Package 'GPArotation'

May 7, 2026

Version 2026.4-1

Title Gradient Projection Factor Rotation

Depends R (>= 3.5.0)

Description Gradient projection algorithms for orthogonal and oblique rotation of factor loadings matrices in factor analysis. Implements a comprehensive set of rotation criteria including quartimax, quartimin, oblimin, geomin, simplimax, the Crawford-Ferguson family, and target rotation, among others. Supports multiple random starts. For details see Bernaards and Jennrich (2005) <doi:10.1177/0013164404272507>.

License GPL (>= 2)

URL https://cran.r-project.org/package=GPArotation

Imports stats

LazyData yes

NeedsCompilation no

Author Coen Bernaards [aut, cre],
Paul Gilbert [aut],
Robert Jennrich [aut]

Maintainer Coen Bernaards <cab.gparotation@gmail.com>

Repository CRAN

Date/Publication 2026-04-29 07:30:19 UTC

Contents

`00.GPArotation` *Gradient Projection Algorithms for Factor Rotation*

Description

GPA Rotation for Factor Analysis

The GPArotation package contains functions for the rotation of factor loadings matrices. The functions implement Gradient Projection (GP) algorithms for orthogonal and oblique rotation. Additionally, a number of rotation criteria are provided. The GP algorithms minimize the rotation criterion function and provide the corresponding rotation matrix. For oblique rotation, the covariance/correlation matrix of the factors is also provided. The rotation criteria implemented in this package are described in Bernaards and Jennrich (2005). Theory of the GP algorithm is described in Jennrich (2001, 2002).

Additionally, two rotation methods are provided that do not rely on GP (`eiv` and `echelon`).

Four vignettes are provided covering general usage, local minima diagnostics, bifactor rotation and reliability, and derivative-free gradient projection. Access them via Access them via `browseVignettes("GPArotation")`.

Package:	GPArotation
Depends:	R (>= 3.5.0)
License:	GPL Version 2.

Index of functions:

Rotations using gradient projection algorithms

`oblimin`	Oblimin rotation
`quartimin`	Quartimin rotation
`targetT`	Orthogonal target rotation
`targetQ`	Oblique target rotation
`pstT`	Orthogonal partially specified target rotation
`pstQ`	Oblique partially specified target rotation
`oblimax`	Oblimax rotation
`entropy`	Minimum entropy rotation
`quartimax`	Quartimax rotation
`Varimax`	Varimax rotation
`simplimax`	Simplimax rotation
`bentlerT`	Orthogonal Bentler invariant pattern simplicity rotation

bentlerQ	Oblique Bentler invariant pattern simplicity rotation
tandemI	Tandem criteria principle I rotation
tandemII	Tandem criteria principle II rotation
geominT	Orthogonal Geomin rotation
geominQ	Oblique Geomin rotation
bigeominT	Orthogonal Bi-Geomin rotation
bigeominQ	Oblique Bi-Geomin rotation
cfT	Orthogonal Crawford-Ferguson family rotation
cfQ	Oblique Crawford-Ferguson family rotation
equamax	Equamax rotation
parsimax	Parsimax rotation
infomaxT	Orthogonal Infomax rotation
infomaxQ	Oblique Infomax rotation
mccammon	McCammon minimum entropy ratio rotation
varimin	Varimin rotation
bifactorT	Orthogonal bifactor rotation
bifactorQ	Oblique bifactor rotation
lpT	Orthogonal L^p rotation
lpQ	Oblique L^p rotation

Other rotations not using gradient projection algorithms

eiv	Errors-in-variables rotation
echelon	Echelon rotation
varimax	varimax [The R Stats Package]
promax	promax [The R Stats Package]

Core gradient projection algorithms

GPForth	Orthogonal rotation function
GPFoblq	Oblique rotation function

Random-start wrappers and internal engine

GPFRSorth	Random-start wrapper for orthogonal rotation
GPFRSoblq	Random-start wrapper for oblique rotation
.GPA_RS_engine	Internal random-start engine (not exported)

Legacy gradient projection algorithms (code unchanged since 2008)

GPForth.legacy	Orthogonal rotation, original implementation (not exported)
GPFoblq.legacy	Oblique rotation, original implementation (not exported)

Utility functions

print.GPArotation	Print results (S3 method)
summary.GPArotation	Summary of results (S3 method)
.sortGPALoadings	Sort and sign-correct factors (not exported)
Random.Start	Random starting matrix for factor rotation
NormalizingWeight	Normalizing weights utility (not exported)
GPForth.lp	Single-start L^p orthogonal rotation
GPFoblq.lp	Single-start L^p oblique rotation

Rotation criterion functions (not exported)

vgQ.oblimin	Oblimin
vgQ.quartimin	Quartimin
vgQ.target	Target
vgQ.pst	Partially specified target
vgQ.oblimax	Oblimax
vgQ.entropy	Minimum entropy
vgQ.quartimax	Quartimax
vgQ.varimax	Varimax
vgQ.simplimax	Simplimax
vgQ.bentler	Bentler invariant pattern simplicity
vgQ.tandemI	Tandem criteria principle I
vgQ.tandemII	Tandem criteria principle II
vgQ.geomin	Geomin
vgQ.bigeomin	Bi-Geomin
vgQ.cf	Crawford-Ferguson family
vgQ.infomax	Infomax
vgQ.mccammon	McCammon minimum entropy ratio
vgQ.varimin	Varimin
vgQ.bifactor	Bifactor
vgQ.lp.wls	Weighted least squares for L^p rotation

Data sets

Harman8	Harman's 8 physical variables; centroid loadings
NetherlandsTV	Wansbeek and Meijer Netherlands TV viewership; correlation matrix
box26	Thurstone's 26 box variables; unrotated factor loadings

box20	Thurstone's 20 box variables (deprecated, use box26)
CCAI	CCAI Climate-Friendly Purchasing Choices domain; correlation matrix, pattern matrix, and factor intercorr

Vignettes

GPA1guide	Gradient Projection Factor Rotation (main guide)
GPA2local	Assessing Local Minima in Factor Rotation
GPA3bifactor	Bifactor Rotation and Reliability Coefficients

Author(s)

Coen A. Bernaards and Robert I. Jennrich with some R modifications by Paul Gilbert.

References

The software reference is:

Bernaards, C.A. and Jennrich, R.I. (2005). Gradient projection algorithms and software for arbitrary rotation criteria in factor analysis. *Educational and Psychological Measurement*, **65**, 676–696. doi: 10.1177/0013164404272507

Theory of gradient projection algorithms:

Jennrich, R.I. (2001). A simple general procedure for orthogonal rotation. *Psychometrika*, **66**, 289–306. doi: 10.1007/BF02294840

Jennrich, R.I. (2002). A simple general method for oblique rotation. *Psychometrika*, **67**, 7–19. doi: 10.1007/BF02294706

A clear and accessible introduction to gradient projection algorithms for factor rotation is provided in:

Mansolf, M. and Reise, S.P. (2016). Exploratory bifactor analysis: The Schmid-Leiman orthogonalization and Jennrich-Bentler analytic rotations. *Multivariate Behavioral Research*, **51**(5), 698–717. doi: 10.1080/00273171.2016.1215898

See Also

GPFRSorth, GPFRSoblq, rotations, vgQ `browseVignettes("GPArotation")`

CCAI *The CCAI Climate-Friendly Purchasing Choices Domain*

Description

Correlation matrix, factor pattern matrix, and factor intercorrelations for the Climate Change Action Inventory (CCAI) Climate-Friendly Purchasing Choices domain, from Bi and Barchard (2024). The scale measures the frequency with which individuals make purchasing choices aimed at reducing climate change. Data were collected from 500 United States MTurk workers. After 15 climate change deniers and 24 multivariate outliers were removed, 461 participants remained. Climate change deniers were excluded because the scale measures behaviors intended to reduce climate change — including individuals who do not believe climate change exists would be inconsistent with the measure's intent.

Usage

```
data(CCAI)
```

Format

Three objects are loaded by `data(CCAI)`:

- `CCAI_R`: a 14×14 numeric correlation matrix. Row and column names are item labels `CCAI1` through `CCAI14`, ordered by factor to match `CCAI_pattern`.
- `CCAI_pattern`: a 14×3 numeric matrix of factor pattern coefficients. Row names are item labels `CCAI1` through `CCAI14`, ordered by factor to facilitate interpretation. Column names are factor labels `SustainableOptions`, `CollectiveAction`, and `AvoidBuyingNew`.
- `CCAI_Phi`: a 3×3 numeric matrix of factor intercorrelations. Row and column names are the three factor labels.

Details

The CCAI Climate-Friendly Purchasing Choices domain consists of 14 items. Items used a nine-point frequency scale ranging from 1 (less than once a year) to 9 (at least 14 times a week). For full details on study design, data collection, analysis, and interpretation see Bi and Barchard (2024). The Climate Change Action Inventory contains eight domains; the Climate-Friendly Purchasing Choices domain analyzed here is one of them.

`CCAI_R` is the observed 14×14 correlation matrix for the 14 items of the Climate-Friendly Purchasing Choices domain. The correlation matrix was kindly provided by the authors and has not been published separately.

`CCAI_pattern` is a 14×3 factor pattern matrix from principal components extraction followed by direct oblimin rotation, reported in Table 2 of Bi and Barchard (2024) (the Table is labeled "Factor Structure" in the paper but contains pattern coefficients). The three factors are: Choosing Sustainable Options (F1), Supporting Collective Action (F2), and Avoiding Buying New (F3).

Item	Description	F1	F2	F3

CCAI8	Choose products with less impact on climate change	.95	.00	-.02
CCAI6	Choose products with less packaging	.91	-.04	.01
CCAI7	Choose products made locally	.89	-.09	.07
CCAI11	Encourage others to choose climate-friendly products	.56	.40	.05
CCAI12	Problem solve to reduce impact of purchases	.52	.44	.04
CCAI10	Encourage others to buy less	.41	.44	.12
CCAI14	Give time/money to orgs reducing purchase impact	-.01	.97	-.02
CCAI13	Give time/money to orgs reducing purchases	.02	.93	.01
CCAI5	Donate to charity rather than buying a gift	-.02	.78	.20
CCAI2	Use borrowed/rented/digital rather than buying	-.02	-.18	.90
CCAI4	Buy used rather than new	.00	.15	.76
CCAI1	Repair rather than buying replacements	.03	.06	.76
CCAI3	Use borrowed/rented tools rather than buying	.10	.21	.62
CCAI9	Donate or sell old possessions	.22	.27	.46

`CCAI_Phi` is a 3×3 factor intercorrelation matrix. All three intercorrelations exceed 0.50.

	F1	F2	F3
Choosing Sustainable Options	1.00	0.59	0.59
Supporting Collective Action	0.59	1.00	0.53
Avoiding Buying New	0.59	0.53	1.00

See `vignette("GPA3bifactor", package = "GPArotation")` for a bifactor analysis of these data.

Note

The raw data are publicly available on the Open Science Framework. As the data are subject to a license, users should consult the OSF page for terms of use before using the raw data directly: https://osf.io/h38yb/overview.

References

Barchard, K.A., Okagawa, K., Hoffman, C.K., and Odents, O. (2021). *Climate Change Action Inventory.* Unpublished psychological test. Available from kim.barchard@unlv.edu.

Bi, Y. and Barchard, K.A. (2024). Purchasing choices that reduce climate change: An exploratory factor analysis. *Spectra Undergraduate Research Journal*, **3**(2), 8–14. doi: 10.9741/2766-7227.1028.

See Also

`rotations`, `bifactorT`, `eigen`, `factanal`, `Harman`, `Thurstone`, `WansbeekMeijer`

Examples

```
data(CCAI, package = "GPArotation")

# Observed correlation matrix
round(CCAI_R, 2)
```

```
# Published pattern matrix and factor intercorrelations
round(CCAI_pattern, 2)
round(CCAI_Phi, 2)

# Reproduce published analysis: PCA extraction via eigendecomposition
# followed by direct oblimin rotation --- no additional packages
# required. This gives the same result as psych:::principal used
# by Bi and Barchard (2024).
ev          <- eigen(CCAI_R)
k           <- 3
L_unrotated <- ev$vectors[, 1:k] %*% diag(sqrt(ev$values[1:k]))
rownames(L_unrotated) <- colnames(CCAI_R)
res_oblimin <- oblimin(L_unrotated, randomStarts = 100)

# print applies sorting --- capture the sorted result
res_sorted  <- print(res_oblimin)
L_repro <- loadings(res_sorted)

# Compare reproduced vs published side by side, alternating by factor
comparison <- cbind(round(L_repro[, 1], 2),      round(CCAI_pattern[, 1], 2),
                    round(L_repro[, 2], 2),      round(CCAI_pattern[, 2], 2),
                    round(L_repro[, 3], 2),      round(CCAI_pattern[, 3], 2))
colnames(comparison) <- c("F1.repro", "F1.pub",
                          "F2.repro", "F2.pub",
                          "F3.repro", "F3.pub")
print(comparison)

# Orthogonal bifactor rotation on observed correlation matrix using MLE extraction
fa_unrotated <- factanal(factors = 3, covmat = CCAI_R, n.obs = 461,
                         rotation = "none")
bif <- bifactorT(loadings(fa_unrotated))
print(bif, sortLoadings = FALSE, digits = 3)
```

`echelon` *Echelon Rotation*

Description

Rotates a factor loading matrix to an echelon parameterization.

Usage

```
echelon(L, reference = seq(NCOL(L)), ...)
```

Arguments

`L`	a factor loading matrix.

reference	integer vector indicating which rows of the loading matrix are used to determine the rotation transformation. Default uses the first k rows. If the submatrix indicated by reference is singular, a different choice of rows must be supplied.
...	additional arguments discarded.

Details

The loading matrix is rotated so that the k rows indicated by reference form the Cholesky factorization given by t(chol(L[reference,] %*% t(L[reference,]))). This defines the rotation transformation, which is then applied to all rows to give the rotated loading matrix.

The optimization is not iterative and does not use the gradient projection algorithm. The function can be used directly or passed to factor analysis functions like factanal via the rotation argument.

This parameterization has several useful properties:

1. It can be useful for comparison with published results in this parameterization.
2. Standard errors are more straightforward to compute because the solution corresponds to an unconstrained optimization.
3. Models with k and $k + 1$ factors are nested, making it straightforward to test the k-factor model versus the $(k + 1)$-factor model. In particular, the Wald test and LM test can be used in addition to the LR test. The test of a k-factor model versus a $(k + 1)$-factor model is a joint test of whether all free parameters (loadings) in the $(k + 1)$st column are zero.
4. For some purposes, only the subspace spanned by the factors matters, not the specific parameterization within this subspace.
5. Back-predicted indicators (the explained portion of the indicators) do not depend on the rotation method. Combined with the greater ease of obtaining correct standard errors, this allows easier and more accurate prediction standard errors.
6. This parameterization and its standard errors can be used to detect identification problems (McDonald, 1999, pp. 181–182).

One use of echelon rotation is obtaining good starting values for subsequent rotation, though it may seem counterintuitive to rotate towards this solution afterwards.

Value

A GPArotation object which is a list with elements:

loadings	The rotated loadings matrix.
Th	The rotation matrix.
method	A string indicating the rotation method ("echelon").
orthogonal	Always TRUE (an orthogonal solution is assumed; Φ is the identity matrix).
convergence	Always TRUE (the optimization is not iterative).

Author(s)

Erik Meijer and Paul Gilbert.

References

McDonald, R.P. (1999). *Test Theory: A Unified Treatment*. Erlbaum.

Wansbeek, T. and Meijer, E. (2000). *Measurement Error and Latent Variables in Econometrics*. North-Holland.

See Also

eiv, rotations, GPForth, GPFoblq

Examples

```
data("WansbeekMeijer", package = "GPArotation")
fa.unrotated <- factanal(factors = 2, covmat = NetherlandsTV,
                         rotation = "none")

# Direct call
fa.ech <- echelon(fa.unrotated$loadings)

# Equivalent via factanal rotation argument
fa.ech2 <- factanal(factors = 2, covmat = NetherlandsTV,
                    rotation = "echelon")

# Compare unrotated, echelon, and factanal echelon loadings
cbind(loadings(fa.unrotated), loadings(fa.ech), loadings(fa.ech2))

# Echelon rotation with a different reference set
fa.ech3 <- echelon(fa.unrotated$loadings, reference = 6:7)
cbind(loadings(fa.unrotated), loadings(fa.ech), loadings(fa.ech3))
```

---

eiv *Errors-in-Variables Rotation*

Description

Rotates a factor loading matrix to an errors-in-variables representation.

Usage

```
eiv(L, identity = seq(NCOL(L)), ...)
```

Arguments

L	a factor loading matrix.
identity	integer vector indicating which rows of the loading matrix should form an identity matrix. Default uses the first k rows. If inverting the submatrix indicated by identity fails, a different choice of rows must be supplied.
...	additional arguments discarded.

Details

The loading matrix is rotated so that the k rows indicated by `identity` form an identity matrix, with the remaining $M - k$ rows as free parameters. Φ is also free.

The optimization is not iterative and does not use the gradient projection algorithm. The function can be used directly or passed to factor analysis functions like `factanal` via the `rotation` argument.

Viewed as a rotation method it is oblique, with an explicit solution. Given an initial loadings matrix L partitioned as $L = (L_1^T, L_2^T)^T$, the rotated loadings matrix is $(I, (L_2 L_1^{-1})^T)^T$ and $\Phi = L_1 L_1^T$, where I is the $k \times k$ identity matrix. It is assumed that $\Phi = I$ for the initial loadings matrix.

Not all authors consider this representation to be a rotation in the strict sense.

This parameterization has several useful properties:

1. It can be useful for comparison with published results in this parameterization.
2. Standard errors are more straightforward to compute because the solution corresponds to an unconstrained optimization.
3. One may have prior knowledge about which reference variables load on only one factor without imposing restrictive constraints on other loadings — in this sense it has similarities to CFA.
4. For some purposes, only the subspace spanned by the factors matters, not the specific parameterization within this subspace.
5. Back-predicted indicators (the explained portion of the indicators) do not depend on the rotation method. Combined with the greater ease of obtaining correct standard errors, this allows easier and more accurate prediction standard errors.

One use of this parameterization is obtaining good starting values for subsequent rotation, though it may seem counterintuitive to rotate towards this solution afterwards.

Value

A `GPArotation` object which is a list with elements:

`loadings`	The rotated loadings matrix.
`Th`	The rotation matrix.
`method`	A string indicating the rotation method (`"eiv"`).
`orthogonal`	Always `FALSE` (oblique rotation).
`convergence`	Always `TRUE` (the optimization is not iterative).
`Phi`	The covariance matrix of the rotated factors.

Author(s)

Erik Meijer and Paul Gilbert.

References

Hägglund, G. (1982). Factor analysis by instrumental variables methods. *Psychometrika*, **47**, 209–222.

Lewin-Koh, S.C. and Amemiya, Y. (2003). Heteroscedastic factor analysis. *Biometrika*, **90**, 85–97.

Wansbeek, T. and Meijer, E. (2000). *Measurement Error and Latent Variables in Econometrics*. North-Holland.

See Also

echelon, rotations, GPForth, GPFoblq

Examples

```
data("WansbeekMeijer", package = "GPArotation")
fa.unrotated <- factanal(factors = 2, covmat = NetherlandsTV,
                         rotation = "none")

# Direct call
fa.eiv <- eiv(fa.unrotated$loadings)

# Equivalent via factanal rotation argument
fa.eiv2 <- factanal(factors = 2, covmat = NetherlandsTV,
                    rotation = "eiv")

# Compare unrotated, eiv, and factanal eiv loadings
cbind(loadings(fa.unrotated), loadings(fa.eiv), loadings(fa.eiv2))

# Eiv rotation with a different identity set
fa.eiv3 <- eiv(fa.unrotated$loadings, identity = 6:7)
cbind(loadings(fa.unrotated), loadings(fa.eiv), loadings(fa.eiv3))
```

GPA *Core Algorithms and Random-Start Wrappers*

Description

Gradient projection rotation optimization routines for orthogonal and oblique factor rotation. These functions can be used directly to rotate a loadings matrix, or indirectly through a rotation objective passed to a factor estimation routine such as factanal.

Usage

```
GPForth(A, Tmat=diag(ncol(A)), normalize=FALSE, eps=1e-5, maxit=1000,
   method="varimax", methodArgs=NULL)
GPFoblq(A, Tmat=diag(ncol(A)), normalize=FALSE, eps=1e-5, maxit=1000,
   method="quartimin", methodArgs=NULL)

GPFRSorth(A, Tmat=diag(ncol(A)), normalize=FALSE, eps=1e-5, maxit=1000,
```

```
   method="varimax", methodArgs=NULL, randomStarts=0, ...)
GPFRSoblq(A, Tmat=diag(ncol(A)), normalize=FALSE, eps=1e-5, maxit=1000,
   method="quartimin", methodArgs=NULL, randomStarts=0, ...)
```

Arguments

A	initial factor loadings matrix for which the rotation criterion is to be optimized.
Tmat	initial rotation matrix.
normalize	see details.
eps	convergence is assumed when the norm of the gradient is smaller than eps.
maxit	maximum number of iterations allowed in the main loop.
method	rotation objective criterion.
methodArgs	a list of additional arguments passed to the rotation objective.
randomStarts	number of random starts (GPFRSorth and GPFRSoblq only).
...	additional arguments passed to GPForth or GPFoblq, such as eps and maxit when calling via GPFRSorth or GPFRSoblq.

Details

The GPFRSorth and GPFRSoblq functions serve as the primary user interfaces for orthogonal and oblique rotations, respectively. They act as wrappers for the core GP algorithms (GPForth for orthogonal rotation and GPFoblq for oblique rotation), extending them with the ability to perform multiple random starts. Any additional arguments provided to these wrappers are passed directly down to the underlying GP algorithms. While the wrappers are generally recommended, the core functions GPForth and GPFoblq can also be invoked directly.

All of these functions require an initial loadings matrix, A, which fixes the equivalence class over which the optimization is performed. This matrix must be the solution to an orthogonal factor analysis problem, such as one obtained from factanal or another factor estimation routine.

Mathematically, a general rotation of a matrix A is defined as A %*% solve(t(Th)). In the case of orthogonal rotation, the initial rotation matrix Tmat is orthonormal, which simplifies the rotation formula to A %*% Th. In all scenarios, the final rotation matrix Th is computed by the GP rotation algorithm.

An accessible introduction to gradient projection algorithms for factor rotation is provided in Mansolf and Reise (2016).

The normalize **argument:** The normalize argument specifies whether and how the loadings matrix should be normalized prior to rotation, and subsequently denormalized after rotation.

- If FALSE (the default), no normalization is performed.
- If TRUE, Kaiser normalization is applied so that the squared row entries of the normalized matrix A sum to 1.0. This procedure is sometimes referred to as Horst normalization.
- If provided as a **vector** (which must have a length equal to the number of indicators, i.e., the number of rows in A), the columns of A are divided by this vector before rotation and multiplied by it afterward.

- If provided as a **function**, it can be used to apply a custom normalization scheme. The function must take `A` as an argument and return a vector, which is then applied in the same manner as the vector input described above. See `NormalizingWeight` for an example implementing Cureton-Mulaik normalization.

For a detailed investigation into how normalization affects factor rotations, including its potential impact on the qualitative interpretation of loadings, see Nguyen and Waller (2022).

The `method` argument: The `method` argument takes a string specifying the rotation objective function. By default, oblique rotations use `"quartimin"`, while orthogonal rotations default to `"varimax"`. The package supports a comprehensive suite of rotation objectives: `"oblimin"`, `"quartimin"`, `"target"`, `"pst"`, `"oblimax"`, `"entropy"`, `"quartimax"`, `"Varimax"`, `"simplimax"`, `"bentler"`, `"tandemI"`, `"tandemII"`, `"geomin"`, `"cf"`, `"infomax"`, `"mccammon"`, `"bifactor"`, `"lp"`, and `"varimin"`.

Internally, this string is prefixed with `"vgQ."` to invoke the actual calculation function (see `vgQ` for underlying mathematical details). It is important to note that several rotation criteria—specifically `"oblimin"`, `"target"`, `"pst"`, `"simplimax"`, `"geomin"`, `"cf"`, and `"lp"`—require one or more supplementary arguments. These additional arguments can be seamlessly passed via the `methodArgs` list in the wrapper functions. Default values and direct usage examples for these arguments can be found in the `rotations` documentation.

The `randomStarts` argument: Because factor rotation criteria frequently suffer from local minima, trying multiple starting configurations can help identify a superior solution. The `randomStarts` argument, available exclusively in the `GPFRSorth` and `GPFRSoblq` wrappers, facilitates this robust search approach.

- By default, `randomStarts = 0`, which defaults to using the identity matrix as the initial rotation matrix `Tmat`. The initial rotation matrix `Tmat` can also be set by the user.
- Setting `randomStarts = 1` initializes `Tmat` with a single random matrix.
- Setting `randomStarts > 1` attempts multiple random starts and returns the rotated loadings matrix that achieved the lowest criterion value `f` across all attempts. Note that this returned solution is technically still a local minimum, and is not guaranteed to be the global minimum. Users are encouraged to review the random start diagnostics detailed in the package examples.

Under the hood, an internal, unexported engine named `.GPA_RS_engine` safely manages the random start loop, tracks convergence diagnostics, and handles factor correlation matrix naming.

While the core algorithms `GPForth` and `GPFoblq` do not support the `randomStarts` argument directly, users can manually supply a single random initial rotation matrix to them using `Tmat = Random.Start(ncol(A))`.

Legacy functions: The original implementations authored by Bernaards and Jennrich (2005) have been retained as `GPForth.legacy` and `GPFoblq.legacy`. These functions are kept purely for historical reference and backward compatibility for reproducibility. They are not exported into the package namespace, meaning they must be explicitly invoked using the triple-colon operator:

```
GPArotation:::GPForth.legacy(A, method = "varimax")
GPArotation:::GPFoblq.legacy(A, method = "quartimin")
```

The results generated by these legacy functions should be numerically identical to those produced by the current implementations. You can see a direct comparison of this in the examples section.

Value

A `GPArotation` object which is a list with elements:

`loadings`	The rotated loadings matrix, one column per factor. If random starts were requested, this is the solution with the lowest criterion value.
`Th`	The rotation matrix, satisfying `loadings %*% t(Th) = A` for orthogonal rotation and `loadings = A %*% solve(t(Th))` for oblique rotation.
`Table`	A matrix recording the iteration history: iteration number, criterion value, log10 of the gradient norm, and step size (alpha).
`method`	A string indicating the rotation criterion.
`orthogonal`	A logical indicating if the rotation is orthogonal.
`convergence`	A logical indicating if convergence was obtained.
`Phi`	`t(Th) %*% Th`, the covariance matrix of the rotated factors. Omitted (`NULL`) for orthogonal rotations.
`Gq`	The gradient of the criterion at the rotated loadings.
`randStartChar`	A named vector summarising random start results: `randomStarts`, `Converged`, `atMinimum`, `localMins`. Only present when `randomStarts > 1`.

Author(s)

Coen A. Bernaards and Robert I. Jennrich with some R modifications by Paul Gilbert

References

Bernaards, C.A. and Jennrich, R.I. (2005) Gradient Projection Algorithms and Software for Arbitrary Rotation Criteria in Factor Analysis. *Educational and Psychological Measurement*, **65**, 676–696. doi: 10.1177/0013164404272507

Jennrich, R.I. (2001). A simple general procedure for orthogonal rotation. *Psychometrika*, **66**, 289–306. doi: 10.1007/BF02294840

Jennrich, R.I. (2002). A simple general method for oblique rotation. *Psychometrika*, **67**, 7–19. doi: 10.1007/BF02294706

Mansolf, M., & Reise, S. P. (2016). Exploratory Bifactor Analysis: The Schmid-Leiman Orthogonalization and Jennrich-Bentler Analytic Rotations. *Multivariate Behavioral Research*, **51**(5), 698–717. doi: 10.1080/00273171.2016.1215898

Nguyen, H.V. and Waller, N.G. (2023). Local minima and factor rotations in exploratory factor analysis. *Psychological Methods*. **28**(5), 1122–1141. doi: 10.1037/met0000467

See Also

`rotations`, `Random.Start`, `factanal`, `Harman8`, `box26`, `CCAI`, `NetherlandsTV`

Examples

```
# --- Basic rotation calls ---
data(Harman, package = "GPArotation")           # 8 physical variables
quartimax(Harman8)                              # direct rotation call
GPFRSorth(Harman8, method = "quartimax")        # equivalent via wrapper
GPFRSoblq(Harman8, method = "quartimin", normalize = TRUE)
loadings(quartimin(Harman8, normalize = TRUE))  # extract loadings directly

# --- Passing criterion arguments via methodArgs ---
# Crawford-Ferguson family: kappa selects the criterion.
# For box26: p = 26 variables, m = 3 factors.
# Equamax:  kappa = m / (2 * p) = 3 / 52
# Parsimax: kappa = (m - 1) / (p + m - 2) = 2 / 27
data(Thurstone, package = "GPArotation") # 26 variable box problem
GPFRSoblq(box26, method = "cf", methodArgs = list(kappa = 3/52))  # Equamax
GPFRSoblq(box26, method = "cf", methodArgs = list(kappa = 2/27))  # Parsimax

# --- Two-step vs single-step factanal for oblique rotation ---
#
# The recommended approach for oblique rotation is the two-step procedure:
# (1) obtain unrotated loadings from factanal, then
# (2) rotate separately using GPArotation.
# This gives full control over the rotation, including random starts.
#
# Prior to R 4.5.1, the single-step approach (rotation inside factanal)
# had a bug in factor reordering after oblique rotation. This was reported
# by Bernaards and others and fixed by the R core team in R 4.5.1.

data("WansbeekMeijer", package = "GPArotation")

# Step 1: unrotated 3-factor solution
fa.unrotated <- factanal(factors = 3, covmat = NetherlandsTV,
                         normalize = TRUE, rotation = "none")

# Step 2: oblique Crawford-Ferguson rotation with kappa = 0.3
# (non-standard kappa, not corresponding to any named special case)
set.seed(44)
fa.cf <- cfQ(loadings(fa.unrotated), kappa = 0.3, normalize = TRUE,
             randomStarts = 100)
fa.cf

# Single-step via factanal - correct in R >= 4.5.1
if (getRversion() >= "4.5.1") {
  set.seed(44)
  fa.factanal <- factanal(factors = 3, covmat = NetherlandsTV, rotation = "cfQ",
    control = list(rotate = list(normalize = TRUE, kappa = 0.3, randomStarts = 100)))
  # The two approaches should agree after sorting
  fa.sorted <- print(fa.cf, sortLoadings = TRUE)
  cat("Maximum difference in loadings between two-step and single-step:\n")
  print(max(abs(abs(fa.sorted$loadings) - abs(fa.factanal$loadings))))
} else {
  cat("Single-step factanal oblique rotation requires R >= 4.5.1.\n")
```

```
  cat("Use the two-step procedure above for correct results.\n")
}

# --- Displaying rotation output ---
origdigits <- options("digits")
data("CCAI", package = "GPArotation")
fa.unrotated <- factanal(factors = 3, covmat = CCAI_R, n.obs = 461, rotation = "none")
res <- oblimin(loadings(fa.unrotated), gam = -0.5, randomStarts = 20)
                                 # gam = -0.5: more orthogonal than quartimin
res                              # default print
print(res)                       # equivalent to above
print(res, Table = TRUE)         # include iteration table
print(res, rotateMat = TRUE)     # include rotating matrix
print(res, digits = 2)           # rounded to 2 decimal places
summary(res)                    # pattern and structure matrices for oblique rotation
summary(res, Structure = FALSE)   # pattern matrix only
options(digits = origdigits$digits)

# --- Random start diagnostics ---
# When randomStarts > 1, the output includes randStartChar which summarizes
# the random start results:
#   randomStarts : number of random starts attempted
#   Converged    : number of starts that converged
#   atMinimum    : number of starts at the same lowest minimum
#   localMins    : number of distinct local minima found
data(Thurstone, package = "GPArotation")
res <- GPFRSoblq(box26, method = "geomin", normalize = TRUE, randomStarts = 50)
res$randStartChar

# --- Factor ordering ---
# Raw GPArotation output is unsorted — factors may appear in any order
# depending on the starting matrix. Use print() to obtain sorted loadings.
# Once sorted, repeated calls to print() are stable.
set.seed(334)
xusl <- quartimin(Harman8, normalize = TRUE, randomStarts = 100)

loadings(xusl)                                    # unsorted raw output
max(abs(print(xusl)$loadings - xusl$loadings)) == 0  # FALSE: print() reorders
xsl <- print(xusl)                                 # capture sorted result
max(abs(print(xsl)$loadings - xsl$loadings)) == 0    # TRUE: already sorted

# --- Normalization ---
# Kaiser normalization
data("CCAI", package = "GPArotation")
fa.unrotated <- factanal(factors = 3, covmat = CCAI_R, n.obs = 461, rotation = "none")
oblimin(loadings(fa.unrotated), normalize = TRUE, randomStarts = 100)

# --- Legacy engine ---
# Demonstrates numerical equivalence of current and legacy implementations.
# The maximum absolute difference should be zero or at machine epsilon.
data("Harman", package = "GPArotation")

# Rotate using current engine
```

```
res_new    <- oblimin(Harman8)

# Rotate using legacy engine
res_legacy <- GPArotation:::GPFoblq.legacy(Harman8, method = "oblimin")

# Numerically identical
max(abs(loadings(res_new) - loadings(res_legacy)))
```

Harman *Harman's Eight Physical Variables*

Description

Unrotated factor loading matrix extracted by the centroid method for Harman's 8 physical variables, a classic dataset in factor analysis.

Usage

```
data(Harman)
```

Format

`Harman8`: a 8×2 numeric matrix of unrotated centroid factor loadings. Row names are variable names; column names are `CF1` and `CF2`.

Details

`Harman8` is a 8×2 matrix of unrotated factor loadings extracted by the centroid method (Table 14.3 in Harman, 1976) from the correlation matrix of 8 physical measurements taken on 305 girls (Table 2.3 in Harman, 1976). The centroid method was a standard extraction procedure before iterative methods became computationally feasible. The two columns represent unrotated centroid factors (`CF1` and `CF2`). Row names are the 8 variable names.

The covariance matrix for these 8 physical variables is available in base R (see `Harman23.cor`).

Source

Harman, H.H. (1976). *Modern Factor Analysis* (3rd ed.). University of Chicago Press.

See Also

`Harman23.cor`, `GPForth`, `rotations`, `Thurstone`, `WansbeekMeijer`, `CCAI`

Examples

```
data(Harman, package = "GPArotation")

# Centroid unrotated loadings from Harman (1976) Table 14.3
print(Harman8)

# Rotate the centroid solution
quartimax(Harman8)
oblimin(Harman8, randomStarts = 100)
```

---

lp *L^p Rotation*

Description

Performs L^p rotation to obtain sparse factor loadings.

Usage

```
GPForth.lp(A, Tmat = diag(ncol(A)), p = 1, normalize = FALSE, eps = 1e-05,
           maxit = 10000, gpaiter = 5)
GPFoblq.lp(A, Tmat = diag(ncol(A)), p = 1, normalize = FALSE, eps = 1e-05,
           maxit = 10000, gpaiter = 5)
```

Arguments

A	Initial factor loadings matrix to be rotated.
Tmat	Initial rotation matrix.
p	Component-wise L^p where $0 < p \leq 1$.
normalize	Not recommended for L^p rotation.
eps	Convergence is assumed when the norm of the gradient is smaller than eps.
maxit	Maximum number of iterations allowed in the main loop.
gpaiter	Maximum iterations for the inner GPA rotation loop. The goal is to decrease the objective value, not fully optimize the inner loop. Warnings may appear but can be ignored if the main loop converges.

Details

These functions optimize an L^p rotation objective where `0 < p <= 1`. A smaller value of p promotes greater sparsity in the loading matrix but increases computational difficulty. For guidance on choosing p, see the Concluding Remarks in Liu et al. (2023).

The user-facing wrapper functions lpT and lpQ provide random start functionality on top of `GPForth.lp` and `GPFoblq.lp` respectively, analogous to how GPFRSorth and GPFRSoblq wrap `GPForth` and `GPFoblq` for standard rotation criteria. For most users `lpT` and `lpQ` are the recommended entry points.

Since the L^p objective is nonsmooth, a different optimization method is required compared to smooth rotation criteria. `GPForth.lp` and `GPFoblq.lp` replace `GPForth` and `GPFoblq` for orthogonal and oblique L^p rotations, respectively.

The optimization uses an iterative reweighted least squares (IRLS) approach. The nonsmooth objective is approximated by a smooth weighted least squares function in the outer loop, which is then optimized using GPA in the inner loop (`gpaiter` controls the maximum inner iterations).

Normalization is not recommended for L^p rotation and may produce unexpected results.

Value

A `GPArotation` object, which is a list containing:

`loadings`	Rotated loadings matrix, with one column per factor.
`Th`	Rotation matrix, satisfying `loadings %*% t(Th) = A`.
`Table`	Data frame recording iteration details: iteration count, objective value, and elapsed time.
`method`	String indicating the rotation objective function.
`orthogonal`	Logical indicating whether the rotation is orthogonal.
`convergence`	Logical indicating whether convergence was achieved. Convergence is assessed element-wise using `eps` as tolerance.
`Phi`	Covariance matrix of rotated factors, `t(Th) %*% Th`.

Author(s)

Xinyi Liu, with minor modifications for GPArotation by C. Bernaards.

References

Liu, X., Wallin, G., Chen, Y., and Moustaki, I. (2023). Rotation to sparse loadings using L^p losses and related inference problems. *Psychometrika*, **88**(2), 527–553. doi: 10.1007/s11336-023-09911-y

See Also

`lpT`, `lpQ`, `vgQ.lp.wls`

Examples

```
data("WansbeekMeijer", package = "GPArotation")
fa.unrotated <- factanal(factors = 2, covmat = NetherlandsTV, rotation = "none")
options(warn = -1)

# Orthogonal rotation - single start
fa.lpT1 <- GPForth.lp(loadings(fa.unrotated), p = 1)

# Orthogonal rotation - 10 random starts
fa.lpT <- lpT(loadings(fa.unrotated), Tmat = Random.Start(2), p = 1,
              randomStarts = 10)
```

```
print(fa.lpT, digits = 5, sortLoadings = FALSE, Table = TRUE, rotateMat = TRUE)

# Oblique rotation - single start
fa.lpQ1 <- GPFoblq.lp(loadings(fa.unrotated), p = 1)

# Oblique rotation - 10 random starts
fa.lpQ <- lpQ(loadings(fa.unrotated), p = 1, randomStarts = 10)
summary(fa.lpQ, Structure = TRUE)

# Compare Lp (p=1), Lp (p=0.5), and Geomin oblique rotations
set.seed(1020)
fa.lpQ1   <- lpQ(loadings(fa.unrotated), p = 1,   randomStarts = 10)
fa.lpQ0.5 <- lpQ(loadings(fa.unrotated), p = 0.5, randomStarts = 10)
fa.geo    <- geominQ(loadings(fa.unrotated),       randomStarts = 10)

# With factor ordering using internal sortGPALoadings helper
res <- round(cbind(GPArotation:::.sortGPALoadings(fa.lpQ1)$loadings,
                   GPArotation:::.sortGPALoadings(fa.lpQ0.5)$loadings,
                   GPArotation:::.sortGPALoadings(fa.geo)$loadings), 3)
print(c("oblique --  Lp p=1        Lp p=0.5       Geomin"))
print(res)

# Without factor ordering
res <- round(cbind(fa.lpQ1$loadings, fa.lpQ0.5$loadings, fa.geo$loadings), 3)
print(c("oblique --  Lp p=1        Lp p=0.5       Geomin"))
print(res)

options(warn = 0)
```

`print.GPArotation` *Print and Summary Methods for GPArotation Class Objects*

Description

Print and summary methods for objects returned by `GPFRSorth`, `GPFRSoblq`, `GPForth`, or `GPFoblq`.

Both `print.GPArotation` and `summary.GPArotation` apply consistent factor sorting and sign correction via the internal helper `.sortGPALoadings` when `sortLoadings = TRUE`. Factors are ordered by descending variance explained and signs are adjusted so that the sum of loadings per factor is positive. This convention matches that used by `factanal`.

For oblique rotations, `summary.GPArotation` displays both the pattern matrix (regression coefficients of items on factors, controlling for factor intercorrelations) and the structure matrix (`loadings %*% Phi`, correlations between items and factors) when `Structure = TRUE`. The two matrices coincide for orthogonal rotations where $\Phi = I$.

Output includes contributions of factors via `SS loadings` (sum of squared loadings); see Harman (1976), sections 2.4 and 12.4.

Usage

```
    ## S3 method for class 'GPArotation'
print(x, digits=3, sortLoadings=TRUE, rotateMat=FALSE, Table=FALSE, ...)
    ## S3 method for class 'GPArotation'
summary(object, digits=3, Structure=TRUE, ...)
    ## S3 method for class 'summary.GPArotation'
print(x, ...)
```

Arguments

x	a GPArotation or summary.GPArotation object to print.
object	a GPArotation object to summarize.
digits	precision of printed numbers.
sortLoadings	logical; if TRUE (default) factors are sorted by descending variance explained and factor signs are adjusted so that the sum of loadings per factor is positive. Adapted from factanal sorting conventions. Use sortLoadings = FALSE to display the raw unsorted solution, for example when the factor order is meaningful as in bifactor rotation.
rotateMat	logical; if TRUE the rotation matrix is displayed.
Table	logical; if TRUE the iteration table is displayed.
Structure	logical; if TRUE (default) the structure matrix (loadings %*% Phi) is displayed for oblique rotations in summary.
...	further arguments passed to other methods.

Details

Factor sorting and sign correction are applied consistently in both print and summary via the internal function .sortGPALoadings, adapted from factanal sorting conventions (R Core Team). This ensures that the pattern matrix shown by summary is consistent with the loadings shown by print.

The digits argument controls the number of decimal places shown in the loadings, structure matrix, and Phi.

For examples see GPFRSorth and the package vignettes: vignette("GPA1guide", package = "GPArotation").

Value

print.GPArotation returns the sorted GPArotation object invisibly when sortLoadings = TRUE, or the unsorted object when sortLoadings = FALSE. summary.GPArotation returns a summary.GPArotation object with sorted loadings and, for oblique rotations, the structure matrix. print.summary.GPArotation returns the object invisibly.

References

Harman, H.H. (1976). *Modern Factor Analysis*. The University of Chicago Press.

See Also

GPFRSorth, GPForth, factanal, summary

Examples

```
data(Harman, package = "GPArotation")
res <- oblimin(Harman8, normalize = TRUE, randomStarts = 100)

# Print sorted loadings (default)
print(res)

# Print unsorted loadings
print(res, sortLoadings = FALSE)

# Summary with pattern and structure matrices
summary(res, Structure = TRUE)

# Summary without structure matrix
summary(res, Structure = FALSE)

# Print with iteration table
print(res, Table = TRUE)
```

Random.Start *Random Starting Matrix for Factor Rotation*

Description

Generates a random orthogonal matrix for use as an initial rotation matrix (`Tmat`) in gradient projection rotation functions.

Usage

```
Random.Start(k = 2L)
```

Arguments

`k`	a positive integer specifying the dimension of the square orthogonal matrix to generate (default 2).

Details

The function generates a random orthogonal matrix using QR decomposition of a matrix of standard normal variates, with a sign correction applied to the diagonal of R to ensure uniform sampling from the Haar measure. This follows the approach of Stewart (1980) and Mezzadri (2007).

The naive approach of `qr.Q(qr(matrix(rnorm(k*k), k)))` does not guarantee uniform sampling from the Haar measure. The sign correction `Q %*% diag(sign(diag(R)))` is required to achieve this. This was updated in GPArotation 2024.2-1 following a suggestion by Yves Rosseel.

For oblique rotation a random starting transformation matrix can be generated by normalizing the columns of a random matrix: `X %*% diag(1/sqrt(diag(crossprod(X))))` where `X <- matrix(rnorm(k*k), k)`.

Value

A $k \times k$ orthogonal matrix drawn uniformly from the Haar measure on the orthogonal group O(k). Columns have unit length and are mutually orthogonal.

Author(s)

Coen A. Bernaards and Robert I. Jennrich with some R modifications by Paul Gilbert. Updated following a suggestion by Yves Rosseel.

References

Stewart, G.W. (1980). The efficient generation of random orthogonal matrices with an application to condition estimators. *SIAM Journal on Numerical Analysis*, **17**(3), 403–409. doi: 10.1137/0717034

Mezzadri, F. (2007). How to generate random matrices from the classical compact groups. *Notices of the American Mathematical Society*, **54**(5), 592–604. arXiv:math-ph/0609050

See Also

GPFRSorth, GPFRSoblq, GPForth, GPFoblq, rotations

Examples

```
# Generate a 5 x 5 random orthogonal matrix
Random.Start(5)

# Verify orthogonality: t(Q) %*% Q should be the identity matrix
Q <- Random.Start(4)
round(t(Q) %*% Q, 10)

# Use as starting matrix for rotation
data("Thurstone", package = "GPArotation")
simplimax(box26, Tmat = Random.Start(3))
```

---

rotations *Rotations Functions Using Gradient Projection Algorithms*

Description

Optimize factor loading rotation objective.

Usage

```
oblimin(A, Tmat=diag(ncol(A)), gam=0, normalize=FALSE, randomStarts=0, ...)
quartimin(A, Tmat=diag(ncol(A)), normalize=FALSE, randomStarts=0, ...)
targetT(A=NULL, Tmat=diag(ncol(A)), Target=NULL, normalize=FALSE, eps=1e-5,
    maxit=1000, randomStarts=0, L=NULL, ...)
targetQ(A=NULL, Tmat=diag(ncol(A)), Target=NULL, normalize=FALSE, eps=1e-5,
    maxit=1000, randomStarts=0, L=NULL, ...)
```

```
pstT(A=NULL, Tmat=diag(ncol(A)), W=NULL, Target=NULL, normalize=FALSE, eps=1e-5,
     maxit=1000, randomStarts=0, L=NULL, ...)
pstQ(A=NULL, Tmat=diag(ncol(A)), W=NULL, Target=NULL, normalize=FALSE, eps=1e-5,
     maxit=1000, randomStarts=0, L=NULL, ...)
 oblimax(A, Tmat=diag(ncol(A)), normalize=FALSE, randomStarts=0, ...)
 entropy(A, Tmat=diag(ncol(A)), normalize=FALSE, randomStarts=0, ...)
 quartimax(A, Tmat=diag(ncol(A)), normalize=FALSE, randomStarts=0, ...)
 Varimax(A, Tmat=diag(ncol(A)), normalize=FALSE, randomStarts=0, ...)
simplimax(A, Tmat=diag(ncol(A)), k=nrow(A), normalize=FALSE, randomStarts=0, ...)
 bentlerT(A, Tmat=diag(ncol(A)), normalize=FALSE, randomStarts=0, ...)
 bentlerQ(A, Tmat=diag(ncol(A)), normalize=FALSE, randomStarts=0, ...)
 tandemI(A, Tmat=diag(ncol(A)), normalize=FALSE, randomStarts=0, ...)
 tandemII(A, Tmat=diag(ncol(A)), normalize=FALSE, randomStarts=0, ...)
geominT(A, Tmat=diag(ncol(A)), delta=0.01, normalize=FALSE, randomStarts=0, ...)
geominQ(A, Tmat=diag(ncol(A)), delta=0.01, normalize=FALSE, randomStarts=0, ...)
bigeominT(A, Tmat=diag(ncol(A)), delta=0.01, normalize=FALSE, randomStarts=0, ...)
bigeominQ(A, Tmat=diag(ncol(A)), delta=0.01, normalize=FALSE, randomStarts=0, ...)
 cfT(A, Tmat=diag(ncol(A)), kappa=0, normalize=FALSE, randomStarts=0, ...)
 cfQ(A, Tmat=diag(ncol(A)), kappa=0, normalize=FALSE, randomStarts=0, ...)
 equamax(A, Tmat=diag(ncol(A)), kappa=ncol(A)/(2*nrow(A)), normalize=FALSE,
     randomStarts=0, ...)
 parsimax(A, Tmat=diag(ncol(A)), kappa=(ncol(A)-1)/(ncol(A)+nrow(A)-2),
     normalize=FALSE, randomStarts=0, ...)
 infomaxT(A, Tmat=diag(ncol(A)), normalize=FALSE, randomStarts=0, ...)
 infomaxQ(A, Tmat=diag(ncol(A)), normalize=FALSE, randomStarts=0, ...)
 mccammon(A, Tmat=diag(ncol(A)), normalize=FALSE, randomStarts=0, ...)
 varimin(A, Tmat=diag(ncol(A)), normalize=FALSE, randomStarts=0, ...)
 bifactorT(A, Tmat=diag(ncol(A)), normalize=FALSE, randomStarts=0, ...)
 bifactorQ(A, Tmat=diag(ncol(A)), normalize=FALSE, randomStarts=0, ...)
 lpT(A, Tmat=diag(ncol(A)), p=1, normalize=FALSE, eps=1e-05, maxit=1000,
         randomStarts=0, gpaiter=5)
 lpQ(A, Tmat=diag(ncol(A)), p=1, normalize=FALSE, eps=1e-05, maxit=1000,
     randomStarts=0, gpaiter=5)
```

Arguments

A	an initial loadings matrix to be rotated.
Tmat	initial rotation matrix.
gam	Obliqueness parameter (γ in the mathematical definition). 0=Quartimin, .5=Bi-quartimin, 1=Covarimin.
Target	rotation target for objective calculation.
W	weighting of each element in target.
k	number of close to zero loadings.
delta	constant added to Λ^2 in the objective calculation.
kappa	see details.

normalize	parameter passed to optimization routine (GPForth or GPFoblq).
eps	convergence tolerance passed to GPForth or GPFoblq via .... Convergence is assumed when the norm of the gradient is smaller than eps. Default is 1e-5.
maxit	maximum number of iterations passed to GPForth or GPFoblq via .... Default is 1000.
...	additional arguments passed to GPForth or GPFoblq, including eps and maxit.
randomStarts	parameter passed to optimization routine (GPFRSorth or GPFRSoblq).
L	provided for backward compatibility in target rotations only. Use A going forward.
p	Component-wise L^p, where 0 < p =< 1.
gpaiter	Maximum iterations for GPA rotation loop in L^p rotation.

Details

These functions optimize a rotation objective. They can be used directly or the function name can be passed to factor analysis functions like factanal. Several of the function names end in T or Q, which indicates if they are orthogonal or oblique rotations (using GPFRSorth or GPFRSoblq respectively). The gradient projection algorithms are described in Bernaards and Jennrich (2005).

oblimin	oblique	oblimin family; gam controls obliqueness
quartimin	oblique	oblimin with gam = 0
targetT	orthogonal	rotation towards a target matrix
targetQ	oblique	rotation towards a target matrix
pstT	orthogonal	partially specified target rotation
pstQ	oblique	partially specified target rotation
oblimax	oblique	maximizes overall kurtosis of loadings
entropy	orthogonal	minimizes entropy of squared loadings
quartimax	orthogonal	maximizes variance of squared loadings within variables
Varimax	orthogonal	maximizes variance of squared loadings within factors
simplimax	oblique	minimizes the k smallest squared loadings
bentlerT	orthogonal	invariant pattern simplicity
bentlerQ	oblique	invariant pattern simplicity
tandemI	orthogonal	factors share high loadings on same variables
tandemII	orthogonal	factors do not share high loadings on same variables
geominT	orthogonal	minimizes geometric mean of squared loadings
geominQ	oblique	minimizes geometric mean of squared loadings
bigeominT	orthogonal	geomin with a general factor in column 1
bigeominQ	oblique	geomin with a general factor in column 1
cfT	orthogonal	Crawford-Ferguson family; kappa controls complexity
cfQ	oblique	Crawford-Ferguson family; kappa controls complexity
equamax	orthogonal	Crawford-Ferguson with kappa = m/(2p)
parsimax	orthogonal	Crawford-Ferguson with kappa = (m-1)/(p+m-2)
infomaxT	orthogonal	infomax information criterion
infomaxQ	oblique	infomax information criterion
mccammon	orthogonal	minimizes entropy ratio across factors
varimin	orthogonal	minimizes variance of squared loadings within factors

bifactorT	orthogonal	bifactor; general factor in column 1
bifactorQ	oblique	biquartimin; general factor in column 1
lpT	orthogonal	L^p sparsity rotation
lpQ	oblique	L^p sparsity rotation

The Varimax implementation in the list uses the gradient projection algorithm applied to vgQ.varimax. This implementation is different that the varimax rotation defined in the stats package. Additionally, varimax does Kaiser normalization by default whereas GPArotation::Varimax does not.

The argument kappa parameterizes the family for the Crawford-Ferguson method. If m is the number of factors and p is the number of indicators then kappa values having special names are 0 =Quartimax, $1/p$ =Varimax, $m/(2*p)$ =Equamax, $(m-1)/(p+m-2)$ =Parsimax, 1 =Factor parsimony.

Bifactor rotations, bifactorT and bifactorQ are called bifactor and biquartimin in Jennrich and Bentler (2011). For a comparison of exploratory bifactor analysis algorithms including those implemented here, see Garcia-Garzon, Abad and Garrido (2021).

The argument p is needed for L^p rotation. See Lp rotation for details on the rotation method.

Value

A GPArotation object which is a list with elements:

loadings	The rotated loadings matrix, one column per factor. If random starts were requested, this is the solution with the lowest criterion value.
Th	The rotation matrix, satisfying loadings %*% t(Th) = A for orthogonal rotation and loadings = A %*% solve(t(Th)) for oblique rotation.
Table	A matrix recording the iteration history: iteration number, criterion value, log10 of the gradient norm, and step size (alpha).
method	A string indicating the rotation criterion.
orthogonal	A logical indicating if the rotation is orthogonal.
convergence	A logical indicating if convergence was obtained.
Phi	t(Th) %*% Th, the covariance matrix of the rotated factors. Omitted (NULL) for orthogonal rotations.
Gq	The gradient of the criterion at the rotated loadings.
randStartChar	A named vector summarising random start results: randomStarts, Converged, atMinimum, localMins. Only present when randomStarts > 1.

Author(s)

Coen A. Bernaards and Robert I. Jennrich with some R modifications by Paul Gilbert.

References

Bernaards, C.A. and Jennrich, R.I. (2005) Gradient Projection Algorithms and Software for Arbitrary Rotation Criteria in Factor Analysis. *Educational and Psychological Measurement*, **65**, 676–696. doi: 10.1177/0013164404272507

Bi, Y. and Barchard, K.A. (2024). Purchasing choices that reduce climate change: An exploratory factor analysis. *Spectra Undergraduate Research Journal*, **3**(2), 8–14. doi: 10.9741/2766-7227.1028.

Fischer, R., & Fontaine, J. (2010). Methods for investigating structural equivalence. In D. Matsumoto & F. van de Vijver (Eds.), Cross-Cultural Research Methods in Psychology (pp. 179–215). Cambridge University Press. doi: 10.1017/CBO9780511779381.010

Garcia-Garzon, E., Abad, F.J. and Garrido, L.E. (2021). On omega hierarchical estimation: A comparison of exploratory bi-factor analysis algorithms. *Multivariate Behavioral Research*, **56**(1), 101–119. doi: 10.1080/00273171.2020.1736977

Jennrich, R.I. and Bentler, P.M. (2011). Exploratory bi-factor analysis. *Psychometrika*, **76**(4), 537–549. doi: 10.1007/s11336-011-9218-4

For references to individual rotation criteria see `vignette("GPA1guide", package = "GPArotation")`.

See Also

`factanal`, `GPFRSorth`, `GPFRSoblq`, `vgQ`, `Harman8`, `NetherlandsTV`, `CCAI`, `box26`

Examples

```
# For extended examples see the vignettes:
# vignette("GPA1guide",    package = "GPArotation")
# vignette("GPA2local",    package = "GPArotation")
# vignette("GPA3bifactor", package = "GPArotation")

# --- Accessing rotated loadings ---
data("Harman", package = "GPArotation") # 8 physical variables
qHarman <- quartimax(Harman8)
loadings(qHarman)                             # via extractor (recommended)
qHarman$loadings                              # via direct list access
all.equal(loadings(qHarman), qHarman$loadings) # identical

# --- Rotating factanal loadings ---
data("WansbeekMeijer", package = "GPArotation") # Netherlands TV viewership
fa.unrotated <- factanal(factors = 2, covmat = NetherlandsTV,
                         normalize = TRUE, rotation = "none")
quartimax(loadings(fa.unrotated), normalize = TRUE)
geominQ(loadings(fa.unrotated), normalize = TRUE, randomStarts = 100)

# --- Passing rotation to factanal ---
# CCAI:Climate-Friendly Purchasing Choices domain of the Climate Change Action Inventory
data("CCAI", package = "GPArotation")
factanal(factors = 3, covmat = CCAI_R, n.obs = 461, rotation = "infomaxT")
factanal(factors = 3, covmat = CCAI_R, n.obs = 461, rotation = "infomaxT",
         control = list(rotate = list(normalize = TRUE, eps = 1e-6)))

# --- Target rotation ---
```

```
# Orthogonal target rotation of two varimax rotated matrices
# towards each other. Data from Fischer and Fontaine (2010).
# See vignette("GPA1guide", package = "GPArotation") for further analyses.
trBritain <- matrix(c(.783, -.163, .811, .202, .724, .209, .850, .064,
                      -.031, .592, -.028, .723, .388, .434, .141, .808,
                      .215, .709), byrow = TRUE, ncol = 2)

trGermany <- matrix(c(.778, -.066, .875, .081, .751, .079, .739, .092,
                      .195, .574, -.030, .807, -.135, .717, .125, .738,
                      .060, .691), byrow = TRUE, ncol = 2)
trx <- targetT(trGermany, Target = trBritain)
round(trx$loadings - trBritain, 3)  # difference from target

# --- Partially specified target rotation ---
# See vignette("GPA1guide", package = "GPArotation") for full context.
# Unrotated loadings matrix A and partially specified target SPA
# NA entries in SPA are unspecified --- rotation is free there
# Numeric entries are the target values the rotation aims towards
A <- matrix(c(.664, .688, .492, .837, .705, .82, .661, .457, .765, .322,
              .248, .304, -0.291, -0.314, -0.377, .397, .294, .428,
              -0.075, .192, .224, .037, .155, -.104, .077, -.488, .009), ncol = 3)
SPA <- matrix(c(rep(NA, 6), .7, .0, .7, rep(0, 3), rep(NA, 7),
                0, 0, NA, 0, rep(NA, 4)), ncol = 3)
comparison <- cbind(round(A, 3), rep(NA, nrow(A)), SPA)
colnames(comparison) <- c("A.F1", "A.F2", "A.F3", "|", "T.F1", "T.F2", "T.F3")
cat("Unrotated loadings (A) and partially specified target (SPA):\n")
print(comparison, na.print = "NA")
targetT(A, Target = SPA)

# --- Random starts ---
# CCAI Climate-Friendly Purchasing Choices domain, 14 items, 3 oblique factors.
# High factor intercorrelations make oblimin the natural choice.
# Note: factanal uses MLE extraction; results differ somewhat from
# PCA-based extraction used in Bi and Barchard (2024).
data("CCAI", package = "GPArotation")
fa.unrotated <- factanal(factors = 3, covmat = CCAI_R, n.obs = 461, rotation = "none")
oblimin(loadings(fa.unrotated), Tmat = Random.Start(3))  # single random start
oblimin(loadings(fa.unrotated), randomStarts = 1)        # equivalent
oblimin(loadings(fa.unrotated), randomStarts = 100)      # multiple starts

# Directly via factanal call
factanal(factors = 3, covmat = CCAI_R, n.obs = 461, rotation = "oblimin",
    control = list(rotate = list(normalize = TRUE, gam = -0.1, randomStarts = 100)))

# --- Assessing local minima ---
# For detailed investigation of local minima across all random starts
# see vignette("GPA2local", package = "GPArotation").
data(Thurstone, package = "GPArotation")
infomaxQ(box26, normalize = TRUE, randomStarts = 150)
geominQ(box26,  normalize = TRUE, randomStarts = 150)
```

Thurstone *Thurstone Box Data*

Description

Factor loading matrices derived from Thurstone's box data, a classic dataset in factor analysis consisting of measurements of physical dimensions of a set of boxes.

Usage

```
data(Thurstone)
```

Format

Two numeric matrices:

- `box26`: 26 x 3 matrix of unrotated factor loadings (recommended).
- `box20`: 20 x 3 matrix of unrotated factor loadings (deprecated; use `box26` instead).

Details

Loading the data makes two objects available:

`box26` is a 26×3 unrotated factor loading matrix for 26 variables measured on a set of boxes. This is the recommended matrix for use in examples and benchmarking. It appears frequently in the rotation literature as a benchmark dataset for comparing rotation criteria, particularly for assessing local minima.

`box20` is a 20×3 unrotated factor loading matrix for 20 variables measured on the same set of boxes. **Deprecated:** `box20` will be removed in a future version of `GPArotation`. Use `box26` instead.

Both matrices are suitable as input to `GPForth`, `GPFoblq`, and all rotation wrapper functions in `GPArotation`. For a richer collection of factor analysis datasets, see the `psych` package.

Source

Thurstone, L.L. (1947). *Multiple Factor Analysis*. University of Chicago Press.

See Also

`GPForth`, `rotations`, `Harman`, `CCAI`, `WansbeekMeijer`

vgQ *Rotation Criteria: Value and Gradient Functions*

Description

Rotation criterion functions that compute the value and gradient of the rotation objective Q. Not exported from the package NAMESPACE.

Usage

```
vgQ.oblimin(L, gam=0)
vgQ.quartimin(L)
vgQ.target(L, Target=NULL)
vgQ.pst(L, W=NULL, Target=NULL)
vgQ.oblimax(L)
vgQ.entropy(L)
vgQ.quartimax(L)
vgQ.varimax(L)
vgQ.simplimax(L, k=nrow(L))
vgQ.bentler(L)
vgQ.tandemI(L)
vgQ.tandemII(L)
vgQ.geomin(L, delta=0.01)
vgQ.bigeomin(L, delta=0.01)
vgQ.cf(L, kappa=0)
vgQ.infomax(L)
vgQ.mccammon(L)
vgQ.varimin(L)
vgQ.bifactor(L)
vgQ.lp.wls(L, W)
```

Arguments

L	a factor loading matrix.
gam	0 = Quartimin, 0.5 = Biquartimin, 1 = Covarimin.
Target	rotation target matrix for objective calculation.
W	weight matrix; weighting of each element in target rotation (vgQ.pst) or in iterative reweighted least squares (vgQ.lp.wls).
k	number of near-zero loadings to target.
delta	small constant added to Λ^2 for numerical stability in the objective calculation.
kappa	complexity weight for the Crawford-Ferguson family; see cfT for details.

Details

The vgQ.* functions are called internally by the gradient projection optimization routines and would typically not be used directly. They can be inspected using :::, for example: GPArotation:::vgQ.oblimin. The gradient projection algorithms are described in Bernaards and Jennrich (2005).

The T or Q suffix on rotation function names should be omitted for the corresponding vgQ.* function. For example, GPArotation:::vgQ.target is the criterion used by both targetT and targetQ.

vgQ.oblimin	orthogonal or oblique	oblimin family
vgQ.quartimin	oblique	oblimin with gam = 0
vgQ.target	orthogonal or oblique	target rotation
vgQ.pst	orthogonal or oblique	partially specified target rotation
vgQ.oblimax	oblique	maximizes overall kurtosis of loadings
vgQ.entropy	orthogonal	minimum entropy
vgQ.quartimax	orthogonal	maximizes variance of squared loadings within variables
vgQ.varimax	orthogonal	maximizes variance of squared loadings within factors
vgQ.simplimax	oblique	minimizes the k smallest squared loadings
vgQ.bentler	orthogonal or oblique	Bentler invariant pattern simplicity
vgQ.tandemI	orthogonal	Tandem principle I criterion
vgQ.tandemII	orthogonal	Tandem principle II criterion
vgQ.geomin	orthogonal or oblique	minimizes geometric mean of squared loadings
vgQ.bigeomin	orthogonal or oblique	geomin with a general factor in column 1
vgQ.cf	orthogonal or oblique	Crawford-Ferguson family
vgQ.infomax	orthogonal or oblique	infomax information criterion
vgQ.mccammon	orthogonal	McCammon minimum entropy ratio
vgQ.varimin	orthogonal	varimin criterion
vgQ.bifactor	orthogonal or oblique	bifactor/biquartimin rotation
vgQ.lp.wls	orthogonal or oblique	iterative reweighted least squares for L^p rotation

See rotations for details on rotation arguments.

New rotation criteria can be added by defining a function named vgQ.newmethod. The function takes L as its first argument, plus any additional arguments, and must return a list with elements f, Gq, and Method.

Gradient projection *without* derivatives can be performed using the GPArotateDF package; type vignette("GPArotateDF", package = "GPArotation") at the command line.

Value

A list with:

f	The value of the rotation criterion at L.
Gq	The gradient of the criterion at L.
Method	A string indicating the criterion name.

Author(s)

Coen A. Bernaards and Robert I. Jennrich with some R modifications by Paul Gilbert.

References

Bernaards, C.A. and Jennrich, R.I. (2005) Gradient Projection Algorithms and Software for Arbitrary Rotation Criteria in Factor Analysis. *Educational and Psychological Measurement*, **65**, 676–696. doi: 10.1177/0013164404272507

See Also

rotations, GPFRSorth

Examples

```
GPArotation:::vgQ.oblimin
```

WansbeekMeijer *Netherlands Television Viewership Data*

Description

Correlation matrix for Netherlands television viewership, used as an example dataset for factor rotation. From Wansbeek and Meijer (2000), page 171.

Usage

```
data(WansbeekMeijer)
```

Format

NetherlandsTV is a list with components:

- $cov: a 7×7 numeric correlation matrix
- $n.obs: sample size (2154)

Details

NetherlandsTV is a list with components $cov (a 7×7 correlation matrix for 7 television viewership variables measured in the Netherlands) and $n.obs (sample size, 2154). Use cov2cor(NetherlandsTV\$cov) to obtain the correlation matrix, or pass NetherlandsTV directly to factanal which handles the list structure automatically. It is used throughout the GPArotation documentation and vignettes as an example dataset for oblique rotation with 2 or 3 factors.

Source

Wansbeek, T. and Meijer, E. (2000). *Measurement Error and Latent Variables in Econometrics*. North-Holland.

See Also

GPForth, rotations, Thurstone, Harman, CCAI

Examples

```
data(WansbeekMeijer, package = "GPArotation")

# Correlation matrix
round(cov2cor(NetherlandsTV$cov), 2)

# factanal picks up n.obs automatically from the list
factanal(factors = 2, covmat = NetherlandsTV, rotation = "none")

# Two-step oblique rotation
fa.unrotated <- factanal(factors = 3, covmat = NetherlandsTV,
                         rotation = "none")
oblimin(loadings(fa.unrotated), randomStarts = 100)
```

Index