Package: facomplex (via r-universe)

November 23, 2026

Title Methods for Assessing Factor Complexity and Simplicity in Factor Analysis Solutions

Version 2.0.0

Description Provides methods for examining factorial complexity and simplicity in multidimensional scales, as well as in unidimensional scales when compared, in exploratory and ESEM factor analysis (EFA) results. The package implements coefficients derived from the work of Kaiser, Hofmann, Flemings, Bentler, and Lorenzo-Seva. The sums of the squares of the factorial loads (target and non-target loadings) can be estimated for descriptive purposes. Point estimates and confidence intervals are also obtained using bootstrap simulation.

Depends R (>= 3.5.0)

Imports ggplot2, stats, boot, lavaan, utils

License GPL-3

Encoding UTF-8

LazyData TRUE

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

VignetteBuilder knitr

Config/roxygen2/version 8.1.1

Config/testthat/edition 3

NeedsCompilation no

Author Merino-Soto Cesar A. [aut, cre], Dominguez-Lara Sergio [ctb]

Maintainer Merino-Soto Cesar A. <sikayax@yahoo.com.ar>

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

Date/Publication 2026-11-23 07:50:23 UTC

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

RemoteRef HEAD

RemoteSha 6460449911aecb2c6544524c1debbe2246a3e964

Contents

bootLoadings 2
BSI 5
ciBSI 6
ciHofmann 7
ciLSI 8
ciSimload 9
ciSSindices 10
entropyFL 11
fullclean 13
Hofmann 14
HofmannFac 15
KC 17
LSIglobal 18
plotFacomplex 19
print.KC 21
profileFacomplex 21
simload 23
SSindices 25

Index **27**

`bootLoadings` *Bootstrap of Factor Loadings for facomplex Indices*

Description

Executes a non-parametric bootstrap of factor loadings for EFA or ESEM models fitted with **lavaan**. Each replicate is aligned to the original solution using Tucker's congruence coefficient and optimal assignment via the Hungarian algorithm, thereby resolving rotational indeterminacy. The resulting object is passed to the `ci*` functions in the package to obtain asymmetric confidence intervals for the factorial simplicity indices.

Usage

```
bootLoadings(
  data,
  engine = c("EFA", "ESEM"),
  nfactors = NULL,
  model = NULL,
  target = NULL,
  rotation = NULL,
  rotation_list = list(),
  R = 1000,
  conf.level = 0.95,
  type = c("perc", "bc"),
  parallel = c("no", "multicore", "snow"),
```

```
  ncpus = 1,
  progress = TRUE,
  align = TRUE,
  ...
)
```

Arguments

`data`	A `data.frame` containing the items (rows = subjects, columns = items).
`engine`	Character. Either `"EFA"` or `"ESEM"`.
`nfactors`	Integer. Number of factors. Required if `engine = "EFA"`.
`model`	Character string. Model syntax. Required if `engine = "ESEM"`.
`target`	Target structure. Accepts: (a) a 0/1 matrix (1 = expected loading, 0 = cross-loading); (b) a 0/NA matrix (**lavaan** convention: NA = free loading, 0 = fixed cross-loading); (c) a named list of indices per factor (the `items_target` format used by `simload`). If `NULL`, no target rotation is applied.
`rotation`	Character string indicating the rotation method for **lavaan**. If `target` is not `NULL`, defaults to `"target"`. If `NULL` and no target is provided, defaults to `"oblimin"` (EFA) or `"geomin"` (ESEM).
`rotation_list`	A list of additional arguments to pass to the rotation algorithm (e.g., `list(target = Tmat)`).
`R`	Integer. Number of bootstrap replicates. Defaults to 1000.
`conf.level`	Numeric. Confidence level for the intervals. Defaults to 0.95.
`type`	Character. Type of asymmetric confidence interval: `"perc"` (percentile) or `"bc"` (bias-corrected). Defaults to `"perc"`.
`parallel`	Character. Controls for parallelization; see `boot`.
`ncpus`	Integer. Number of cores for parallelization.
`progress`	Logical. If `TRUE`, displays progress every 10 percent.
`align`	Logical. If `TRUE` (recommended), aligns bootstrap loadings with the original solution. See **Details**.
`...`	Additional arguments passed to `lavaan::efa` or `lavaan::sem` (e.g., `estimator`, `ordered`, `missing`).

Details

Factor alignment. Each bootstrap replicate may return the factors in a different order or with opposite signs. If left uncorrected, between-replicate variability would be artificially inflated, and confidence intervals for indices such as `HofmannFac` or `simload` would be computed by comparing incompatible columns. Alignment involves three steps:

1. Tucker's congruence coefficient between each bootstrap column and each original column.
2. Optimal assignment via the Hungarian algorithm (`clue::solve_LSAP`) applied to the absolute congruences.
3. Sign reflection to ensure positive congruence with the original solution.

Alignment corrects for permutation and reflection, but not for local optima of the rotation algorithm. If a replicate converges to a genuinely different structure, the congruence will be low, and the confidence interval will reflect this (wider intervals). This is valid information, not an error.

Value

An object of class `facomplexBoot` containing:

- `loadings_original`: A p x r matrix of loadings from the full sample.
- `loadings_boot`: An R x p x r array of aligned bootstrap loadings.
- `boot`: The `boot` object from **boot**.
- `R`, `R_valid`: Requested and valid bootstrap replicates.
- `conf.level`, `type`: Confidence interval parameters.
- `engine`, `rotation`, `target`, `nfactors`: Metadata.

References

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

See Also

ciHofmann, ciBSI, ciLSI, ciSimload, ciSSindices

Examples

```
data(HolzingerSwineford1939, package = "lavaan")
d <- HolzingerSwineford1939[, paste0("x", 1:9)]

# Without target: free rotation + automatic alignment
br <- bootLoadings(d, nfactors = 3, R = 50)
print(br)

# With target (lavaan's 0/NA convention)
target <- matrix(0, 9, 3)
target[1:3, 1] <- NA; target[4:6, 2] <- NA; target[7:9, 3] <- NA
br <- bootLoadings(d, nfactors = 3, target = target, R = 50)

# ESEM
model <- paste0('efa("efa1")*f1 + efa("efa1")*f2 + efa("efa1")*f3 =~ ',
                'x1 + x2 + x3 + x4 + x5 + x6 + x7 + x8 + x9')
br <- bootLoadings(d, engine = "ESEM", model = model, target = target,
                   R = 50)
```

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

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

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

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

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)
```

---

ciBSI *IC bootstrap para el Bentler Simplicity Index (BSI)*

Description

Intervalos de confianza asimetricos para el indice global de simplicidad de Bentler (1977), a partir de un objeto `facomplexBoot`.

Usage

```
ciBSI(boot_result, conf.level = NULL, type = NULL)
```

Arguments

`boot_result`	Objeto `facomplexBoot`.
`conf.level`	Nivel de confianza. Si `NULL`, hereda.
`type`	`"perc"` o `"bc"`. Si `NULL`, hereda.

Value

`data.frame` con una fila (`level = "matrix"`, `unit = "global"`).

See Also

`bootLoadings`, `BSI`

Examples

```
data(HolzingerSwineford1939, package = "lavaan")
d <- HolzingerSwineford1939[, paste0("x", 1:9)]
br <- bootLoadings(d, nfactors = 3, R = 50)
ciBSI(br)
```

ciHofmann *Bootstrap Confidence Intervals for Hofmann's Index (CHof)*

Description

Computes asymmetric bootstrap confidence intervals for Hofmann's (1977) factorial complexity index at the item and/or factor level, based on a `facomplexBoot` object generated by `bootLoadings`.

Usage

```
ciHofmann(
  boot_result,
  level = c("item", "factor", "both"),
  scaledVersion = FALSE,
  simplVersion = FALSE,
  includeMean = TRUE,
  conf.level = NULL,
  type = NULL
)
```

Arguments

`boot_result`	A `facomplexBoot` object.
`level`	Character. One of `"item"`, `"factor"`, or `"both"`.
`scaledVersion`	Logical. If `TRUE`, adds `CHof.R` (complexity scaled from 0 to 1).
`simplVersion`	Logical. If `TRUE`, adds `CHof.S` (simplicity scaled from 0 to 1).
`includeMean`	Logical. If `TRUE`, adds a row `unit = "mean"` with the average of the items (item level only).
`conf.level`	Confidence level. If `NULL`, inherits from `boot_result`.
`type`	Character. `"perc"` (percentile) or `"bc"` (bias-corrected). If `NULL`, inherits from `boot_result`.

Details

For each bootstrap replica, `Hofmann` (item level) or `HofmannFac` (factor level) is recalculated. Confidence intervals are constructed using the empirical quantiles of the replicates (percentile method) or with bias correction (`"bc"`).

Value

A `data.frame` with columns `level`, `unit`, `index`, `estimate`, `lwr.ci`, `upr.ci`, `conf.level`, and `type`.

See Also

`bootLoadings`, `Hofmann`, `HofmannFac`

Examples

```
data(HolzingerSwineford1939, package = "lavaan")
d <- HolzingerSwineford1939[, paste0("x", 1:9)]
br <- bootLoadings(d, nfactors = 3, R = 50)

ciHofmann(br)
ciHofmann(br, level = "factor")
ciHofmann(br, level = "both", scaledVersion = TRUE, simplVersion = TRUE)
```

---

ciLSI *IC bootstrap para el Loading Simplicity Index (LSI)*

Description

Intervalos de confianza asimetricos para el indice global de simplicidad de Lorenzo-Seva (2003), a partir de un objeto `facomplexBoot`.

Usage

```
ciLSI(boot_result, conf.level = NULL, type = NULL)
```

Arguments

`boot_result`	Objeto `facomplexBoot`.
`conf.level`	Nivel de confianza. Si `NULL`, hereda.
`type`	`"perc"` o `"bc"`. Si `NULL`, hereda.

Value

`data.frame` con una fila (`level = "matrix"`, `unit = "global"`).

See Also

`bootLoadings`, `LSIglobal`

Examples

```
data(HolzingerSwineford1939, package = "lavaan")
d <- HolzingerSwineford1939[, paste0("x", 1:9)]
br <- bootLoadings(d, nfactors = 3, R = 50)
ciLSI(br)
```

ciSimload *IC bootstrap para los indices de Fleming (simload)*

Description

Intervalos de confianza asimetricos para TSFI (nivel matriz), SFI (nivel factor) e IFS (nivel item), calculados con simload sobre las cargas bootstrap.

Usage

```
ciSimload(boot_result, conf.level = NULL, type = NULL)
```

Arguments

boot_result	Objeto facomplexBoot. Debe haberse construido con un target (ver bootLoadings).
conf.level	Nivel de confianza. Si NULL, hereda.
type	"perc" o "bc". Si NULL, hereda.

Details

El items_target se toma del target guardado en boot_result y se mantiene fijo en todas las replicas: la definicion de que items pertenecen a que factor es parte de la hipotesis estructural, no algo que deba variar bajo remuestreo.

Value

data.frame largo con filas para matriz (TSFI), factores (SFI) e items (IFS).

See Also

bootLoadings, simload

Examples

```
data(HolzingerSwineford1939, package = "lavaan")
d <- HolzingerSwineford1939[, paste0("x", 1:9)]
target <- matrix(0, 9, 3)
target[1:3, 1] <- NA; target[4:6, 2] <- NA; target[7:9, 3] <- NA
br <- bootLoadings(d, nfactors = 3, target = target, R = 50)
ciSimload(br)
```

`ciSSindices` *IC bootstrap para SSindices (SStarget, SSntarget, SSratio)*

Description

Intervalos de confianza asimetricos para los indices de simplicidad basados en target de `SSindices`, a nivel global o por factor, a partir de un objeto `facomplexBoot`.

Usage

```
ciSSindices(boot_result, per.factor = FALSE, conf.level = NULL, type = NULL)
```

Arguments

`boot_result`	Objeto `facomplexBoot`. Debe haberse construido con un `target`.
`per.factor`	Logico. Si `TRUE`, devuelve resultados por factor.
`conf.level`	Nivel de confianza. Si `NULL`, hereda.
`type`	`"perc"` o `"bc"`. Si `NULL`, hereda.

Value

`data.frame` largo. Si `per.factor = FALSE`, filas `level = "matrix"`; si `per.factor = TRUE`, filas `level = "factor"`.

See Also

`bootLoadings`, `SSindices`

Examples

```
data(HolzingerSwineford1939, package = "lavaan")
d <- HolzingerSwineford1939[, paste0("x", 1:9)]
target <- matrix(0, 9, 3)
target[1:3, 1] <- 1; target[4:6, 2] <- 1; target[7:9, 3] <- 1
br <- bootLoadings(d, nfactors = 3, target = target, R = 50)

ciSSindices(br)
ciSSindices(br, per.factor = TRUE)
```

`entropyFL` *Entropy Index for Factor Simplicity*

Description

Computes entropy-based indices to quantify factorial simplicity or complexity from a matrix of factor loadings (or network loadings from EGA). Two modes are available: exploratory (`type = "expl"`) and confirmatory (`type = "conf"`). In confirmatory mode, a target vector must be provided indicating the expected factor for each item; entropy is then computed on a binary distribution (target vs. non-target variance).

The function returns entropy values per item, per factor, and a global index. Entropy can be normalized (divided by the maximum possible entropy) to obtain values between 0 and 1.

In exploratory mode, the global entropy is calculated on the entire matrix of squared loadings, normalized by $\log(n \cdot k)$. In confirmatory mode, the global entropy is computed as the average of two components: (1) the entropy of the squared loadings of target items within each factor (how concentrated are the expected loadings), and (2) the entropy of the squared loadings of non-target items within each factor (how dispersed are the cross-loadings). This provides a measure of overall factorial clarity that aligns with the confirmatory logic.

Usage

```
entropyFL(
  loadings_matrix,
  base = 2,
  normalized = TRUE,
  type = c("expl", "conf"),
  target = NULL
)
```

Arguments

`loadings_matrix`	A numeric matrix of factor loadings (rows = items, columns = factors).
`base`	Logarithmic base (default 2, entropy in bits).
`normalized`	Logical. If `TRUE`, entropy is divided by the maximum possible entropy: for items $\log(k)$ (or $\log(2)$ in conf mode), for factors $\log(n)$ (or $\log(2)$ in conf mode), and for global as described above. Default is `TRUE`.
`type`	Character. Either `"expl"` (exploratory, default) or `"conf"` (confirmatory).
`target`	Integer vector of length `nrow(loadings_matrix)`. Required when `type = "conf"`. Each entry indicates the factor (column index) that the item is expected to load on. Values must be between 1 and number of factors.

Details

In exploratory mode, entropy for an item is computed from the normalized squared loadings across all factors. In confirmatory mode, the squared loadings are collapsed into two categories: the target factor and all others combined. Similarly for factors: target items vs. non-target items.

The global entropy in confirmatory mode is calculated as:

1. For each factor, compute the entropy of the distribution of squared loadings among its target items (if any). Average these values across factors to obtain H_{target}.
2. For each factor, compute the entropy of the distribution of squared loadings among its non-target items (if any). Average these values across factors to obtain $H_{non-target}$.
3. The total confirmatory entropy is the average of the two: $H_{total} = (H_{target} + H_{non-target})/2$.

This measure reflects how well the loading matrix conforms to the expected structure: low values indicate that target items have concentrated loadings and non-target items have low or dispersed cross-loadings (good fit), while high values suggest ambiguity or poor discrimination.

Interpretation:

- **Item entropy** (`H_i`): Low values (near 0) indicate that the item loads predominantly on a single factor (simple structure). High values (near 1) suggest cross-loadings or factorial complexity.
- **Factor entropy** (`H_f`): Low values indicate that the factor is defined by few items (specific factor). High values suggest that the factor is broadly dispersed across many items (general or diffuse factor).
- **Global entropy** (`H_total`): In exploratory mode, reflects the overall dispersion of variance across all matrix cells. In confirmatory mode, it reflects the clarity of the target structure, with lower values indicating better alignment with the expected pattern.

When applied to network loadings (NT) from **Exploratory Graph Analysis (EGA)**, this index quantifies the clarity of node–community assignment. Low entropy indicates that nodes load predominantly on a single community (simple structure), high entropy suggests diffuse or ambiguous membership, complementing the *Total Entropy Fit Index (TEFI)* for global fit assessment.

Value

A list of data frames:

`item` Data frame with columns `item` (names) and `entropy`.

`factor` Data frame with columns `factor` (names) and `entropy`.

`total` Data frame with a single row: `metric` = `"global"` and `entropy`.

If `normalized` = `FALSE`, the raw entropy values are returned; otherwise, normalized values (0–1) are returned.

References

Shannon, C. E. (1948). A mathematical theory of communication. *Bell System Technical Journal*, 27(3), 379–423. 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. McCammon, R. B. (1966). Minimum entropy criterion for factor analysis. *Journal of the ACM*, 13(2), 247–250.

Examples

```
# Exploratory mode
loadings_expl <- matrix(c(0.7,0.1,0.1, 0.2,0.8,0.1, 0.3,0.3,0.9), nrow=3, byrow=TRUE)
rownames(loadings_expl) <- paste0("Item", 1:3)
colnames(loadings_expl) <- paste0("F", 1:3)

entropyFL(loadings_expl, type = "expl", normalized = TRUE)

# Confirmatory mode
loadings_conf <- loadings_expl
target <- c(1, 2, 3)
entropyFL(loadings_conf, type = "conf", target = target, normalized = FALSE)
```

---

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 Scaled Versions*

Description

Computes Hofmann's (1977) coefficient of factorial complexity for each item in a factor loading matrix, along with optionally scaled versions for complexity (0–1) and simplicity (0–1).

Usage

```
Hofmann(data, scaledVersion = FALSE, simplVersion = FALSE)
```

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.
`scaledVersion`	Logical; if `TRUE`, adds a column `CHof.R` with complexity scaled to 0 to 1. Default is `FALSE`.
`simplVersion`	Logical; if `TRUE`, adds a column `CHof.S` with simplicity scaled to 0 to 1. Default is `FALSE`.

Details

The original Hofmann index (`CHof`) 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 scaled complexity version (`CHof.R`, when `scaledVersion = TRUE`) linearly transforms the original index to a 0–1 scale:

$$CHof.R = \frac{CHof - 1}{p - 1}$$

where p is the number of factors. Values close to 0 indicate high factorial simplicity; values close to 1 indicate high complexity.

The simplicity version (`CHof.S`, when `simplVersion = TRUE`) is the complement of the scaled complexity:

$$CHof.S = \frac{p - CHof}{p - 1} = 1 - CHof.R$$

Values close to 1 indicate high factorial simplicity; values close to 0 indicate high complexity.

The Hofmann complexity index for item i is computed as:

$$CHof_i = \frac{(\sum_{j=1}^{p} \lambda_{ij}^2)^2}{\sum_{j=1}^{p} \lambda_{ij}^4}$$

where λ_{ij} is the loading of item i on factor j, and p is the number of factors.

The scaled versions are useful for comparing items across studies with different numbers of factors, as they are bounded between 0 and 1. They complement other simplicity indices such as Bentler's or Fleming's.

Value

A data frame with the original `CHof` column and optionally `CHof.R` (scaled complexity) and/or `CHof.S` (scaled simplicity) depending on the arguments.

References

Hofmann, R. J. (1977). Indices descriptive of factor complexity. *The Journal of General Psychology*, 96(1), 103-110. doi: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. doi: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. doi:10.1016/j.jrp.2010.03.002

Examples

```
# Simulated factor loadings (3 factors)
loadings <- 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)
)

# Original Hofmann index only
Hofmann(loadings)

# With scaled complexity (0-1)
Hofmann(loadings, scaledVersion = TRUE)

# With simplicity (0-1)
Hofmann(loadings, simplVersion = TRUE)

# Both scaled versions
Hofmann(loadings, scaledVersion = TRUE, simplVersion = TRUE)
```

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

∗ **datasets**
fullclean, 13

boot, *3*
bootLoadings, 2, *6–10*
BSI, 5, *6*

ciBSI, *4*, 6
ciHofmann, *4*, 7
ciLSI, *4*, 8
ciSimload, *4*, 9
ciSSindices, *4*, 10

entropyFL, 11

fullclean, 13

Hofmann, *7*, 14
HofmannFac, *3*, *7*, 15

KC, 17

LSIglobal, *8*, 18

plotFacomplex, 19
print.KC, 21
profileFacomplex, 21

simload, *3*, *9*, 23
SSindices, *10*, 25