–58. doi: 10.3758/BF03206397

Grieder, S., & Grob, A. (2019). Exploratory factor analyses of the intelligence and development scales–2: Implications for theory and practice. Assessment. Advance online publication. doi:10.1177/1073191119845051

Grob, A., & Hagmann-von Arx, P. (2018). Intelligence and Development Scales–2 (IDS-2). Intelligenz- und Entwicklungsskalen für Kinder und Jugendliche. [Intelligence and Development Scales for Children and Adolescents.]. Bern, Switzerland: Hogrefe.

Frey, R., Pedroni, A., Mata, R., Rieskamp, J., & Hertwig, R. (2017). Risk preference shares the psychometric structure of major psychological traits. Science Advances, 3, e1701381.

McGrew, K. S., LaForte, E. M., & Schrank, F. A. (2014). Technical Manual. Woodcock-Johnson IV. Rolling Meadows, IL: Riverside.

Schrank, F. A., McGrew, K. S., & Mather, N. (2014). Woodcock-Johnson IV. Rolling Meadows, IL: Riverside.

Costa, P. T., & McCrae, R. R. (1992). NEO PI-R professional manual. Odessa, FL: Psychological Assessment Resources, Inc.

SPSS_27 *Various outputs from SPSS (version 27) FACTOR*

Description

Various outputs from SPSS (version 27) FACTOR for the IDS-2 (Grob & Hagmann-von Arx, 2018), the WJIV (3 to 5 and 20 to 39 years; McGrew, LaForte, & Schrank, 2014), the DOSPERT (Frey et al., 2017; Weber, Blais, & Betz, 2002), and four simulated datasets (baseline, case_1a, case_6b, and case_11b, see test_models and population_models) used in Grieder and Steiner (2022).

Usage

```
SPSS_27
```

Format

A list of 8 containing EFA results for each of the data sets mentioned above. Each of these eight entries is a list of 4, of the following structure:

paf_load (matrix) - F1 to FN = unrotated factor loadings obtained with the FACTOR algorithm with PAF. Rownames are the abbreviated subtest or item names, and generic variable labels (V1 to VN) for the simulated datasets.

var_load (matrix) - F1 to FN = varimax rotated factor loadings obtained with the FACTOR algorithm with PAF. Rownames are the abbreviated subtest or item names, and generic variable labels (V1 to VN) for the simulated datasets.

pro_load (matrix) - F1 to FN = promax rotated factor loadings obtained with the FACTOR algorithm with PAF. Rownames are the abbreviated subtest or item names, and generic variable labels (V1 to VN) for the simulated datasets.

pro_phi (matrix) - F1 to FN = intercorrelations of the promax rotated loadings.

Details

The principal axis factoring was run with the iteration limit raised above SPSS's own default of 25, so reproducing these solutions requires the same: `case_1a` needs 60 iterations and `case_11b` needs 33, and at `max_iter = 25` both stop short of convergence and differ from the stored loadings in the second decimal. Use `estimate_control(type = "SPSS", max_iter = 500)` when checking a preset against these references; the other two simulated cases converge in six iterations and are unaffected.

Source

Grieder, S., & Steiner, M. D. (2022). Algorithmic jingle jungle: A comparison of implementations of principal axis factoring and promax rotation in R and SPSS. Behavior Research Methods, 54, 54–74. doi: 10.3758/s13428-021-01581-x

Grieder, S., & Grob, A. (2019). Exploratory factor analyses of the intelligence and development scales–2: Implications for theory and practice. Assessment. Advance online publication. doi:10.1177/1073191119845051

Grob, A., & Hagmann-von Arx, P. (2018). Intelligence and Development Scales–2 (IDS-2). Intelligenz- und Entwicklungsskalen für Kinder und Jugendliche. [Intelligence and Development Scales for Children and Adolescents.]. Bern, Switzerland: Hogrefe.

Frey, R., Pedroni, A., Mata, R., Rieskamp, J., & Hertwig, R. (2017). Risk preference shares the psychometric structure of major psychological traits. Science Advances, 3, e1701381.

McGrew, K. S., LaForte, E. M., & Schrank, F. A. (2014). Technical Manual. Woodcock-Johnson IV. Rolling Meadows, IL: Riverside.

Schrank, F. A., McGrew, K. S., & Mather, N. (2014). Woodcock-Johnson IV. Rolling Meadows, IL: Riverside.

`test_models` *Four test models used in Grieder and Steiner (2022)*

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 (2022) 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

Grieder, S., & Steiner, M. D. (2022). Algorithmic jingle jungle: A comparison of implementations of principal axis factoring and promax rotation in R and SPSS. Behavior Research Methods, 54, 54–74. doi: 10.3758/s13428-021-01581-x

UPPS_raw *UPPS_raw*

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

A data.frame with 645 rows (participants) and 45 columns, named by a subscale prefix and item number, covering the four UPPS subscales:

perseverance_1 to perseverance_10 (numeric) - Perseverance-subscale items.

premeditation_1 to premeditation_11 (numeric) - Premeditation-subscale items.

ss_1 to ss_12 (numeric) - Sensation-seeking-subscale items.

urgency_1 to urgency_12 (numeric) - Urgency-subscale items.

Source

Whiteside, S. P., Lynam, D. R., Miller, J. D., & Reynolds, S. K. (2005). Validation of the UPPS impulsive behaviour scale: A four-factor model of impulsivity. European Journal of Personality, 19 (7), 559–574.

Steiner, M., & Frey, R. (2020). Representative design in psychological assessment: A case study using the Balloon Analogue Risk Task (BART). PsyArXiv Preprint. doi:10.31234/osf.io/dg4ks

WJIV_ages_14_19 *Woodcock Johnson IV: ages 14 to 19*

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.

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

McGrew, K. S., LaForte, E. M., & Schrank, F. A. (2014). Technical Manual. Woodcock-Johnson IV. Rolling Meadows, IL: Riverside.

Schrank, F. A., McGrew, K. S., & Mather, N. (2014). Woodcock-Johnson IV. Rolling Meadows, IL: Riverside.

`WJIV_ages_20_39` *Woodcock Johnson IV: ages 20 to 39*

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

```
WJIV_ages_20_39
```

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

McGrew, K. S., LaForte, E. M., & Schrank, F. A. (2014). Technical Manual. Woodcock-Johnson IV. Rolling Meadows, IL: Riverside.

Schrank, F. A., McGrew, K. S., & Mather, N. (2014). Woodcock-Johnson IV. Rolling Meadows, IL: Riverside.

WJIV_ages_3_5 *Woodcock Johnson IV: ages 3 to 5*

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.

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.

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.

SCI (numeric) - Science.

SOC (numeric) - Social Studies.

HUM (numeric) - Humanities.

Source

McGrew, K. S., LaForte, E. M., & Schrank, F. A. (2014). Technical Manual. Woodcock-Johnson IV. Rolling Meadows, IL: Riverside.

Schrank, F. A., McGrew, K. S., & Mather, N. (2014). Woodcock-Johnson IV. Rolling Meadows, IL: Riverside.

`WJIV_ages_40_90` *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.

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

McGrew, K. S., LaForte, E. M., & Schrank, F. A. (2014). Technical Manual. Woodcock-Johnson IV. Rolling Meadows, IL: Riverside.

Schrank, F. A., McGrew, K. S., & Mather, N. (2014). Woodcock-Johnson IV. Rolling Meadows, IL: Riverside.

WJIV_ages_6_8 *Woodcock Johnson IV: ages 6 to 8*

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

McGrew, K. S., LaForte, E. M., & Schrank, F. A. (2014). Technical Manual. Woodcock-Johnson IV. Rolling Meadows, IL: Riverside.

Schrank, F. A., McGrew, K. S., & Mather, N. (2014). Woodcock-Johnson IV. Rolling Meadows, IL: Riverside.

`WJIV_ages_9_13` *Woodcock Johnson IV: ages 9 to 13*

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.

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

McGrew, K. S., LaForte, E. M., & Schrank, F. A. (2014). Technical Manual. Woodcock-Johnson IV. Rolling Meadows, IL: Riverside.

Schrank, F. A., McGrew, K. S., & Mather, N. (2014). Woodcock-Johnson IV. Rolling Meadows, IL: Riverside.

Index

∗ **Control functions**
  estimate_control, 122
  print.efa_control, 161
∗ **data simulation**
  efa_simulate, 112
  print.efa_simulated, 173
∗ **datasets**
  DOSPERT, 8
  DOSPERT_raw, 9
  GRiPS_raw, 129
  IDS2_R, 132
  population_models, 152
  RiskDimensions, 179
  SPSS_23, 183
  SPSS_27, 185
  test_models, 186
  UPPS_raw, 187
  WJIV_ages_14_19, 187
  WJIV_ages_20_39, 189
  WJIV_ages_3_5, 191
  WJIV_ages_40_90, 192
  WJIV_ages_6_8, 194
  WJIV_ages_9_13, 196
∗ **factor analysis suitability**
  efa_bartlett, 25
  efa_kmo, 58
  efa_screen, 108
  print.efa_screen, 171
∗ **factor analysis**
  efa_average, 13
  efa_fit, 33
  efa_group, 47
  efa_mi, 62
  plot.efa_group, 149
  print.efa_group, 162
∗ **factor comparison**
  efa_compare, 28
∗ **factor retention criteria**
  efa_cd, 26
  efa_ekc, 31
  efa_hull, 52
  efa_kgc, 56
  efa_map, 60
  efa_nest, 70
  efa_parallel, 73
  efa_retain, 94
  efa_scree, 106
  efa_smt, 119
∗ **factor rotation**
  efa_procrustes, 84
  efa_schmid_leiman, 99
∗ **factor scoring**
  efa_scores, 102
  print.efa_scores, 170
∗ **power analysis**
  efa_power, 78
  plot.efa_power, 150
  print.efa_power, 166
∗ **reliability coefficients**
  efa_reliability, 87
  efa_schmid_leiman, 99
  print.efa_reliability, 167

BARTLETT, 4
base::abbreviate(), *165*
base::mean(), *15*, *22*
base::round(), *176*
base::set.seed(), *27*, *38*, *54*, *72*, *75*, *97*

CD, 5
cli::ansi_strip(), *155*, *158–160*, *162*, *163*, *167*, *169*, *170*, *172*, *175*
COMPARE, 6

DOSPERT, 8
DOSPERT_raw, 9

EFA, 9
EFA_AVERAGE, 21

efa_average, 13
efa_average(), *21, 22*, *24*, *46*, *51*, *70*, *147*, *149*, *158*, *163*
efa_bartlett, 25
efa_bartlett(), *4, 5*, *59*, *97, 98*, *109*, *111*, *159*, *172*
efa_cd, 26
efa_cd(), *5, 6*, *33*, *55*, *58*, *62*, *73*, *76*, *95*, *97, 98*, *107*, *121*, *137*
efa_compare, 28
efa_compare(), *6*, *8*, *46*, *148*, *160, 161*
efa_ekc, 31
efa_ekc(), *28*, *55*, *57, 58*, *62*, *73*, *76*, *96–98*, *107*, *121, 122*, *129*, *138*, *152*, *169*
efa_fit, 33, *54*
efa_fit(), *9*, *11*, *13–18*, *20*, *22–24*, *31*, *40*, *47–51*, *53, 54*, *56, 57*, *62–71*, *74*, *77, 78*, *80*, *83*, *88, 89*, *91*, *93*, *95–103*, *105*, *107*, *116*, *118*, *120*, *124–126*, *132*, *134*, *138, 139*, *147*, *149*, *155, 156*, *163*, *178*, *180–182*
efa_group, 47
efa_group(), *20*, *46*, *70*, *149*, *162, 163*
efa_hull, 52
efa_hull(), *28*, *33*, *57, 58*, *62*, *73*, *76*, *95–98*, *107*, *121*, *129–132*, *137, 138*, *152*, *169*
efa_kgc, 56
efa_kgc(), *28*, *31–33*, *55*, *61, 62*, *73*, *75, 76*, *95–98*, *107*, *121*, *133, 134*, *138*
efa_kmo, 58
efa_kmo(), *25, 26*, *97, 98*, *109*, *111*, *134, 135*, *163, 164*, *172*
efa_map, 60
efa_map(), *28*, *33*, *55*, *58*, *73*, *76*, *97, 98*, *107*, *121*, *135, 136*, *151*
efa_mi, 62
efa_mi(), *20*, *37*, *44*, *46*, *51*, *63*, *76*, *78*, *116*, *149*, *155, 156*, *163*, *178*
efa_nest, 70
efa_nest(), *28*, *33*, *55*, *58*, *62*, *76*, *95–98*, *107*, *121*, *138–140*
efa_parallel, *53, 54*, 73, *131*
efa_parallel(), *28*, *32, 33*, *53–55*, *57, 58*, *62*, *71*, *73*, *95–98*, *107*, *121*, *131*, *137, 138*, *145*, *147*
EFA_POOLED, 76
EFA_POOLED(), *63*, *78*
efa_power, 78
efa_power(), *113*, *118*, *150, 151*, *166, 167*
efa_procrustes, 84
efa_procrustes(), *31*, *51*, *63*, *69*, *78*, *101*, *176*, *178*
efa_reliability, 87
efa_reliability(), *46*, *100, 101*, *140*, *144*, *168*
efa_retain, 94
efa_retain(), *10*, *26*, *28*, *33, 34*, *46*, *55*, *57–59*, *62*, *72, 73*, *75, 76*, *80*, *83*, *107*, *111*, *121*, *125*, *128*, *136–138*, *151*, *168*
efa_schmid_leiman, 99
efa_schmid_leiman(), *46*, *87–89*, *91*, *93*, *125*, *140–142*, *168–170*, *181, 182*
efa_scores, 102
efa_scores(), *46*, *126, 127*, *170, 171*
efa_scree, 106
efa_scree(), *28*, *33*, *55*, *58*, *62*, *73*, *76*, *95–98*, *121*, *138*, *180, 181*
efa_screen, 108
efa_screen(), *26*, *59*, *98*, *172*
efa_simulate, 112
efa_simulate(), *78*, *80–83*, *173*
efa_smt, 119
efa_smt(), *28*, *33*, *55*, *57, 58*, *62*, *73*, *76*, *97, 98*, *107*, *151*, *182, 183*
EKC, 121
estimate_control, 122
estimate_control(), *9*, *13*, *33*, *35*, *43*, *46*, *49*, *53*, *56*, *64*, *67*, *71*, *74*, *78*, *97*, *100*, *107*, *120*, *125*, *132*, *134*, *138, 139*, *147*, *162*, *181, 182*

FACTOR_SCORES, 126
format.efa *(*print.efa*)*, 155
format.efa_average *(*print.efa_average*)*, 158
format.efa_bartlett *(*print.efa_bartlett*)*, 159
format.efa_compare *(*print.efa_compare*)*, 160
format.efa_estimate_control *(*print.efa_control*)*, 161
format.efa_group *(*print.efa_group*)*, 162
format.efa_kmo *(*print.efa_kmo*)*, 163
format.efa_loadings *(*print.efa_loadings*)*, 164

format.efa_mi *(print.efa)*, 155
format.efa_power *(print.efa_power)*, 166
format.efa_reliability *(print.efa_reliability)*, 167
format.efa_retain, 128
format.efa_retention, 129
format.efa_rotate_control *(print.efa_control)*, 161
format.efa_schmid_leiman *(print.efa_schmid_leiman)*, 169
format.efa_scores *(print.efa_scores)*, 170
format.efa_screen *(print.efa_screen)*, 171
format.efa_simulated *(print.efa_simulated)*, 173
format.efa_sl_loadings *(print.efa_sl_loadings)*, 174
format.OMEGA *(print.OMEGA)*, 175
format.summary.efa *(print.efa)*, 155
format.summary.efa_scores *(print.efa_scores)*, 170
future::plan(), *16*, *39*, *48*, *54*, *73*, *75*, *83*, *117*
future_lapply(), *16*, *48*, *73*

ggplot2::ggplot, *149–152*
GRiPS_raw, 129

HULL, 130

IDS2_R, 132

KGC, 133
KMO, 134

MAP, 135

N_FACTORS, 136
NEST, 139

OMEGA, 140
OMEGA(), *93*, *101*, *175*, *176*

PARALLEL, 145
plot.efa_average, 147
plot.efa_average(), *159*
plot.efa_compare, 148
plot.efa_compare(), *7*, *30*
plot.efa_group, 149
plot.efa_group(), *20*, *46*, *51*, *70*, *162*, *163*
plot.efa_power, 150
plot.efa_power(), *83*, *167*
plot.efa_retain, 151
plot.efa_retention, 152
plot.efa_retention(), *28*, *32*, *54*, *57*, *61*, *72*, *75*, *107*, *120*, *151*
population_models, 152, *183*, *185*, *186*
print.efa, 155
print.efa_average, 158
print.efa_bartlett, 159
print.efa_compare, 160
print.efa_compare(), *7*, *30*
print.efa_control, *125*, 161
print.efa_estimate_control *(print.efa_control)*, 161
print.efa_group, 162
print.efa_group(), *20*, *46*, *51*, *70*, *149*
print.efa_kmo, 163
print.efa_loadings, 164
print.efa_loadings(), *156*, *174*, *175*
print.efa_mi *(print.efa)*, 155
print.efa_power, 166
print.efa_power(), *83*, *151*
print.efa_reliability, 167
print.efa_reliability(), *92*, *93*, *101*
print.efa_retain, 168
print.efa_retention, 169
print.efa_retention(), *28*, *32*, *54*, *57*, *61*, *72*, *75*, *107*, *120*
print.efa_rotate_control *(print.efa_control)*, 161
print.efa_schmid_leiman, 169
print.efa_scores, 170
print.efa_scores(), *105*
print.efa_screen, 171
print.efa_screen(), *26*, *59*, *111*
print.efa_simulated, 173
print.efa_simulated(), *118*
print.efa_sl_loadings, 174
print.OMEGA, 175
print.summary.efa *(print.efa)*, 155
print.summary.efa_scores *(print.efa_scores)*, 170
PROCRUSTES, 176
psych::cor.smooth(), *61*, *114*
psych::cortest.bartlett(), *26*
psych::fa(), *11*, *12*, *15*, *16*, *23*, *24*, *40*, *54*,

100, *101*, *123*, *124*, *181*
psych::factor.model(), *141*, *143*
psych::factor.scores(), *126*
psych::KMO(), *59*
psych::omega(), *90*, *140*, *143*
psych::schmid(), *88*, *89*, *91*, *99*, *101*, *140–143*

residuals.efa, 178
RiskDimensions, 179
rotate_control *(*estimate_control*)*, 122
rotate_control(), *9*, *13*, *33*, *35*, *46*, *49*, *64*, *67*, *78*, *162*
round, *165*, *174*

SCREE, 180
SL, 181
SMT, 182
SPSS_23, 183
SPSS_27, 185
stats::cor(), *4–6*, *12*, *16*, *24*, *25*, *27*, *32*, *35*, *36*, *53*, *56*, *58*, *60*, *71*, *74*, *95*, *106*, *108*, *119*, *120*, *122*, *131*, *133–135*, *137*, *139*, *146*, *180*, *182*, *183*
stats::cov2cor(), *113*, *115*
stats::factanal(), *12*, *16*, *24*, *40*, *123*
stats::na.omit(), *27*
stats::optim(), *19*, *42*
stats::varimax(), *12*, *16*, *23*, *124*
summary.efa *(*print.efa*)*, 155
summary.efa(), *178*
summary.efa_mi *(*print.efa*)*, 155
summary.efa_scores *(*print.efa_scores*)*, 170

test_models, *183*, *185*, 186

UPPS_raw, 187
utils::combn(), *44*

WJIV_ages_14_19, 187
WJIV_ages_20_39, 189
WJIV_ages_3_5, 191
WJIV_ages_40_90, 192
WJIV_ages_6_8, 194
WJIV_ages_9_13, 196