Package 'bifactor'

June 8, 2022

Type Package

Title Exploratory Factor, Bi-factor, and Generalized Bi-factor Modeling

Version 0.1.0

Date 2022-05-04

Description Fit exploratory factor, bi-factor, and Generalized Bi-factor Models.

License GPL-3

LazyData TRUE

Depends R (>= 4.0)

Imports Rcpp, Rcsdp

URL `https://github.com/Marcosjnez/bifactor`

BugReports `https://github.com/Marcosjnez/bifactor/issues`

Encoding UTF-8

RoxygenNote 7.2.0

LinkingTo Rcpp, RcppArmadillo

NeedsCompilation yes

Author Marcos Jimenez [aut, cre, cph],
Francisco J. Abad [aut, cph],
Eduardo Garcia-Garzon [aut, cph],
Luis E. Garrido [aut, cph]
Vithor R. Franco [aut, cph]

Maintainer Marcos Jimenez `<marcosjnezhquez@gmail.com>`

R topics documented:

---

asymp_cov *Asymptotic standard errors for correlation matrices.*

Description

Get the asymptotic standard errors of correlation matrices of normal or arbitrary random deviates.

Usage

```
asymp_cov(R, X = NULL, eta = 1, type = "normal")
```

Arguments

R	Correlation matrix.
X	Optional raw data matrix.
eta	Skewness parameter for elliptical data distributions.
type	Type of random deviates: "normal", "elliptical" or "general".

Details

If `type = "normal"`, the calculation assumes that the raw data follows a multivariate normal distribution. If `type = "elliptical"`, the calculation assumes that the raw data follows an elliptical distribution with skewness parameter `eta`. If `type = "general"`, no assumption is made but need to provide the raw data via the `X` argument.

Value

The asymptotic covariance matrix of `R`.

References

M.W. Browne and A. Shapiro (1986). The asymptotic covariance matrix of sample correlation coefficients under general conditions. Linear Algebra and its Applications, 82, 169-176. https://doi.org/10.1016/0024-3795(86)90150-3

`bifactor` *Fit an exploratory bi-factor or generalized bi-factor model.*

Description

Fit an exploratory bi-factor or generalized bi-factor model with correlated factors.

Usage

```
bifactor(R, n_generals, n_groups, bifactor_method = "GSLiD",
method = "minres", projection = "oblq", nobs = NULL,
PhiTarget = NULL, PhiWeight = NULL,
blocks = NULL, blocks_list = NULL, block_weights = NULL,
oblq_blocks = NULL, init_Target = NULL, maxit = 20L,
cutoff = 0, w = 1, random_starts = 1L, cores = 1L, init = NULL,
efa_control = NULL, rot_control = NULL,
first_efa = NULL, second_efa = NULL, verbose = TRUE)
```

Arguments

`R`	Correlation matrix.
`n_generals`	Number of general factors to extract.
`n_groups`	Number of group factors to extract.
`bifactor_method`	"GSLiD", "SL" (Schmid-Leiman), and "botmin" (bifactor-oblimin-target minimization). Defaults to "GSLiD".
`method`	EFA fitting method: "ml" (maximum likelihood for multivariate normal variable), "minres" (minimum residuals), "pa" (principal axis) or "minrank" (minimum rank). Defaults to "minres".
`projection`	Projection method. Available projections: "orth" (orthogonal), "oblq" (oblique) and "poblq" (partially oblique). Defaults to "oblq".
`nobs`	Sample size. Defaults to NULL.
`PhiTarget`	Target matrix for the factor correlations. Defaults to NULL.
`PhiWeight`	Weight matrix for the factor correlations. Defaults to NULL.
`blocks`	Vector with the number of factors for which separately applying the rotation criterion. Defaults to NULL.
`blocks_list`	List containing the columns to which applying the rotation criterion.
`block_weights`	Vector of weights for each block of factors.

oblq_blocks	Vector with the number of factors for each oblique block. E.g.: c(2, 4) means that there are two blocks of oblique factors: one block with 2 factors and another block with 4 factors. Everything else is orthogonal. Defaults to NULL.
init_Target	Initial target matrix for the loadings. Defaults to NULL.
maxit	Maximum number of iterations for the GSLiD algorithm. Defaults to 20L.
cutoff	Cut-off used to update the target matrix upon each iteration. Defaults to 0.
w	w parameter for the extended target criterion ("xtarget"). Defaults to 1L.
random_starts	Number of rotations with different random starting values. The rotation with the smallest cost function value is returned. Defaults to 1L.
cores	Number of cores for parallel execution of multiple rotations. Defaults to 1L.
init	Initial uniquenesses values for exploratory factor analysis estimation. Defaults to NULL.
efa_control	List of control parameters for efa fitting. Defaults to NULL.
rot_control	List of control parameters for the rotation algorithm. Defaults to NULL.
first_efa	List of arguments to pass to `efast` to perform the first-order solution for the Schmid-Leiman method. Defaults to NULL.
second_efa	List of arguments to pass to `efast` to perform the second-order solution for the Schmid-Leiman method. Defaults to NULL.
verbose	Print the convergence progress information. Defaults to TRUE.

Details

If `efa.control = NULL`, then `list(maxit = 1e4)` is passed to `efa.control`. If `rot_control = NULL`, then `list(maxit = 1000, eps = 1e-05)` is passed to `rot_control`, where `eps` is the absolute tolerance. When the objective function does not make a larger improvement than `eps`, the algorithm is assumed to converge.

If `Target` is provided but not `Weight`, then `Weight = 1 - Target` by default, which means a partially specified target rotation is performed. The same applies for `PhiTarget` and `PhiWeight`.

If `init = NULL`, then the squared multiple correlations of each item with the remaining ones are used as initial values (These are known to be upper bounds).

If `init_Target` is provided, then an initial target by means of the Schmid-Leiman transformation is not necessary.

If `cutoff` is not 0, loadings smaller than such a cut-off are fixed to 0. When `cutoff = 0`, an empirical cut-off is used for each column of the loading matrix. They are the mean of the one-lagged differences of the sorted squared normalized loadings. Then, the target is determined by fixing to 0 the squared normalized loadings smaller than such cut-offs.

Value

List of class `bifactor`.

efa	List containing objects related to the exploratory factor analysis estimation. See `efast`.
bifactor	List with the following components:

- loadings - Rotated loading matrix.
- Phi - Factor correlation matrix.
- T - Transformation matrix.
- f - Objective value at the minimum.
- iterations - Number of iterations performed by the rotation algorithm.
- convergence - Convergence of the rotation algorithm.
- uniquenesses - Vector of uniquenesses.
- Rhat - Correlation matrix predicted by the model.
- Target - Updated target matrix.
- Weight - Weight matrix. It is the complement of the updated target.
- GSLiD_iterations - Number of iterations performed by the GSLiD algorithm.
- GSLiD_convergence - Convergence of the GSLiD algorithm.
- min_congruences - Vector containing, for each iteration, the minimum Tucker's congruence between the current loading matrix and the previous loading matrix.
- max_abs_diffs - Vector containing, for each iteration, the maximum absolute difference between the current loading matrix and the previous loading matrix.

`elapsed` Total amount of time spent for execution (in nanoseconds).

References

Jiménez, M., Abad, F.J., Garcia-Garzon, E., Garrido, L.E. (2021, June 24). Generalized exploratory bi-factor Modeling. Under review. Retrieved from https://osf.io/7aszj/?view_only=8f7bd98025104347a96f60a6736f5a64

Examples

```
## Not run: # Simulate data:
sim <- sim_factor(n_generals = 3, groups_per_general = 5, items_per_group = 6,
generals_rho = 0.3)
scores <- MASS::mvrnorm(1e4, rep(0, nrow(sim$R)), Sigma = sim$R)
s <- cor(scores)

# Fit an Generalized exploratory bi-factor model with GSLiD:
GSLiD <- bifactor(s, n_generals = 3, n_groups = 15, bifactor_method = "GSLiD",
method = "minres", projection = "poblq", nobs = NULL, oblq_blocks = 3,
random_starts = 10, cores = 8, w = 1, maxit = 20, verbose = TRUE)

## End(Not run)
```

check_deriv *Check the derivatives and differentials of rotation criteria.*

Description

Check the derivatives and differentials of rotation criteria.

Usage

```
check_deriv(L, Phi, dL, dP, rotation = "oblimin", projection = "oblq",
Target = NULL, Weight = NULL, PhiTarget = NULL, PhiWeight = NULL,
blocks = NULL, blocks_list = NULL, block_weights = NULL,
oblq_blocks = NULL, between_blocks = "none", gamma = 0,
epsilon = 0.01, k = 0L, w = 1, alpha = 1, a = 30, b = 0.36)
```

Arguments

`L`	Loading matrix.
`Phi`	Factor correlation matrix.
`dL`	Perturbation for L.
`dP`	Perturbation for Phi.
`rotation`	Rotation criterion. Available rotations: "varimax", "cf" (Crawford-Ferguson), "oblimin", "geomin", "target", "xtarget" (extended target) and "none". Defaults to "oblimin".
`projection`	Projection method. Available projections: "orth" (orthogonal), "oblq" (oblique), "poblq" (partially oblique). Defaults to "oblq".
`Target`	Target matrix for the loadings. Defaults to NULL.
`Weight`	Weight matrix for the loadings. Defaults to NULL.
`PhiTarget`	Target matrix for the factor correlations. Defaults to NULL.
`PhiWeight`	Weight matrix for the factor correlations. Defaults to NULL.
`blocks`	Vector with the number of factors for which separately applying the rotation criterion. Defaults to NULL.
`blocks_list`	List containing the columns to which applying the rotation criterion.
`block_weights`	Vector of weights for each block of factors.
`oblq_blocks`	Vector with the number of factors for each oblique block. E.g.: c(2, 4) means that there are two blocks of oblique factors: one block with 2 factors and another block with 4 factors. Everything else is orthogonal. Defaults to NULL.
`between_blocks`	Available between_blocks: "TL" and "TLM". Defaults to none.
`gamma`	γ parameter for the oblimin criterion. Defaults to 0 (quartimin).
`epsilon`	ϵ parameter for the geomin criterion. Defaults to 0.01.
`k`	k parameter for the Crawford-Ferguson family of rotation criteria. Defaults to 0.

`w`	w parameter for the extended target criterion ("xtarget"). Defaults to 1L.
`alpha`	α parameter for the Tian and Liu between_blocks. Defaults to 1.
`a`	Discrimination parameter for simplix rotation. Defaults to 30.
`b`	Difficulty parameter for simplix rotation. Defaults to 0.36.

Details

None yet.

Value

A list with the objective value and the gradients and differentials of L and Phi.

`cv_eigen` *Cross-validated eigenvalues.*

Description

Estimate cross-validated eigenvalues and the dimensionality using the Kaiser's rule.

Usage

```
cv_eigen(X, N = 100L, hierarchical = FALSE, efa = NULL, cores = 1L)
```

Arguments

`X`	Raw data matrix.
`N`	Number of cross-validated samples.
`hierarchical`	Logical indicating whether a second cross-validated eigenvalues estimation should be performed from the factor scores obtained after a first factor analysis analysis.
`efa`	A list of arguments to pass to `efast` when `hierarchical = TRUE`.
`cores`	Number of cores to perform parallel computations.

Details

None yet.

Value

A list with the cross-validated eigenvalues and the estimated dimensionality.

References

Chen F., Roch S., Rohe K., Yu S (2021). Estimating Graph Dimension with Cross-validated Eigenvalues, arXiv. https://arxiv.org/abs/2108.03336

efast *Fast exploratory factor analysis.*

Description

Fast exploratory factor analysis.

Usage

```
efast(R, nfactors, method = "minres",
rotation = "oblimin", projection = "oblq", nobs = NULL,
Target = NULL, Weight = NULL, PhiTarget = NULL, PhiWeight = NULL,
blocks = NULL, blocks_list = NULL, block_weights = NULL,
oblq_blocks = NULL, normalization = "none", between_blocks = "none", gamma = 0,
epsilon = 1e-02, k = 0, w = 1, alpha = 1, a = 30, b = 0.36,
random_starts = 1L, cores = 1L,
init = NULL, efa_control = NULL, rot_control = NULL)
```

Arguments

`R`	Correlation matrix.
`nfactors`	Number of common factors to extract.
`method`	EFA fitting method: "ml" (maximum likelihood for multivariate normal variables), "minres" (minimum residuals), "pa" (principal axis) and "minrank" (minimum rank). Defaults to "minres".
`rotation`	Rotation criterion. Available rotations: "varimax", "cf" (Crawford-Ferguson), "oblimin", "geomin", "target", "xtarget" (extended target) and "none". Defaults to "oblimin".
`projection`	Projection method. Available projections: "orth" (orthogonal), "oblq" (oblique), "poblq" (partially oblique). Defaults to "oblq".
`nobs`	Sample size. Defaults to NULL.
`Target`	Target matrix for the loadings. Defaults to NULL.
`Weight`	Weight matrix for the loadings. Defaults to NULL.
`PhiTarget`	Target matrix for the factor correlations. Defaults to NULL.
`PhiWeight`	Weight matrix for the factor correlations. Defaults to NULL.
`blocks`	Vector with the number of factors for which separately applying the rotation criterion. Defaults to NULL.
`blocks_list`	List containing the columns to which applying the rotation criterion.
`block_weights`	Vector of weights for each block of factors.
`oblq_blocks`	Vector with the number of factors for each oblique block. E.g.: c(2, 4) means that there are two blocks of oblique factors: one block with 2 factors and another block with 4 factors. Everything else is orthogonal. Defaults to NULL.
`normalization`	Available normalizations: "kaiser". Defaults to "none".

between_blocks Available between_blocks: "TL" and "TLM". Defaults to none.

gamma γ parameter for the oblimin criterion. Defaults to 0 (quartimin).

epsilon ϵ parameter for the geomin criterion. Defaults to 0.01.

k k parameter for the Crawford-Ferguson family of rotation criteria. Defaults to 0.

w w parameter for the extended target criterion ("xtarget"). Defaults to 1L.

alpha α parameter for the Tian and Liu between_blocks. Defaults to 1.

a Discrimination parameter for simplix rotation. Defaults to 30.

b Difficulty parameter for simplix rotation. Defaults to 0.36.

random_starts Number of rotations with different random starting values. The rotation with the smallest cost function value is returned. Defaults to 1L.

cores Number of cores for parallel execution of random starts. Defaults to 1L.

init Initial uniquenesses values for exploratory factor analysis estimation. Defaults to NULL.

efa_control List of control parameters for efa fitting. Defaults to NULL.

rot_control List of control parameters for the rotation algorithm. Defaults to NULL.

Details

If `efa.control = NULL`, then `list(maxit = 1e4)` is passed to `efa.control`. If `rot_control = NULL`, then `list(maxit = 1000, eps = 1e-05)` is passed to `rot_control`, where `eps` is the absolute tolerance. When the objective function does not make a larger improvement than `eps`, the algorithm is assumed to converge.

If `Target` is provided but not `Weight`, then `Weight = 1 - Target` by default, which means a partially specified target rotation is performed. The same applies for `PhiTarget` and `PhiWeight`.

If `init = NULL`, then the squared multiple correlations of each item with the remaining ones are used as initial values (These are known to be upper bounds).

If a Heywood case is encountered, then `method = "minrank"` is automatically applied to ensure positive uniquenesses.

Value

List of class `efast` with the following components:

efa List containing the following objects:

- loadings - Unrotated loadings.
- uniquenesses - Vector of uniquenesses.
- Rhat - Correlation matrix predicted by the model.
- residuals - Residual correlation matrix.
- f - Objective value at the minimum.
- Heywood - TRUE if any Heywood case is encountered and FALSE otherwise.
- iterations - Number of iterations for the L-BFGS-B algorithm to converge.

- convergence - TRUE if the L-BFGS-B algorithm converged and FALSE otherwise.
- method - Method used to fit the exploratory factor analysis.

`rotation`	List of class `rotation`. Only if the argument `rotation` is not "none". See `rotate` for the components.
`elapsed`	Total amount spent for execution (in nanoseconds).

References

Jiménez, M., Abad, F.J., Garcia-Garzon, E., Garrido, L.E. (2021, June 24). Generalized exploratory bi-factor Modeling. Under review. Retrieved from https://osf.io/7aszj/?view_only=8f7bd98025104347a96f60a6736f5a64

Examples

```
## Not run:
# Simulate data:
sim <- sim_factor(n_generals = 0, groups_per_general = 5, items_per_group = 6)
scores <- MASS::mvrnorm(1e3, rep(0, nrow(sim$R)), Sigma = sim$R)
s <- cor(scores)

# Fit efa:
efa <- efast(s, nfactors = 5, method = "minres", rotation = "oblimin",
projection = "oblq", gamma = 0, random_starts = 10L, cores = 1L)

## End(Not run)
```

`fitMeasures` *Compute fit measures for exploratory factor models.*

Description

Compute fit measures for exploratory factor models.

Usage

```
fitMeasures(efa, nobs=NULL)
```

Arguments

`efa`	Object of class efa.
`nobs`	Sample size. Defaults to NULL.

Details

`fitMeasures`... to be explained

Value

Vector of fit measures.

Author(s)

Vithor R. Franco & Marcos Jiménez

`get_target` *Get a target from a loading matrix.*

Description

Get a target for the loading matrix using a custom or empirical cut-off.

Usage

```
get_target(loadings, Phi = NULL, cutoff = 0)
```

Arguments

`loadings`	A matrix of loadings.
`Phi`	A correlation matrix among the factors. Defaults to NULL.
`cutoff`	The cut-off used to create the target matrix. Defaults to 0.

Details

If `cutoff` is not 0, loadings smaller than such a cut-off are fixed to 0. When `cutoff = 0`, an empirical cut-off is used for each column of the loading matrix. They are the mean of the one-lagged differences of the sorted squared normalized loadings. Then, the target is determined by fixing to 0 the squared normalized loadings smaller than such cut-offs.

Value

A target matrix.

References

Garcia-Garzon, E., Abad, F. J., & Garrido, L. E. (2019). Improving bi-factor exploratory modeling: Empirical target rotation based on loading differences. Methodology: European Journal of Research Methods for the Behavioral and Social Sciences, 15(2), 45–55. https://doi.org/10.1027/1614-2241/a000163

Jiménez, M., Abad, F.J., Garcia-Garzon, E., Garrido, L.E. (2021, June 24). Generalized exploratory bi-factor Modeling. Under review. Retrieved from https://osf.io/7aszj/?view_only=8f7bd98025104347a96f60a6736f5a64

parallel — *Hierarchical parallel analysis using either principal components (PCA) or principal axis factoring (PAF).*

Description

Perform hierarchical parallel analysis to detect dimensionality using either principal components or principal axis factoring.

Usage

```
parallel(X, n_boot = 100L, quant = NULL, mean = TRUE, replace = FALSE,
PA = NULL, hierarchical = FALSE, efa = NULL, cores = 1L)
```

Arguments

`X`	Raw data matrix.
`n_boot`	Number of bootstrap samples.
`quant`	Vector of quantiles of the distribution of bootstrap eigenvalues to which the compare the sample eigenvalues.
`mean`	Logical. Compare the sample eigenvalues to the mean of the bootstrap eigenvalues. Defaults to TRUE.
`replace`	Logical indicating whether the columns of `X` should be permuted with replacement.
`PA`	Parallel analysis method. It can be either principal components ("PCA"), principal axis ("PAF") or both ("PCA" and "PAF"). Defaults to NULL, which sets c("PCA", "PAF").
`hierarchical`	Logical indicating whether a second parallel analysis should be performed from the factor scores obtained after a first factor analysis analysis.
`efa`	A list of arguments to pass to `efast` when `hierarchical = TRUE`.
`cores`	Number of cores to perform the parallel bootstrapping.

Details

Not yet.

Value

A list with the bootstrapped eigenvalues and the estimated dimensionality.

References

Horn, J. L. (1965). A Rationale and Test For the Number of Factors in Factor Analysis, Psychometrika, 30, 179-85. https://doi.org/10.1007/BF02289447

`print.bifactor` *Print summary information from exploratory factor models.*

Description

Print summary information from exploratory factor models.

Usage

```
## S3 method for class 'bifactor'
print(bifactor, nobs=NULL, ...)
print.bifactor(bifactor, nobs=NULL, ...)
```

Arguments

`nobs`	Sample size. Defaults to NULL.
`...`	Arguments to be passed to or from other methods.
`efa`	Object of class bifactor

Details

to be explained

Value

Matrix of variance accounted for the factors.

Author(s)

Vithor R. Franco & Marcos Jiménez

`print.efa` *Print summary information from exploratory factor models.*

Description

Print summary information from exploratory factor models.

Usage

```
## S3 method for class 'efa'
print(efa, nobs=NULL, ...)
print.efa(efa, nobs=NULL, ...)
```

Arguments

`efa`	Object of class efa.
`nobs`	Sample size. Defaults to NULL.
`...`	Arguments to be passed to or from other methods.

Details

to be explained

Value

Matrix of variance accounted for the factors.

Author(s)

Vithor R. Franco & Marcos Jiménez

`random_oblq` *Generate random oblique matrices.*

Description

Generate random oblique matrices from a standard normal distribution.

Usage

```
random_oblq(p, q)
```

Arguments

`p`	Number of rows.
`q`	Number of columns. Should not be greater than p.

Value

An oblique matrix with normally distributed data.

References

Jiménez, M., Abad, F.J., Garcia-Garzon, E., Garrido, L.E. (2021, June 24). Generalized exploratory bi-factor Modeling. Under review. Retrieved from https://osf.io/7aszj/?view_only=8f7bd98025104347a96f60a6736f5a64

`random_orth` *Generate random orthogonal matrices.*

Description

Generate random orthogonal matrices from a standard normal distribution. First, a matrix of random standard normal variables is simulated and then, the Q factor from the QR decomposition is returned.

Usage

```
random_orth(p, q)
```

Arguments

`p`	Number of rows.
`q`	Number of columns. Should not be greater than p.

Value

An orthogonal matrix with normally distributed data.

References

Jiménez, M., Abad, F.J., Garcia-Garzon, E., Garrido, L.E. (2021, June 24). Generalized exploratory bi-factor Modeling. Under review. Retrieved from https://osf.io/7aszj/?view_only=8f7bd98025104347a96f60a6736f5a64

`random_poblq` *Generate a random partially oblique matrix.*

Description

First, a matrix is simulated from a standard normal distribution. Second,the matrix is normalized and the Gram-Schmidt process is performed between the oblique blocks. Finally, the orthogonal blocks correspond to those columns of the Q matrix from the QR decomposition.

Usage

```
random_poblq(p, q, oblq_blocks)
```

Arguments

`p`	Number of rows.
`q`	Number of columns. Should not be greater than p.
`oblq_blocks`	A vector with the number of factors for each oblique block. E.g.: c(2, 4) means that there are two blocks of oblique factors: one with 2 factors and another with 4 factors. Everything else is orthogonal.

Value

A partially oblique matrix.

References

Jiménez, M., Abad, F.J., Garcia-Garzon, E., Garrido, L.E. (2021, June 24). Generalized exploratory bi-factor Modeling. Under review. Retrieved from https://osf.io/7aszj/?view_only=8f7bd98025104347a96f60a6736f5a64

Examples

```
random_poblq(p = 7, q = 7, oblq_blocks = c(3, 2))
```

---

retr_oblq *Retraction of a matrix onto the oblique manifold.*

Description

Transform a matrix into an oblique matrix.

Usage

```
retr_oblq(X)
```

Arguments

X	A matrix.

Value

An oblique matrix.

References

Jiménez, M., Abad, F.J., Garcia-Garzon, E., Garrido, L.E. (2021, June 24). Generalized exploratory bi-factor Modeling. Under review. Retrieved from https://osf.io/7aszj/?view_only=8f7bd98025104347a96f60a6736f5a64

retr_orth *Retraction of a matrix onto the orthogonal manifold.*

Description

Transform a matrix into an orthogonal matrix.

Usage

```
retr_orth(X)
```

Arguments

`X`	A matrix.

Value

An orthogonal matrix.

References

Jiménez, M., Abad, F.J., Garcia-Garzon, E., Garrido, L.E. (2021, June 24). Generalized exploratory bi-factor Modeling. Under review. Retrieved from https://osf.io/7aszj/?view_only=8f7bd98025104347a96f60a6736f5a64

retr_poblq *Retraction of a matrix onto the partially oblique manifold.*

Description

Transform a matrix into a partially oblique matrix.

Usage

```
retr_poblq(X, oblq_blocks)
```

Arguments

`X`	A matrix.
`oblq_blocks`	A vector with the number of factors for each oblique block. E.g.: c(2, 4) means that there are two blocks of oblique factors: one with 2 factors and another with 4 factors. Everything else is orthogonal.

Value

A partially oblique matrix.

References

Jiménez, M., Abad, F.J., Garcia-Garzon, E., Garrido, L.E. (2021, June 24). Generalized exploratory bi-factor Modeling. Under review. Retrieved from https://osf.io/7aszj/?view_only=8f7bd98025104347a96f60a6736f5a64

Examples

```
X <- replicate(8, rnorm(8))
retr_poblq(X, c(2, 3, 3))
```

`rotate` *Fast rotation algorithm for factor analysis.*

Description

Riemannian Newton Trust-Region algorithm to quickly perform (parallel) rotations with different random starting values.

Usage

```
rotate(loadings, rotation = "oblimin", projection = "oblq",
gamma = 0, epsilon = 0.01, k = 0, w = 1, alpha = 1, a = 30, b = 0.36,
Target = NULL, Weight = NULL, PhiTarget = NULL, PhiWeight = NULL,
blocks = NULL, blocks_list = NULL, block_weights = NULL, oblq_blocks = NULL,
between_blocks = "none", normalization = "none",
rot_control = NULL, random_starts = 1L, cores = 1L)
```

Arguments

`loadings`	Unrotated loading matrix.
`rotation`	Rotation criterion. Available rotations: "varimax", "cf" (Crawford-Ferguson), "oblimin", "geomin", "target", "xtarget" (extended target) and "none". Defaults to "oblimin".
`projection`	Projection method. Available projections: "orth" (orthogonal), "oblq" (oblique), "poblq" (partially oblique). Defaults to "oblq".
`gamma`	γ parameter for the oblimin criterion. Defaults to 0 (quartimin).
`epsilon`	ϵ parameter for the geomin criterion. Defaults to 0.01.
`k`	k parameter for the Crawford-Ferguson family of rotation criteria. Defaults to 0.
`w`	w parameter for the extended target criterion ("xtarget"). Defaults to 1.
`alpha`	α parameter for the Tian and Liu between_blocks. Defaults to 1.
`a`	Discrimination parameter for simplix rotation. Defaults to 30.
`b`	Difficulty parameter for simplix rotation. Defaults to 0.36.
`Target`	Target matrix for the loadings. Defaults to NULL.

`Weight`	Weight matrix for the loadings. Defaults to NULL.
`PhiTarget`	Target matrix for the factor correlations. Defaults to NULL.
`PhiWeight`	Weight matrix for the factor correlations. Defaults to NULL.
`blocks`	Vector with the number of factors for which separately applying the rotation criterion. Defaults to NULL.
`blocks_list`	List containing the vectors with the columns to which applying the rotation criterion.
`block_weights`	Vector of weights for each block of factors.
`oblq_blocks`	Vector with the number of factors for each oblique block. E.g.: c(2, 4) means that there are two blocks of oblique factors: one block with 2 factors and another block with 4 factors. Everything else is orthogonal. Defaults to NULL.
`between_blocks`	Available between_blocks: "TL" and "TLM". Defaults to none.
`normalization`	Available normalizations: "kaiser". Defaults to "none".
`rot_control`	List of control parameters for the rotation algorithm. Defaults to NULL.
`random_starts`	Number of rotations with different random starting values. The rotation with the smallest cost function value is returned. Defaults to 1L.
`cores`	Number of cores for parallel execution of random starts. Defaults to 1L.

Details

If `rot_control = NULL`, then `list(maxit = 1000, eps = 1e-05)` is passed to `rot_control`, where `eps` is the absolute tolerance. When the objective function does not make a larger improvement than `eps`, the algorithm is assumed to converge. If `Target` is provided but not `Weight`, then `Weight = 1 - Target` by default, which means a partially specified target rotation is performed. The same applies for `PhiTarget` and `PhiWeight`.

Value

List of class `rotation` with the following components:

`loadings`	Rotated loading matrix.
`Phi`	Correlation matrix among the factors.
`T`	Rotation matrix.
`f`	Objective value at the minimum.
`iterations`	Number of iterations for the rotation algorithm to converge.
`convergence`	TRUE if the algorithm converged and FALSE otherwise.
`elapsed`	Total amount of time spent for execution (in nanoseconds).

References

Jiménez, M., Abad, F.J., Garcia-Garzon, E., Garrido, L.E. (2021, June 24). Generalized exploratory bi-factor Modeling. Under review. Retrieved from https://osf.io/7aszj/?view_only=8f7bd98025104347a96f60a6736f5a64

Zhang, G., Hattori, M., Trichtinger, L. A., & Wang, X. (2019). Target rotation with both factor loadings and factor correlations. Psychological Methods, 24(3), 390–402. https://doi.org/10.1037/met0000198

se *Standard errors for rotated factor loadings, factor correlations and uniquenesses.*

Description

Compute the sandwich standard errors of factor loadings, factor correlations and uniquenesses.

Usage

```
se(fit = NULL, n = NULL, X = NULL, type = "normal", eta = 1)
```

Arguments

`fit`	Optional `efast` model.
`n`	Sample size.
`X`	Raw data matrix.
`type`	Type of random deviates: "normal", "elliptical" or "general".
`eta`	Skewness parameter for elliptical data distributions.

Details

Currently, only available for `method` = `minres`.

Value

A list with the standard errors of the rotated factor loadings, factor correlations and uniquenesses.

References

Zhang G, Preacher KJ, Hattori M, Jiang G, Trichtinger LA (2019). A sandwich standard error estimator for exploratory factor analysis with nonnormal data and imperfect models. Applied Psychological Measurement, 43, 360–373. https://doi.org/10.1177/0146621618798669

sim_factor *Simulate a bi-factor or generalized bifactor population structure.*

Description

Simulate a bi-factor or generalized bifactor population structure with cross-loading, pure items and correlated factors.

Usage

```
sim_factor(n_generals, groups_per_general, items_per_group,
loadings_g = "medium", loadings_s = "medium",
crossloadings = 0, pure = FALSE,
generals_rho = 0, groups_rho = 0, confirmatory = TRUE,
method = "minres", fit = "rmsr", misfit = 0, error_method = "yuan")
```

Arguments

`n_generals`	Number of general factors.
`groups_per_general`	Number of group factors per general factor.
`items_per_group`	Number of items per group factor.
`loadings_g`	Loadings' magnitude on the general factors: "low", "medium" or "high". Defaults to "medium".
`loadings_s`	Loadings' magnitude on the group factors: "low", "medium" or "high". Defaults to "medium".
`crossloadings`	Magnitude of the cross-loadings among the group factors. Defaults to 0.
`pure`	Fix a pure item on each general factor. Defaults to FALSE.
`generals_rho`	Correlation among the general factors. Defaults to 0.
`groups_rho`	Correlation among the group factors. Defaults to 0.
`confirmatory`	Logical. Should the misfit value be computed according to a confirmatory model (TRUE) or an exploratory model (FALSE). Defaults to TRUE.
`method`	Method used to generate population error: "minres" or "ml".
`fit`	Fit index to control the population error.
`misfit`	Misfit value to generate population error.
`error_method`	Method used to control population error: c("yuan", "cudeck"). Defaults to "yuan".

Details

`sim_factor` generates bi-factor and generalized bifactor patterns with cross-loadings, pure items and correlations among the general and group factors. When `crossloading` is different than 0, one cross-loading is introduced for an item pertaining to each group factor. When `pure` is TRUE, one item loading of each group factor is removed so that the item loads entirely on the general factor. To maintain the item communalities constant upon these modifications, the item loading on the other factors may shrunk (if adding cross-loadings) or increase (if setting pure items).

Loading magnitudes may range between 0.3-0.5 ("low"), 0.4-0.6 ("medium") and 0.5-0.7 ("high").

Value

List with the following objects:

`lambda`	Population loading matrix.

Phi	Population factor correlation matrix.
R	Population correlation matrix.
R_error	Population correlation matrix with error.
uniquenesses	Vector of population uniquenesses.
delta	Minimum of the objective function that correspond to the misfit value.

References

Jiménez, M., Abad, F.J., Garcia-Garzon, E., Garrido, L.E. (2021, June 24). Exploratory generalized bifactor Modeling. Under review. Retrieved from https://osf.io/7aszj/?view_only=8f7bd98025104347a96f60a6736f5a64

---

sl — *Schmid-Leiman Transformation.*

Description

Schmid-Leiman transformation into a bi-factor or generalized bi-factor pattern.

Usage

```
sl(R, n_generals, n_groups, nobs = NULL, first_efa = NULL, second_efa = NULL)
```

Arguments

R	Correlation matrix.
n_generals	Number of general factors.
n_groups	Number of group factors.
nobs	Sample size. Defaults to NULL.
first_efa	Arguments to pass to `efast` in the first-order factor extraction. See `efast` for the default arguments.
second_efa	Arguments to pass to `efast` in the second-order factor extraction. See `efast` for the default arguments.

Details

First, a hierarchical factor model is fitted using a second-order factor analysis on the factor correlation obtained from a first-order factor analysis. Then, the item loadings on the general factors are assumed to be the direct effects of the general factors according to such hierarchical model. On the other hand, the item loadings on the group factors become the originally first-order loadings post-multiplied by the diagonal matrix containing the root of the item uniquenesses.

Obviously, the first-order factor analysis should be oblique to perform a second exploratory factor analysis.

If the second-order solution does not use an orthogonal projection, then the correlation matrix among the general factors for the Schmid-Leiman solution is simply that obtained from such second-order solution.

Value

`loadings` Loading matrix of the Schmid-Leiman solution.

`first_order_solution`
Object of class `efast` with the first-order solution.

`second_order_solution`
Object of class `efast` with the second-order solution.

`uniquenesses` Vector of uniquenesses.

`Rhat` Correlation matrix predicted by the (hierarchical) model.

References

Jiménez, M., Abad, F.J., Garcia-Garzon, E., Garrido, L.E. (2021, June 24). Generalized exploratory bi-factor Modeling. Under review. Retrieved from https://osf.io/7aszj/?view_only=8f7bd98025104347a96f60a6736f5a64

Examples

```
## Not run:
# Simulate data:
sim <- sim_factor(n_generals = 2, groups_per_general = 3, items_per_group = 5)
lambda <- sim$lambda
Target <- ifelse(lambda > 0, 1, 0)

# Target rotation for the first-order efa and oblimin for the second-order efa:
first <- list(rotation = "target", projection = "oblq", Target = Target[, -c(1:2)])
second <- list(rotation = "oblimin", projection = "oblq", gamma = 0)

SL <- sl(sim$R, n_generals = 2, n_groups = 6, nobs = 100, first, second)

## End(Not run)
```

`summary.bifactor` *Compute summaries, fit indices, and information criteria from exploratory factor models.*

Description

Compute summaries, fit indices, and information criteria from exploratory factor models.

Usage

```
## S3 method for class 'bifactor'
summary(bifactor, nobs=NULL, suppress=0, order=FALSE, ...)
summary.bifactor(bifactor, nobs=NULL, suppress=0, order=FALSE, ...)
```

Arguments

`bifactor`	Object of class bifactor.
`nobs`	Sample size. Defaults to NULL.
`suppress`	Hide the loadings whose absolute magnitudes are smaller than this cutoff. Defaults to 0.
`order`	Order the columns of the pattern matrix according to the variance they account for. Defaults to FALSE.
`digits`	Number of digits to display in the loading and factor correlation matrices.
`...`	Arguments to be passed to or from other methods.

Details

to be explained

Value

List with the following objects:

`loadings`	Loading matrix.
`Phi`	Factor correlation matrix.
`communalities`	Communalities.
`uniquenesses`	Uniquenesses.
`complexity`	Item complexities.
`VAR`	Variance accounted for the factors.
`fit`	fit indices.

Author(s)

Vithor R. Franco & Marcos Jiménez

`summary.efa` *Compute summaries, fit indices, and information criteria from exploratory factor models.*

Description

Compute summaries, fit indices, and information criteria from exploratory factor models.

Usage

```
## S3 method for class 'efa'
summary(efa, nobs=NULL, suppress=0, order=FALSE, ...)
summary.efa(efa, nobs=NULL, suppress=0, order=FALSE, ...)
```

Arguments

`efa`	Object of class efa.
`nobs`	Sample size. Defaults to NULL.
`suppress`	Hide the loadings whose absolute magnitudes are smaller than this cutoff. Defaults to 0.
`order`	Order the columns of the pattern matrix according to the variance they account for. Defaults to FALSE.
`digits`	Number of digits to display in the loading and factor correlation matrices.
`...`	Arguments to be passed to or from other methods.

Details

to be explained

Value

List with the following objects:

`loadings`	Loading matrix.
`Phi`	Factor correlation matrix.
`communalities`	Communalities.
`uniquenesses`	Uniquenesses.
`complexity`	Item complexities.
`VAR`	Variance accounted for the factors.
`fit`	fit indices.

Author(s)

Vithor R. Franco & Marcos Jiménez

Index