Package: facomplex (via r-universe)

July 22, 2026

Title Methods for Assessing Factor Complexity in Factor Analysis Solutions

Version 0.0.3

Depends R (>= 3.1.0)

Imports ggplot2, stats

Description Provides methods for estimating factor complexity coefficients in exploratory and confirmatory factor analysis (EFA/CFA) results. Included indices are the Hofman coefficient, Fleming's approach for factor simplicity, and others. Additional outputs include descriptive statistics (minimum, maximum, and mean) for target and non-target loadings, and visualization of results. References: Fleming, J.S. (2003) <doi:10.3758/bf03195531>; Hofmann, R.J. (1978) <doi:10.1207/s15327906mbr1302_9>; Kaiser, H.F. (1974) <doi:10.1007/BF02291575>; Bentler, P.M. (1977) <doi:10.1007/BF02294054>; Lorenzo-Seva, U. (2003) <doi:10.1007/BF02296652>.

License GPL-3

Encoding UTF-8

Roxygen list(markdown = TRUE)

LazyData TRUE

Suggests knitr, rmarkdown, psych, lavaan, testthat (>= 3.0.0)

VignetteBuilder knitr

Config/roxygen2/version 8.0.0

Config/testthat/edition 3

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

Date/Publication 2026-07-20 18:40:26 UTC

RemoteUrl https://github.com/cmerinos/facomplex

RemoteRef HEAD

RemoteSha 93838d9817463813d2f744dd28a6024d5fc9ccb5

Contents

`BSI` *BSI - Bentler Simplicity Index*

Description

Computes Bentler's Simplicity Index (BSI) for a loading matrix. This is a scale-free, matrix-level measure of factor simplicity originally proposed by Bentler (1977).

Usage

```
BSI(data)
```

Arguments

`data`	A numeric matrix or data frame of factor loadings with `p` rows (items/variables) and `r` columns (factors).

Details

Let L be a $p \times r$ loading matrix, and let $A = [l_{ij}^2]$ be the matrix of squared loadings. Define

$$D = \text{diag}(A'A)$$

where D is the diagonal matrix formed from the diagonal elements of $A'A$. Bentler's Simplicity Index is then:

$$BSI = \left| D^{-1/2} A'AD^{-1/2} \right|$$

where $|.|$ denotes the determinant.

The index ranges from 0 to 1. Higher values indicate a simpler factor structure. A value of 1 is obtained when the loading matrix shows perfect factorial simplicity (i.e., each variable is associated with only one factor). Bentler's index is invariant to the scale of the factors.

Value

A single numeric value in the interval `[0, 1]` representing Bentler's global simplicity index for the full loading matrix.

References

Bentler, P. M. (1977). Factor simplicity index and transformations. *Psychometrika, 42*(2), 277–295. https://doi.org/10.1007/BF02294054

Fleming, J. S. (2003). Computing measures of simplicity of fit for loadings in factor-analytically derived scales. *Behavior Research Methods, Instruments, & Computers, 35*(4), 520–524. https://doi.org/10.3758/BF03195531

Lorenzo-Seva, U. (2003). A factor simplicity index. *Psychometrika, 68*(1), 49–60. https://doi.org/10.1007/BF02296652

Examples

```
ex1_data <- data.frame(
 F1 = c(0.536, 0.708, 0.600, 0.673, 0.767, 0.481, -0.177, 0.209, -0.097, -0.115, 0.047, 0.024),
 F2 = c(-0.110, 0.026, 0.076, 0.011, -0.160, 0.106, 0.668, 0.438, 0.809, 0.167, 0.128, 0.041),
 F3 = c(-0.100, 0.036, 0.086, 0.021, -0.150, 0.116, 0.678, 0.448, 0.819, 0.577, 0.738, 0.751)
)

BSI(ex1_data)
```

---

entropyFL *Entropy Index for Factor Simplicity*

Description

Computes entropy-based indices to quantify the factorial simplicity or complexity of an Exploratory Factor Analysis (EFA) solution. Entropy is computed at three levels: by item, by factor, and globally. Two types of entropy measures are available: normalized and scaled.

Usage

```
entropyFL(
  loadings_matrix,
  base = 2,
  normalized = TRUE,
  scaled = FALSE,
  bounded = TRUE,
  nd = 3
)
```

Arguments

`loadings_matrix`	A numeric matrix or data frame of factor loadings, where rows represent items and columns represent factors.
`base`	The logarithmic base used to compute entropy. Default is `2`, corresponding to entropy in bits.
`normalized`	Logical. If `TRUE` (default), entropy values are normalized to range from 0 to 1 by dividing by $\log_b(k)$ or $\log_b(n)$.
`scaled`	Logical. If `TRUE`, returns the scaled entropy index and the theoretical minimum entropy as proposed by Beisel & Moreteau (1997). Default is `FALSE`.
`bounded`	Logical. If `TRUE` (default), forces scaled entropy values to remain within the 0 to 1 range by truncating negative or >1 values.
`nd`	Integer. Number of decimal places to round the results. Default is `3`. Use `NULL` for no rounding.

Details

The entropy index is based on the squared factor loadings (λ_{ij}^2), interpreted as the proportion of shared variance between item i and factor j (Shannon, 1948).

1. Normalized Entropy:

- For each item i, define the pseudo-proportions:

$$p_{ij} = \frac{\lambda_{ij}^2}{\sum_{j=1}^{k} \lambda_{ij}^2}$$

- Then compute Shannon entropy:

$$H_i = -\sum_{j=1}^{k} p_{ij} \log_b(p_{ij})$$

- If `normalized = TRUE`, divide by $\log_b(k)$ to constrain values to 0 to 1 range.

The same logic applies for factors (across items), replacing p_{ij} with:

$$q_{ij} = \frac{\lambda_{ij}^2}{\sum_{i=1}^{n} \lambda_{ij}^2}$$

2. Global Entropy: Entropy can also be calculated for the full loading matrix as a whole:

$$p_{ij} = \frac{\lambda_{ij}^2}{\sum_{i,j} \lambda_{ij}^2}$$

$$H = -\sum_{i,j} p_{ij} \log_b(p_{ij})$$

and normalized by $\log_b(n \cdot k)$.

3. Scaled Entropy (Beisel & Moreteau, 1997): When `scaled = TRUE`, the function returns:

- H_{min}: A theoretical lower bound for entropy when one factor dominates:

$$H_{min} = -[p_{max} \log_b(p_{max}) + (1 - p_{max}) \log_b((1 - p_{max})/(k - 1))]$$

- H_{scaled}: A scaled measure between H_{min} and H_{max}:

$$H_{scaled} = \frac{H - H_{min}}{H_{max} - H_{min}}$$

This calculation is applied both to items and to factors. For factors, k is replaced by n (number of items).

4. Argument `bounded`**:** Scaled entropy can occasionally produce values outside 0 to 1 range if entropy is below the theoretical minimum. If `bounded = TRUE`, the function truncates those values to stay within 0 to 1 range for interpretive clarity.

Value

A list with:

`Hnormalized` A list with entropy by item, factor, and total.

`Hscaled` A list with `Hmin.items`, `Hscaled.items`, `Hmin.factors`, `Hscaled.factors`, and `Hscaled.total` (if `scaled = TRUE`).

#' **Interpretation:**

- Values near 0 indicate high factorial simplicity (loadings concentrated on a single factor).
- Values near 1 suggest factorial complexity or ambiguity (dispersed loadings across factors).
- `Hnormalized` expresses entropy as a proportion of the maximum possible entropy.
- `Hscaled` expresses entropy relative to its theoretical minimum and maximum, allowing finer differentiation across contexts.
- The index can be used to compare different rotation methods, number of factors, or item structures.

Although no formal cutoff exists, entropy values below 0.20 typically reflect strong factorial simplicity, while values near or above 0.80 may indicate multidimensionality or poor simple structure. Interpretation should always be contextualized using additional indices, visual inspection of loadings, and substantive theory.

References

Shannon, C. E. (1948). A mathematical theory of communication. *Bell System Technical Journal*, 27(3), 379–423. doi:10.1002/j.15387305.1948.tb00917.x

Beisel, J. N., & Moreteau, J.-C. (1997). A new method to estimate the lower bound of the Shannon-Wiener index of diversity. *Ecological Modelling*, 99(1), 99-105.

Hofmann, R. J. (1978). Complexity and simplicity as objective indices descriptive of factor solutions. *Multivariate Behavioral Research*, 13(2), 247-250.

Lorenzo-Seva, U. (2003). A factor simplicity index. *Psychometrika*, 68(1), 49-60. doi:10.1007/BF02296652

Examples

```
# Example: items with different factorial complexity
loadings <- matrix(c(
  0.7, 0.0, 0.01,
  0.1, 0.2, 0.15,
  0.4, 0.8, 0.2,
  0.4, 0.4, 0.4
), nrow = 4, byrow = TRUE)

entropyFL(loadings, normalized = TRUE, scaled = TRUE, bounded = TRUE)
```

`fullclean` *fullclean: Data frame of responses to motivations for conducting research*

Description

Actual data from a study on the motivations for conducting research among Peruvian university professors. The data were collected from several Peruvian universities.

Usage

```
data(fullclean)
```

Format

Data frame: 589 rows, 19 columns:

ID Subject Identification Number

EDAD Subject's age. Type: numeric

ESTUDIO Educational level. Type: character

TIEMPODOC Teaching experience, in years, categorized. Type: character

SEXO Subject's gender: Male, Female. Type: character

INV1 ... INV12 Scale A: Item 1 through INV12. Type: numeric

INV2 ... INV14 Scale B: Item 2 through INV14. Type: numeric

Examples

```
data(fullclean)
head(fullclean)
summary(fullclean)
```

`Hofmann` *Hofmann Index of Factorial Complexity and its Normalized Inverse*

Description

Computes Hofmann's (1977) coefficient of factorial complexity for each item in a factor loading matrix, along with a normalized inverse version rescaled between 0 and 1.

Usage

```
Hofmann(data)
```

Arguments

`data`	A numeric data frame or matrix of factor loadings, where rows represent items and columns represent factors. Factor loadings are typically between -1 and 1.

Details

The original Hofmann index (`Hofmann`) quantifies the extent to which an item loads on multiple factors. It ranges from 1 (perfect factorial simplicity, i.e., loading only on one factor) to p (maximum complexity, where the item loads equally on all p factors).

The modified version is the reciprocal of `Hofmann`, resulting in a simplicity index normalized to the interval 0 to 1. Higher values indicate greater factorial simplicity.

The Hofmann complexity index is computed as:

$$CHof_i = \frac{(\sum_j \lambda_{ij}^2)^2}{\sum_j \lambda_{ij}^4}$$

where λ_{ij} is the loading of item i on factor j.

The rescaled index `Hoff_R` is computed as `1 / CHof_1`, and provides a bounded indicator of simplicity (closer to 1 means simpler structure).

These indices are particularly useful when comparing items in terms of their factorial clarity, and complement other measures such as Bentler's or Fleming's simplicity indices. An extensive use of this coefficient can be found in: Pettersson & Turkheimer (2010, 2014).

Value

A data frame with two columns:

- `CHof`: Hofmann's original complexity coefficient for each item.
- `CHof_R`: The inverse of `Hofmann`, representing in factor-level (range: 0, 1).

References

Hofmann, R. J. (1977). Indices descriptive of factor complexity. *The Journal of General Psychology*, 96(1), 103-110. https://doi.org/10.1080/00221309.1977.9920803

Pettersson, E., & Turkheimer, E. (2014). Self-Reported Personality Pathology Has Complex Structure and Imposing Simple Structure Degrades Test Information. *Multivariate Behavioral Research*, 49(4), 372-389. https://doi.org/10.1080/00273171.2014.911073

Pettersson, E., & Turkheimer, E. (2010). Item selection, evaluation, and simple structure in personality data. *Journal of Research in Personality*, 44(4), 407-420. https://doi.org/10.1016/j.jrp.2010.03.002

Examples

```
# Simulated factor loadings
 ex1.data <- data.frame(
  F1 = c(0.536, 0.708, 0.600, 0.673, 0.767, 0.481, -0.177, 0.209, -0.097, -0.115, 0.047, 0.024),
  F2 = c(-0.11, 0.026, 0.076, 0.011, -0.16, 0.106, 0.668, 0.438, 0.809, 0.167, 0.128, 0.041),
  F3 = c(-0.1, 0.036, 0.086, 0.021, -0.15, 0.116, 0.678, 0.448, 0.819, 0.577, 0.738, 0.751)
)
Hofmann(ex1.data)
```

HofmannFac *Calculate Hofmann Factor Complexity Index for columns/factors*

Description

This function calculates the Hofmann complexity index (`Choff`) for each factor (column) in a factor loading matrix. The index estimates the factorial complexity at the factor level, indicating how many items contribute significantly to each factor.

Usage

```
HofmannFac(data)
```

Arguments

`data`	A `data.frame` or `matrix` containing the factor loadings. Rows represent items, and columns represent factors in the factorial model. The values in the matrix should be factor loadings.

Details

Hofmann's complexity index for factors evaluates how many items contribute to the definition of each factor. A value close to 1 indicates that a factor is well-defined by only one item (unidimensional), while values closer to p indicate higher complexity.

The formula is based on the sum of squared and quartic factor loadings, and the result is normalized so that higher complexity values suggest factors defined by multiple items.

#' The index ranges from 1 (indicating a factor with a single item) to p (the total number of **items** in the factor, indicating the number of items involved in the definition of the factor). The formula for the index is based on summing squared and quartic factor loadings, similar to the original Hofmann (1977) method applied to the factor (column) dimension. In this case, when calculating the complexity at the factor level, we assess how many items significantly contribute to each factor. In a latent dimension with known structure, the resulting value should be equal or close to the number of items expected in this dimension. Values different from the expected number of items suggests there are items with significant loadings or items close to or in the hyperplane. As more items contribute to the factor, the value of **Chof** increases, reaching p (the number of items) when all items significantly contribute to the factor.

Value

A `data.frame` containing one column:

- `Choff`: The Hofman factor complexity index for each factor, ranging from 1 (1 significant item) to p (maximum number of items in the matrix).

References

Hofmann, R. J. (1977). Indices descriptive of factor complexity. *The Journal of General Psychology*, 96(1), 103-110.

Examples

```
# Example factor loading matrix
 ex1.data <- data.frame(
  F1 = c(0.536, 0.708, 0.600, 0.673, 0.767, 0.481, -0.177, 0.209, -0.097, -0.115, 0.047, 0.024),
  F2 = c(-0.11, 0.026, 0.076, 0.011, -0.16, 0.106, 0.668, 0.438, 0.809, 0.167, 0.128, 0.041),
  F3 = c(-0.1, 0.036, 0.086, 0.021, -0.15, 0.116, 0.678, 0.448, 0.819, 0.577, 0.738, 0.751)
)

# Calculate Hofman factor complexity
results_factor <- HofmannFac(ex1.data)

# View results
print(results_factor)
```

`KC` *KC: Kaiser-Cerny Simplicity Index and Ideal Hyperplane Count*

Description

Computes the Kaiser-Cerny (1978) criterion for factorial simplicity based on a power function of the absolute loadings (inspired by Kendall & Stuart, 1969). Also returns the ideal hyperplane count as an expected benchmark of factorial parsimony (Catell, 1952).

Usage

```
KC(data, b = 4)
```

Arguments

data	A data.frame or numeric matrix of factor loadings, where rows represent items (variables) and columns represent factors.
b	A positive numeric value for the power parameter in the Kaiser-Cerny formula. Default is 4.

Details

The Kaiser-Cerny simplicity index is computed for each factor j using the formula:

$$f_j = \left(\frac{1}{m} \sum_{i=1}^{m} a_{ij}^{2/b} \right)^{b/2}$$

where a_{ij} is the loading of item i on factor j, and m is the number of items.

This index provides a quantitative assessment of factorial parsimony, where lower values of f_j indicate a clearer hyperplane structure-meaning more loadings are close to zero-thus favoring simpler factor interpretation.

The function also reports the *ideal hyperplane count*, defined as:

$$m(p-1)$$

where p is the number of factors. This represents the theoretical number of near-zero loadings required for a perfectly simple structure in factor analysis.

Value

An object of class "KC" containing:

- fj: A numeric vector with the Kaiser-Cerny simplicity index for each factor.
- ideal_hyperplane_count: The ideal hyperplane count.
- m: Number of items.
- p: Number of factors.
- b: The power parameter used.

References

Cattell, R. B. (1952). Factor analysis: an introduction and manual for the psychologist and social scientist. Oxford, England: Harper.

Kaiser, H. F., & Cerny, B. A. (1978). Casey's Method For Fitting Hyperplanes From An Intermediate Orthomax Solution. *Multivariate Behavioral Research*, 13(4), 395-401. https://doi.org/10.1207/s15327906mbr1304_2

Kendall, M. G., & Stuart, A. (1969). *The Advanced Theory of Statistics*, Vol. 2. London: Griffin.

Examples

```
# Simulated example
set.seed(123)
loadings <- matrix(runif(30, -1, 1), nrow = 10, ncol = 3)
KC(loadings)
```

`LSIglobal` *LSIglobal: Loading Simplicity Index (Lorenzo-Seva, 2003)*

Description

Computes the Loading Simplicity Index (LSI) as proposed by Lorenzo-Seva (2003), adapted from the implementation in `lazy.fa::LS_index`. This global index evaluates the overall factorial simplicity of a loading matrix using a non-linear weighting scheme to emphasize dominant factor loadings.

Usage

```
LSIglobal(loadings)
```

Arguments

`loadings`	A numeric matrix or data frame of factor loadings. Rows represent items, columns represent factors.

Details

The LSI reflects the extent to which the factor solution exhibits simple structure. It applies a double normalization to the loading matrix and then computes a non-linear function over the squared normalized loadings. High values (close to 1) indicate greater factorial simplicity, while lower values (close to 0) reflect more diffuse or complex loading patterns.

The index is scaled to the interval 0 to 1 as follows:

$$LSI = \frac{w - e}{1 - e}$$

where w is the weighted average complexity across all loadings, and e is the theoretical minimum expected under uniform distribution of loadings.

Value

A numeric value between 0 and 1 indicating the global simplicity of the solution.

References

Lorenzo-Seva, U. (2003). A factor simplicity index. *Psychometrika*, 68(1), 49-60. doi:10.1007/BF02296652

Code adapted from: `lazy.fa::LS_index`

Examples

```
L <- matrix(c(
  0.6, 0.2,
  0.5, 0.3,
  0.1, 0.7
), nrow = 3, byrow = TRUE)

LSIglobal(L)
```

`plotFacomplex` *Plot Simplicity Index Values*

Description

Creates a horizontal bar plot of item-level simplicity or complexity values (e.g., from FSI, BSI, Hofmann indices). The user must specify the column name that contains the coefficient values (e.g., "IFS", "Complexity", etc.).

Usage

```
plotFacomplex(
  data,
  item.col = "Item",
  value.col = "Coefficient",
  sort.items = c("ascending", "none", "descending"),
  reverse.items = FALSE,
  theme = c("light", "classic", "minimal"),
  title = "Simplicity Index by Item",
  bar.color = "blue",
  threshold.line = NULL,
  threshold.color = "red"
)
```

Arguments

`data`	A data frame with item labels and simplicity or complexity values.
`item.col`	Character. Name of the column with item labels. Default is `"Item"`.
`value.col`	Character. Name of the column with the simplicity or complexity values. This argument is required.
`sort.items`	Character. Sorting order of items: `"none"`, `"ascending"`, or `"descending"`. Default is `"ascending"`.
`reverse.items`	Logical. If `TRUE`, reverses the order of items on the Y axis (top to bottom). Default is `FALSE`.

theme	Character. ggplot2 theme to use: "light", "classic", or "minimal". Default is "light".
title	Title of the plot. Default is "Simplicity Index by Item".
bar.color	Fill color for the bars. Default is "blue".
threshold.line	Optional numeric value. If specified, a horizontal dashed reference line is drawn at this threshold.
threshold.color	Color for the threshold line and label. Default is "red".

Value

A horizontal ggplot2 bar plot of item-level values.

Examples

```
# Data frame of factor loadings: 3 factors, 12 items; ESEM solution from published article
ex1_fl <- data.frame(
F1 = c(0.536, 0.708, 0.600, 0.673, 0.767, 0.481, -0.177, 0.209, -0.097, -0.115, 0.047, 0.024),
F2 = c(-0.110, 0.026, 0.076, 0.011, -0.160, 0.106, 0.668, 0.438, 0.809, 0.167, 0.128, 0.041),
F3 = c(-0.100, 0.036, 0.086, 0.021, -0.150, 0.116, 0.678, 0.448, 0.819, 0.577, 0.738, 0.751))

# Example using FSI output:
FSIout <- simload(ex1_fl,
              items_target = list(F1 = c(1, 2, 3, 4, 5, 6),
                                  F2 = c(7, 8, 9),
                                  F3 = c(10, 11, 12)))

# Basic use (value.col is required)
plotFacomplex(
  data = FSIout$IFS,
  item.col = "Items",
  value.col = "IFS")

# Customizing options:
plotFacomplex(
  data = FSIout$IFS,
  item.col = "Items",
  value.col = "IFS",
  sort.items = "none",
  reverse.items = TRUE,
  theme = "classic",
  bar.color = "darkgreen",
  threshold.line = 0.90)


# This will trigger an error if value.col is missing:
#
# plotFacomplex(data = FSIout$IFS)  # Error: 'value.col' is required
#
```

print.KC *Print method for KC objects*

Description

Print method for KC objects

Usage

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

Arguments

x	An object of class "KC".
...	Additional arguments (not used).

Value

Invisibly returns the input object x (called for side effects of printing).

profileFacomplex *Profile of Factorial Complexity*

Description

Computes descriptive statistics of factor loadings per factor, distinguishing between target and cross-loadings. Optionally, reports the percentage of cross-loadings below and above a user-defined cutoff.

Usage

```
profileFacomplex(loadings, target, abs = TRUE, cutoff = NULL, digits = 3)
```

Arguments

loadings	A numeric matrix or data.frame of factor loadings. Rows are items and columns are factors.
target	A named list, where each name corresponds to a factor in loadings, and each element is a character vector of item names (matching the row names of loadings) assigned as target to that factor.
abs	Logical. Should absolute values of loadings be used? Default is TRUE.
cutoff	Optional numeric. If defined, reports the percentage of cross-loadings <= and > this value.
digits	Integer. Number of decimal places to round the output. Default is 3.

Details

The function returns a data.frame where:

- Each row corresponds to a summary statistic.
- Each column (after the first) corresponds to a factor defined in `target`.

The following statistics are computed per factor:

- **Mean.Target, Median.Target, SD.Target**: Central tendency and dispersion of loadings on the target factor.
- **Min.Target, Max.Target**: Minimum and maximum of the target loadings.
- **Mean.Cross, Median.Cross, SD.Cross**: Descriptive statistics of cross-loadings (i.e., loadings on non-target factors).
- **Min.Cross, Max.Cross**: Minimum and maximum of the cross-loadings.
- **Perc.Cross.<=.cutoff, Perc.Cross.>.cutoff**: If `cutoff` is specified, these show the percentage of cross-loadings <= or > that threshold.
- **n.Items**: Number of items assigned to each factor.

This orientation (statistics in rows, factors in columns) facilitates interpretation and avoids excessively wide tables when the number of factors is large.

Value

A data.frame in long format: each row corresponds to a statistic and each column to a factor.

Examples

```
# Example with simulated data
loadings <- matrix(c(.70, .20,
                     .68, .22,
                     .15, .75,
                     .10, .70), ncol = 2, byrow = TRUE)
rownames(loadings) <- c("item1", "item2", "item3", "item4")
colnames(loadings) <- c("F1", "F2")

target <- list(F1 = c("item1", "item2"),
               F2 = c("item3", "item4"))

profileFacomplex(loadings, target, cutoff = 0.30)
```

`simload` *Factor Simplicity indices for total, scale and items*

Description

Calculates a fit indices to evaluate the factorial simplicity of multidimensional scales. The function estimates factorial simplicity at the item level, factor level, and overall solution level. It is particularly useful for solutions that include expected cross-loadings. The approach used in this function is when there is prior knowledge of the factor structure; that is, the items that correspond to a factor (target items) are known. It is appropriate for target rotations in EFA/ESEM.

Usage

```
simload(data, items_target)
```

Arguments

`data`	A matrix or data frame where rows represent items and columns represent factors. Each value should be a standardized or pattern factor loading.
`items_target`	A named list indicating the target items per factor. Each element should be a numeric vector indicating the row indices (or item positions) expected to load on the corresponding factor (column).

Details

This function is designed for factorial solutions from models such as EFA with target rotation or ESEM. It requires a matrix or data frame of standardized or pattern loadings. These levels of adjustment in the factorial matrix come from Fleming's approach for the SIMLOAD software (Fleming, 2003). Fleming (2003) proposes three levels of fit based on the degree of factorial simplicity: total, scale/factor and item. The item fit he derived from index of factorial simplicity (Kaiser, 1974); at the scale/factor level, factor scale fit index (SFI; Fleming, 1985, 2003); and at the total matrix, a derivated index from SFI. No like SIMLOAD software, here do not calculate the Bentler Simplicity Index (BSI; Bentler, 1977).

Value

A list containing three elements:

TSFI A numeric value representing the factorial simplicity of the overall loading matrix.

SFI A named vector with the factor scale fit index of each factor (column).

IFS A data frame with two columns: `Items` (item names) and `IFS` (index of factorial simplicity of each item).

References

Bentler, P. M. (1977). Factor simplicity index and transformations. *Psychometrika, 42*(2), 277-295. https://doi.org/10.1007/BF02294054

Fleming, J. S., & Merino Soto, C. (2005). Medidas de simplicidad y de ajuste factorial: un enfoque para la evaluación de escalas construidas factorialmente. *Revista De Psicologia*, 23(2), 250-266. https://doi.org/10.18800/psico.200502.002

Fleming, J. S. (1985). An index of fit for factor scales. *Educational and Psychological Measurement*, 45, 725-728. https://doi.org/10.1177/0013164485454002

Kaiser, H. F. (1974). An index of factorial simplicity. *Psychometrika*, 39, 31-35. https://doi.org/10.1007/BF02291575

Fleming, J. S. (2003). Computing measures of simplicity of fit for loadings in factor-analytically derived scales. *Behavior Research Methods, Instruments, & Computers*, 35(4), 520-524. https://doi.org/10.3758/bf03195531

Examples

```
##### Example 1 #####
ex1_fl <- data.frame(
 F1 = c(0.536, 0.708, 0.600, 0.673, 0.767, 0.481, -0.177, 0.209, -0.097, -0.115, 0.047, 0.024),
 F2 = c(-0.110, 0.026, 0.076, 0.011, -0.160, 0.106, 0.668, 0.438, 0.809, 0.167, 0.128, 0.041),
 F3 = c(-0.100, 0.036, 0.086, 0.021, -0.150, 0.116, 0.678, 0.448, 0.819, 0.577, 0.738, 0.751)
)

simload(data = ex1_fl,
    items_target = list(F1 = c(1, 2, 3, 4, 5, 6),
                        F2 = c(7, 8, 9),
                        F3 = c(10, 11, 12)))


##### Example 2 #####
data(fullclean)

INV.target <- matrix(0, 12, 2)
INV.target[1:6, 1] <- NA
INV.target[7:12, 2] <- NA

INV.esem.model <- 'efa("efa1")*f1 +
efa("efa1")*f2 =~ INV1 + INV4 + INV5 + INV7 + INV11 +
                  INV12 + INV3 + INV6 + INV8 + INV9 + INV13 + INV14'

INV.esem.fit <- lavaan::sem(INV.esem.model,
                            data = fullclean,
                            ordered = FALSE,
                            estimator = "ulsmv",
                            rotation = "target",
                            rotation.args = list(target = INV.target,
                                                 geomin.epsilon = 0.01,
                                                 rstarts = 30,
                                                 algorithm = "gpa",
                                                 std.ov = TRUE))

simload(data = lavaan::lavInspect(INV.esem.fit, what = "std")$lambda,
```

```
  items_target = list(f1 = c(1, 2, 3, 4, 5, 6),
                      f2 = c(7, 8, 9, 10, 11, 12)))
```

`SSindices` *SSindices: Target-Based Simple Structure Indices*

Description

Computes three target-based indices of factorial simplicity: `SStarget`, `SSntarget`, and their ratio `SSratio`. These quantify how much of the total explained variance is aligned with a predefined target structure versus misaligned (cross-loading) variance.

Usage

```
SSindices(loadings, target, per.factor = FALSE)
```

Arguments

`loadings`	A numeric matrix or data frame of factor loadings (items x factors).
`target`	A binary matrix or data frame of the same dimensions as `loadings`, indicating the expected loading structure: 1 = expected (target) loading, 0 = non-target.
`per.factor`	Logical. If TRUE, returns a data frame with indices computed per factor (column). Default is FALSE.

Details

This function builds on the logic of `lazy.fa::ss_index`, which evaluates off-diagonal complexity based on squared loadings. `SSindices()` extends the concept by incorporating a user-defined binary target structure, allowing explicit evaluation of how well the factor solution conforms to theoretical expectations.

Suggestive interpretation of the indices:

- `SStarget`: Proportion of total variance that is aligned with the expected structure. Higher values indicate clearer factor-item alignment.
- `SSntarget`: Proportion of variance explained by unexpected (non-target or cross) loadings. Reflects the degree of noise or factorial complexity in the solution.
- `SSratio`: The ratio between expected and non-expected variance. It indicates how dominant the expected structure is over residual complexity.

Suggestive interpretation for SSntarget (cross-loading contribution):

- ~= `0.00`: Perfectly simple structure (each item loads clearly on only one factor).
- < `0.05`: Very good factor differentiation.
- `0.05` - `0.15`: Moderate cross-loading complexity.

- > 0.15: Substantial interdependence or noise across factors.

Suggestive interpretion for SSratio:

- > 4: Excellent structure – target pattern clearly dominates.
- 2 - 4: Good structure with acceptable noise.
- 1 - 2: Target and cross-loadings are comparable – caution advised.
- ~= 1: Equal contribution – borderline structure.
- < 1: Cross-loadings dominate – weak or misaligned structure.

Value

A data.frame with:

- SStarget: Proportion of total explained variance due to target loadings.
- SSntarget: Proportion of variance explained by non-target (cross) loadings.
- SSratio:The ratio between target and non-target variance (SStarget / SSntarget). Values >1 indicate dominance of the expected structure.

References

Thurstone, L. L. (1947). *Multiple factor analysis*. University of Chicago Press.

Examples

```
Lx <- matrix(c(
  0.6, 0.2,
  0.5, 0.3,
  0.1, 0.7
), nrow = 3, byrow = TRUE)

Tx <- matrix(c(
  1, 0,
  1, 0,
  0, 1
), nrow = 3, byrow = TRUE)

SSindices(Lx, Tx)

SSindices(Lx, Tx, per.factor = TRUE)
```

Index