Package 'GPArotation'

April 12, 2025

Version 2025.3-1

Title Gradient Projection Factor Rotation

Depends R (>= 2.0.0)

Description Gradient Projection Algorithms for Factor Rotation.
For details see ?GPArotation. When using this package, please cite
Bernaards and Jennrich (2005) <doi:10.1177/0013164404272507>
'Gradient Projection Algorithms and Software
for Arbitrary Rotation Criteria in Factor Analysis'.

LazyData yes

Imports stats

License GPL (>= 2)

URL https://optimizer.r-forge.r-project.org/GPArotation_www/

NeedsCompilation no

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

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

Repository CRAN

Date/Publication 2025-04-12 21:40:02 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) publications.

Additionally 2 rotation methods are provided that do not rely on GP (eiv and echelon)

Package:	GPArotation
Depends:	R (>= 2.0.0)
License:	GPL Version 2.
URL:	https://optimizer.r-forge.r-project.org/GPArotation_www/

Index of functions:

Wrapper functions that include random starts option

`GPFRSorth`	Orthogonal rotation with random starts
`GPFRSorth`	Oblique rotation with random starts

Gradient Projection Rotation Algorithms (code unchanged since 2008)

`GPForth`	Orthogonal rotation function
`GPForth`	Oblique rotation function

Utility functions

`print.GPArotation`	Print results (S3 level function)
`summary.GPArotation`	Summary of results (S3 level function)
`Random.Start`	Generate random a starting matrix
`NormalizingWeight`	Kaiser normalization (not exported from NAMESPACE)

GPForth.lp	Single start L^p orthogonal rotation function
GPFoblq.lp	Single start L^p oblique rotation function

Rotations using the gradient projection algorithm

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's Invariant Pattern Simplicity rotation
bentlerQ	Oblique Bentler's Invariant Pattern Simplicity rotation
tandemI	The Tandem Criteria Principle I rotation
tandemII	The 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

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

vgQ routines to compute value and gradient of the criterion (not exported from NAMESPACE)

vgQ.oblimin	Oblimin vgQ
vgQ.quartimin	Quartimin vgQ
vgQ.target	Target vgQ
vgQ.pst	Partially Specified Target vgQ
vgQ.oblimax	Oblimax vgQ
vgQ.entropy	Minimum Entropy vgQ
vgQ.quartimax	Quartimax vgQ
vgQ.varimax	Varimax vgQ
vgQ.simplimax	Simplimax vgQ
vgQ.bentler	Bentler's Invariant Pattern Simplicity vgQ
vgQ.tandemI	The Tandem Criteria Principle I vgQ
vgQ.tandemII	The Tandem Criteria Principle II vgQ
vgQ.geomin	Geomin vgQ
vgQ.bigeomin	Bi-Geomin vgQ
vgQ.cf	Crawford-Ferguson Family vgQ
vgQ.infomax	Infomax vgQ
vgQ.mccammon	McCammon Minimum Entropy Ratio vgQ
vgQ.varimin	Varimin vgQ
vgQ.bifactor	Bifactor vgQ
vgQ.lp.wls	Weighted Least Squares vgQ for L^p rotation

Data sets included in the GPArotation package

Harman	Initial factor loading matrix Harman8 for Harman's 8 physical variables
Thurstone	box20 and box26 initial factor loadings matrices
WansbeekMeijer	NetherlandsTV dataset

Author(s)

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

Code is modified from original source 'splusfunctions.net' available at https://optimizer.r-forge.r-project.org/GPArotation_www/.

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.

Theory of gradient projection algorithms may be found in:

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

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

See Also

GPFRSorth, GPFRSoblq, rotations, vgQ

`echelon` *Echelon Rotation*

Description

Rotate to an echelon parameterization.

Usage

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

Arguments

`L`	a factor loading matrix
`reference`	indicates rows of loading matrix that should be used to determine the rotation transformation.
`...`	additional arguments discarded.

Details

The loadings matrix is rotated so the k rows of the loading matrix indicated by `reference` are the Cholesky factorization given by `t(chol(L[reference,] %*% t(L[reference,])))`. This defines the rotation transformation, which is then also applied to other rows to give the new loadings matrix.

The optimization is not iterative and does not use the GPA algorithm. The function can be used directly or the function name can be passed to factor analysis functions like `factanal`. An orthogonal solution is assumed (so Φ is identity).

The default uses the first k rows as the reference. If the submatrix of `L` indicated by reference is singular then the rotation will fail and the user needs to supply a different choice of rows.

One use of this parameterization is for obtaining good starting values (so it may appear strange to rotate towards this solution afterwards). It has a few other purposes:

(1) It can be useful for comparison with published results in this parameterization.

(2) The S.E.s are more straightforward to compute, because it is the solution to an unconstrained optimization (though not necessarily computed as such).

(3) The models with k and (k+1) factors are nested, so it is more straightforward to test the k-factor model versus the (k+1)-factor model. In particular, in addition to the LR test (which does not depend on the rotation), now the Wald test and LM test can be used as well. For these, the test of a k-factor model versus a (k+1)-factor model is a joint test whether all the free parameters (loadings) in the (k+1)st column of `L` are zero.

(4) For some purposes, only the subspace spanned by the factors is important, not the specific parameterization within this subspace.

(5) The back-predicted indicators (explained portion of the indicators) do not depend on the rotation method. Combined with the greater ease to obtain correct standard errors of this method, 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).

Value

A GPArotation object which is a list with elements

`loadings`	The new loadings matrix.
`Th`	The rotation.
`method`	A string indicating the rotation objective function ("echelon").
`orthogonal`	For consistency with other rotation results. Always TRUE.
`convergence`	For consistency with other rotation results. Always TRUE.

Author(s)

Erik Meijer and Paul Gilbert.

References

Roderick P. McDonald (1999) *Test Theory: A Unified Treatment*, Mahwah, NJ: Erlbaum.

Tom Wansbeek and Erik Meijer (2000) *Measurement Error and Latent Variables in Econometrics*, Amsterdam: North-Holland.

See Also

`eiv`, `rotations`, `GPForth`, `GPFoblq`

Examples

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

fa.ech <- echelon(fa.unrotated$loadings)

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

cbind(loadings(fa.unrotated), loadings(fa.ech), loadings(fa.ech2))

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

Rotate to errors-in-variables representation.

Usage

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

Arguments

`L`	a factor loading matrix
`identity`	indicates rows which should be identity matrix.
`...`	additional arguments discarded.

Details

This function rotates to an errors-in-variables representation. The optimization is not iterative and does not use the GPA algorithm. The function can be used directly or the function name can be passed to factor analysis functions like `factanal`.

The loadings matrix is rotated so the k rows indicated by `identity` form an identity matrix, and the remaining $M - k$ rows are free parameters. Φ is also free. The default makes the first k rows the identity. If inverting the matrix of the rows indicated by `identity` fails, the rotation will fail and the user needs to supply a different choice of rows.

Not all authors consider this representation to be a rotation. Viewed as a rotation method, it is oblique, with an explicit solution: given an initial loadings matrix L partitioned as $L = (L_1^T, L_2^T)^T$, then (for the default `identity`) the new loadings matrix is $(I, (L_2 L_1^{-1})^T)^T$ and $\Phi = L_1 L_1^T$, where I is the k by k identity matrix. It is assumed that $\Phi = I$ for the initial loadings matrix.

One use of this parameterization is for obtaining good starting values (so it looks a little strange to rotate towards this solution afterwards). It has a few other purposes: (1) It can be useful for comparison with published results in this parameterization; (2) The S.E.s are more straightfoward to compute, because it is the solution to an unconstrained optimization (though not necessarily computed as such); (3) One may have an idea about which reference variables load on only one factor, but not impose restrictive constraints on the other loadings, so, in a nonrestrictive way, it has similarities to CFA; (4) For some purposes, only the subspace spanned by the factors is important, not the specific parameterization within this subspace; (5) The back-predicted indicators (explained portion of the indicators) do not depend on the rotation method. Combined with the greater ease to obtain correct standard errors of this method, this allows easier and more accurate prediction-standard errors.

Value

A GPArotation object which is a list with elements

`loadings`	The new loadings matrix.
`Th`	The rotation.
`method`	A string indicating the rotation objective function ("eiv").
`orthogonal`	For consistency with other rotation results. Always FALSE.
`convergence`	For consistency with other rotation results. Always TRUE.
`Phi`	The covariance matrix of the rotated factors.

Author(s)

Erik Meijer and Paul Gilbert.

References

Gösta Hägglund. (1982). "Factor Analysis by Instrumental Variables Methods." *Psychometrika*, 47, 209–222.

Sock-Cheng Lewin-Koh and Yasuo Amemiya. (2003). "Heteroscedastic factor analysis." *Biometrika*, 90, 85–97.

Tom Wansbeek and Erik Meijer (2000) *Measurement Error and Latent Variables in Econometrics*, Amsterdam: North-Holland.

See Also

`echelon`, `rotations`, `GPForth`, `GPFoblq`

Examples

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

fa.eiv <- eiv(fa.unrotated$loadings)

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

cbind(loadings(fa.unrotated), loadings(fa.eiv), loadings(fa.eiv2))

fa.eiv3 <- eiv(fa.unrotated$loadings, identity=6:7)
cbind(loadings(fa.unrotated), loadings(fa.eiv), loadings(fa.eiv3))
```

GPA *Rotation Optimization*

Description

Gradient projection rotation optimization routine used by various rotation objective.

Usage

```
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)

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)
```

Arguments

A	initial factor loadings matrix for which the rotation criterian 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 criterian.
methodArgs	a list ofmethodArgs arguments passed to the rotation objective
randomStarts	number of random starts (GPFRSorth and GPFRSoblq)

Details

Gradient projection (GP) rotation optimization routines developed by Jennrich (2001, 2002) and Bernaards and Jennrich (2005). 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. A rotation of a matrix A is defined as `A %*% solve(t(Th))`. In case of orthogonal rotation, the factors the rotation matrix Tmat is orthonormal, and the rotation simplifies to `A %*% Th`. The rotation matrix Th is computed by GP rotation.

The GPFRsorth and GPFRSoblq functions are the primary functions for orthogonal and oblique rotations, respectively. These two functions serve as wrapper functions for GPForth and GPFoblq, with the added functionality of multiple random starts. GPForth is the main GP algorithm for orthogonal rotation. GPFoblq is the main GP algorithm for oblique rotation. The GPForth and GPFoblq may be also be called directly.

Arguments in the wrapper functions GPFRsorth and GPFRSoblq are passed to GP algorithms. Functions require an initial loadings matrix A which fixes the equivalence class over which the optimization is done. It must be the solution to the orthogonal factor analysis problem as obtained from factanal or other factor estimation routines. The initial rotation matrix is given by the Tmat. By default the GP algorithm use the identity matrix as the initial rotation matrix.

For some rotation criteria local minima may exist. To start from random initial rotation matrices, the randomStarts argument is available in GPFRSorth and GPFRSoblq. The returned object includes the rotated loadings matrix with the lowest criterion value f among attemnpted starts.Technically, this does not have to be the global minimum. The randomStarts argument is not available GPForth and GPFoblq. However, for GPForth and GPFoblq a single random initial rotation matrix may be set by Tmat = Random.Start(ncol(A)).

The argument method can be used to specify a string indicating the rotation objective. Oblique rotation defaults to "quartimin" and orthogonal rotation defaults to "varimax". Available rotation objectives are "oblimin", "quartimin", "target", "pst", "oblimax", "entropy", "quartimax", "Varimax", "simplimax", "bentler", "tandemI", "tandemII", "geomin", "cf", "infomax", "mccammon", "bifactor", "lp" and "varimin". The string is prefixed with "vgQ." to give the actual function call. See vgQ for details.

Some rotation criteria ("oblimin", "target", "pst", "simplimax", "geomin", "cf", "lp") require one or more additional arguments. See rotations for details and default values, if applicable.

Note that "lp" rotation uses a modeified version of the GPA rotation algorithm; for details on the use of L^p rotation, please see Lp rotation.

For examples of the indirect use see rotations.

The argument normalize gives an indication of if and how any normalization should be done before rotation, and then undone after rotation. If normalize is FALSE (the default) no normalization is done. If normalize is TRUE then Kaiser normalization is done. (So squared row entries of normalized A sum to 1.0. This is sometimes called Horst normalization.) If normalize is a vector of length equal to the number of indicators (= number of rows of A) then the colums are divided by normalize before rotation and multiplied by normalize after rotation. If normalize is a function then it should take A as an argument and return a vector which is used like the vector above. See Nguyen and Waller (2022) for detailed investigation of normalization on factor rotations, including potential effect on qualitative interpretation of loadings.

Value

A GPArotation object which is a list with elements

loadings	The rotated loadings, one column for each factor. If randomStarts were requested then this is the rotated loadings matrix with the lowest criterion value.
Th	The rotation matrix, loadings %*% t(Th) = A.
Table	A matrix recording the iterations of the rotation optimization.
method	A string indicating the rotation objective function.
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. This will be the identity matrix for orthogonal rotations so is omitted (NULL) for the result from GPFRSorth and GPForth.

Gq	The gradient of the objective function at the rotated loadings.
randStartChar	A vector with characteristics of random starts (GPFRSorth and GPFRSoblq only; omitted if 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.

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

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

Nguyen, H.V. and Waller, N.G. (2022). Local minima and factor rotations in exploratory factor analysis. *Psychological Methods*. Advance online publication. https://doi.org/10.1037/met0000467

See Also

Random.Start, factanal, oblimin, quartimin, targetT, targetQ, pstT, pstQ, oblimax, entropy, quartimax, Varimax, varimax, simplimax, bentlerT, bentlerQ, tandemI, tandemII, geominT, geominQ, bigeominT, bigeominQ, cfT, cfQ, equamax, parsimax, infomaxT, infomaxQ, mccammon, varimin, bifactorT, bifactorQ, lpT, lpQ,

Examples

```
# see rotations for more examples

data(Harman, package = "GPArotation")
GPFRSorth(Harman8, method = "quartimax")
quartimax(Harman8)
GPFRSoblq(Harman8, method = "quartimin", normalize = TRUE)
loadings( quartimin(Harman8, normalize = TRUE) )

# using random starts
data("WansbeekMeijer", package = "GPArotation")
fa.unrotated <- factanal(factors = 3, covmat=NetherlandsTV, normalize=TRUE, rotation="none")
GPFRSoblq(loadings(fa.unrotated), normalize = TRUE, method = "oblimin", randomStarts = 100)
oblimin(loadings(fa.unrotated), randomStarts=100)
data(Thurstone, package = "GPArotation")
geominQ(box26, normalize = TRUE, randomStarts=100)

# displaying results of factor analysis rotation output
origdigits <- options("digits")
Abor.unrotated <- factanal(factors = 2, covmat = ability.cov, rotation = "none")
Abor <- oblimin(loadings(Abor.unrotated), randomStarts = 20)
Abor
print(Abor)
```

```
print(Abor, sortLoadings=FALSE) #this matches the output passed to factanal
print(Abor, Table=TRUE)
print(Abor, rotateMat=TRUE)
print(Abor, digits=2)
# by default provides the structure matrix for oblique rotation
summary(Abor)
summary(Abor, Structure=FALSE)
options(digits = origdigits$digits)

# GPArotation output does sort loadings, but use print to obtain if needed
set.seed(334)
xusl <- quartimin(Harman8, normalize = TRUE, randomStarts=100)
# loadings without ordering (default)
loadings(xusl)
max(abs(print(xusl)$loadings - xusl$loadings)) == 0 # FALSE
# output sorted loadings via print (not default)
xsl <- print(xusl)
max(abs(print(xsl)$loadings - xsl$loadings)) == 0 # TRUE

# Kaiser normalization is used when normalize=TRUE
factanal(factors = 2, covmat = ability.cov, rotation = "oblimin",
 control=list(rotate=list(normalize = TRUE)))
# Cureton-Mulaik normalization can be done by passing values to the rotation
# may result in convergence problems
NormalizingWeightCM <- function (L) {
  Dk <- diag(sqrt(diag(L %*% t(L)))^-1) %*% L
  wghts <- rep(0, nrow(L))
  fpls <- Dk[, 1]
  acosi <- acos(ncol(L)^(-1/2))
  for (i in 1:nrow(L)) {
   num <- (acosi - acos(abs(fpls[i])))
   dem <- (acosi - (function(a, m) ifelse(abs(a) < (m^(-1/2)), pi/2, 0))(fpls[i], ncol(L)))
      wghts[i] <- cos(num/dem * pi/2)^2 + 0.001
   }
  Dv <- wghts * sqrt(diag(L %*% t(L)))^-1
  Dv
}
quartimin(Harman8, normalize = NormalizingWeightCM(Harman8), randomStarts=100)
quartimin(Harman8, normalize = TRUE, randomStarts=100)
```

Harman *Example Data from Harman*

Description

Harman8 is initial factor loading matrix for Harman's 8 physical variables.

Usage

```
data(Harman)
```

Format

The object Harman8 is a matrix.

Details

The object Harman8 is loaded from the data file Harman.

Source

Harman, H. H. (1976) *Modern Factor Analysis*, Third Edition Revised, University of Chicago Press.

See Also

GPForth, Thurstone, WansbeekMeijer

---

lp *L^p Rotation*

Description

Performs L^p rotation to obtain sparse loadings.

Usage

```
GPForth.lp(A, Tmat=diag(rep(1, ncol(A))), p=1, normalize=FALSE, eps=1e-05,
       maxit=10000, gpaiter=5)
GPFoblq.lp(A, Tmat=diag(rep(1, 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 $0 < p =< 1$.
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 GPA rotation. The goal is to decrease the objective value, not optimize the inner loop. Warnings may appear, but they can be ignored if the main loop converges.

Details

These functions optimize an L^p rotation objective, where `0 < p =< 1`. A smaller `p` promotes sparsity in the loading matrix but increases computational difficulty. For guidance on choosing `p`, see the Concluding Remarks in the references.

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

The optimization method follows an iterative reweighted least squares (IRLS) approach. It approximates the nonsmooth objective function with a smooth weighted least squares function in the main loop and optimizes it using GPA in the inner loop.

Normalization is not recommended for L^p rotation. Its use may have unexpected results.

Value

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

`loadings`	Rotated loadings matrix, with one column per factor. If `randomStarts` were used, this contains the loadings with the lowest criterion value.
`Th`	Rotation matrix, satisfying `loadings %*% t(Th) = A`.
`Table`	Matrix recording iteration details during optimization.
`method`	String indicating the rotation objective function.
`orthogonal`	Logical indicating whether the rotation is orthogonal.
`convergence`	Logical indicating whether convergence was achieved. Convergence is controlled element-wise by 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., & Moustaki, I. (2023). Rotation to sparse loadings using L^p losses and related inference problems. *Psychometrika*, **88**(2), 527–553.

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 from random position
fa.lpT1 <- GPForth.lp(loadings(fa.unrotated), p = 1)
# 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)

p <- 1
# Oblique rotation
# Single start
fa.lpQ1 <- GPFoblq.lp(loadings(fa.unrotated), p = p)
# 10 random starts
fa.lpQ <- lpQ(loadings(fa.unrotated), p = p, randomStarts = 10)
summary(fa.lpQ, Structure = TRUE)


# this functions ensures consistent ordering of factors of a
# GPArotation object for cleaner comparison
# Inspired by fungible::orderFactors and fungible::faSort functions
sortFac <- function(x){
  # Only works for object of class GPArotation
  if (!inherits(x, "GPArotation")) {stop("argument not of class GPArotation")}
  cln <- colnames(x$loadings)
  # ordering for oblique slightly different from orthogonal
  ifelse(x$orthogonal, vx <- colSums(x$loadings^2),
    vx <- diag(x$Phi %*% t(x$loadings) %*% x$loadings) )
  # sort by squared loadings from high to low
  vxo <- order(vx, decreasing = TRUE)
  vx <- vx[vxo]
  # maintain the right sign
  Dsgn <- diag(sign(colSums(x$loadings^3))) [ , vxo]
  x$Th <- x$Th %*% Dsgn
  x$loadings <- x$loadings %*% Dsgn
  if (match("Phi", names(x))) {
          # If factor is negative, reverse corresponding factor correlations
          x$Phi <- t(Dsgn) %*% x$Phi %*% Dsgn
    }
  colnames(x$loadings) <- cln
  x
}

# seed set to see the results of sorting
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 ordered factor loadings
res <- round(cbind(sortFac(fa.lpQ1)$loadings, sortFac(fa.lpQ0.5)$loadings,
  sortFac(fa.geo)$loadings),3)
print(c("oblique-  Lp 1           Lp 0.5         geomin")); print(res)

# without ordered factor loadings
res <- round(cbind(fa.lpQ1$loadings, fa.lpQ0.5$loadings, fa.geo$loadings),3)
```

```
print(c("oblique-  Lp 1          Lp 0.5        geomin")); print(res)

options(warn = 0)
```

`Random.Start` *Generate a Random Orthogonal Rotation*

Description

Random orthogonal rotation to use as Tmat matrix to start `GPFRSorth`, `GPFRSoblq`, `GPForth`, or `GPFoblq`.

Usage

```
Random.Start(k)
```

Arguments

`k`	An integer indicating the dimension of the square matrix.

Details

The random start function produces an orthogonal matrix with columns of length one based on the QR decompostion. This randomization procedures follows the logic of Stewart(1980) and Mezzari(2007), as of GPArotation version 2024.2-1.

Value

An orthogonal matrix.

Author(s)

Coen A. Bernaards and Robert I. Jennrich with some R modifications by Paul Gilbert. Additional input from 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. http://www.jstor.org/stable/2156882

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

See Also

`GPFRSorth`, `GPFRSoblq`, `GPForth`, `GPFoblq`, `rotations`

Examples

```
# Generate a random ortogonal matrix of dimension 5 x 5
Random.Start(5)

# function for generating orthogonal or oblique random matrix
Random.Start <- function(k = 2L,orthogonal=TRUE){
  mat <- matrix(rnorm(k*k),k)
  if (orthogonal){
    qr.out <- qr(matrix(rnorm(k * k), nrow = k, ncol = k))
    Q <- qr.Q(qr.out)
    R <- qr.R(qr.out)
    R.diag <- diag(R)
    R.diag2 <- R.diag/abs(R.diag)
    ans <- t(t(Q) * R.diag2)
    ans
    }
  else{
ans <- mat %*% diag(1/sqrt(diag(crossprod(mat))))
  }
  ans
  }

data("Thurstone", package="GPArotation")
simplimax(box26,Tmat = Random.Start(3, orthogonal = TRUE))
simplimax(box26,Tmat = Random.Start(3, orthogonal = FALSE))

# covariance matrix is Phi = t(Th) %*% Th
rms <- Random.Start(3, FALSE)
t(rms) %*% rms # covariance matrix because oblique rms
rms <- Random.Start(3, TRUE)
t(rms) %*% rms # identity matrix because orthogonal rms
```

`rotations` *Rotations*

Description

Optimize factor loading rotation objective.

Usage

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

```
  targetQ(A, Tmat=diag(ncol(A)), Target=NULL, normalize=FALSE, eps=1e-5,
   maxit=1000, randomStarts=0, L=NULL)
  pstT(A, Tmat=diag(ncol(A)), W=NULL, Target=NULL, normalize=FALSE, eps=1e-5,
   maxit=1000, randomStarts=0, L=NULL)
  pstQ(A, 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, eps=1e-5, maxit=1000, randomStarts=0)
entropy(A, Tmat=diag(ncol(A)), normalize=FALSE, eps=1e-5, maxit=1000, randomStarts=0)
quartimax(A, Tmat=diag(ncol(A)), normalize=FALSE, eps=1e-5,maxit=1000,randomStarts=0)
Varimax(A, Tmat=diag(ncol(A)), normalize=FALSE, eps=1e-5, maxit=1000, randomStarts=0)
  simplimax(A, Tmat=diag(ncol(A)), k=nrow(A), normalize=FALSE, eps=1e-5,
   maxit=1000, randomStarts=0)
bentlerT(A, Tmat=diag(ncol(A)), normalize=FALSE, eps=1e-5, maxit=1000, randomStarts=0)
bentlerQ(A, Tmat=diag(ncol(A)), normalize=FALSE, eps=1e-5, maxit=1000, randomStarts=0)
tandemI(A, Tmat=diag(ncol(A)), normalize=FALSE, eps=1e-5, maxit=1000, randomStarts=0)
tandemII(A, Tmat=diag(ncol(A)), normalize=FALSE, eps=1e-5, maxit=1000, randomStarts=0)
  geominT(A, Tmat=diag(ncol(A)), delta=.01, normalize=FALSE, eps=1e-5,
   maxit=1000, randomStarts=0)
  geominQ(A, Tmat=diag(ncol(A)), delta=.01, normalize=FALSE, eps=1e-5,
   maxit=1000, randomStarts=0)
  bigeominT(A, Tmat=diag(ncol(A)), delta=.01, normalize=FALSE, eps=1e-5,
   maxit=1000, randomStarts=0)
  bigeominQ(A, Tmat=diag(ncol(A)), delta=.01, normalize=FALSE, eps=1e-5,
   maxit=1000, randomStarts=0)
  cfT(A, Tmat=diag(ncol(A)), kappa=0, normalize=FALSE, eps=1e-5,
   maxit=1000, randomStarts=0)
  cfQ(A, Tmat=diag(ncol(A)), kappa=0, normalize=FALSE, eps=1e-5,
   maxit=1000, randomStarts=0)
  equamax(A, Tmat=diag(ncol(A)), kappa=ncol(A)/(2*nrow(A)), normalize=FALSE,
   eps=1e-5, maxit=1000, randomStarts = 0)
  parsimax(A, Tmat=diag(ncol(A)), kappa=(ncol(A)-1)/(ncol(A)+nrow(A)-2),
   normalize=FALSE, eps=1e-5, maxit=1000, randomStarts = 0)
infomaxT(A, Tmat=diag(ncol(A)), normalize=FALSE, eps=1e-5, maxit=1000, randomStarts=0)
infomaxQ(A, Tmat=diag(ncol(A)), normalize=FALSE, eps=1e-5, maxit=1000, randomStarts=0)
mccammon(A, Tmat=diag(ncol(A)), normalize=FALSE, eps=1e-5, maxit=1000, randomStarts=0)
varimin(A, Tmat=diag(ncol(A)), normalize=FALSE, eps=1e-5, maxit=1000, randomStarts=0)
bifactorT(A, Tmat=diag(ncol(A)), normalize=FALSE, eps=1e-5, maxit=1000,randomStarts=0)
bifactorQ(A, Tmat=diag(ncol(A)), normalize=FALSE, eps=1e-5, maxit=1000,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	0=Quartimin, .5=Biquartimin, 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	parameter passed to optimization routine (GPForth or GPFoblq).
maxit	parameter passed to optimization routine (GPForth or GPFoblq).
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).

Rotations which are available are

oblimin	oblique	oblimin family
quartimin	oblique	
targetT	orthogonal	target rotation
targetQ	oblique	target rotation
pstT	orthogonal	partially specified target rotation
pstQ	oblique	partially specified target rotation
oblimax	oblique	
entropy	orthogonal	minimum entropy
quartimax	orthogonal	
varimax	orthogonal	
simplimax	oblique	
bentlerT	orthogonal	Bentler's invariant pattern simplicity criterion
bentlerQ	oblique	Bentler's invariant pattern simplicity criterion
tandemI	orthogonal	Tandem principle I criterion
tandemII	orthogonal	Tandem principle II criterion
geominT	orthogonal	
geominQ	oblique	
bigeominT	orthogonal	
bigeominQ	oblique	
cfT	orthogonal	Crawford-Ferguson family

cfQ	oblique	Crawford-Ferguson family
equamax	orthogonal	Crawford-Ferguson family
parsimax	orthogonal	Crawford-Ferguson family
infomaxT	orthogonal	
infomaxQ	oblique	
mccammon	orthogonal	McCammon minimum entropy ratio
varimin	orthogonal	
bifactorT	orthogonal	Jennrich and Bentler bifactor rotation
bifactorQ	oblique	Jennrich and Bentler biquartimin rotation
lpT	orthogonal	L^p rotation
lpQ	oblique	L^p rotation

Note that Varimax defined here uses vgQ.varimax and is not varimax defined in the stats package. stats:::varimax does Kaiser normalization by default whereas Varimax defined here 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, R.I. and Bentler, P.M. (2011).

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 (includes elements used by factanal) with:

loadings	Lh from GPFRSorth or GPFRSoblq.
Th	Th from GPFRSorth or GPFRSoblq.
Table	Table from GPForth or GPFoblq.
method	A string indicating the rotation objective function.
orthogonal	A logical indicating if the rotation is orthogonal.
convergence	Convergence indicator from GPFRSorth or GPFRSoblq.
Phi	t(Th) %*% Th. The covariance matrix of the rotated factors. This will be the identity matrix for orthogonal rotations so is omitted (NULL) for the result from GPFRSorth and GPForth.
randStartChar	Vector indicating results from random starts from GPFRSorth or GPFRSoblq

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.

Jennrich, R.I. and Bentler, P.M. (2011) Exploratory bi-factor analysis. *Psychometrika*, **76**.

See Also

factanal, GPFRSorth, GPFRSoblq, vgQ, Harman, box26, WansbeekMeijer,

Examples

```
 # see GPFRSorth and GPFRSoblq for more examples

 # getting loadings matrices
 data("Harman", package="GPArotation")
 qHarman  <- GPFRSorth(Harman8, Tmat=diag(2), method="quartimax")
 qHarman <- quartimax(Harman8)
 loadings(qHarman) - qHarman$loadings   #2 ways to get the loadings

 # factanal loadings used in GPArotation
 data("WansbeekMeijer", package="GPArotation")
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 arguments to factanal (See vignette for a caution)
 # vignette("GPAguide", package = "GPArotation")
 data(ability.cov)
 factanal(factors = 2, covmat = ability.cov, rotation="infomaxT")
 factanal(factors = 2, covmat = ability.cov, rotation="infomaxT",
   control=list(rotate=list(normalize = TRUE, eps = 1e-6)))
# when using factanal for oblique rotation it is best to use the rotation command directly
 # instead of including it in the factanal command (see Vignette).
fa.unrotated <- factanal(factors = 3, covmat=NetherlandsTV, normalize=TRUE, rotation="none")
 quartimin(loadings(fa.unrotated), normalize=TRUE)

 # oblique target rotation of 2 varimax rotated matrices towards each other
 # See vignette for additional context and computation,
 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 <- targetQ(trGermany, Target = trBritain)
 # Difference between rotated loadings matrix and target matrix
 y <- trx$loadings - trBritain

 # partially specified target; See vignette for additional method
 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)
 targetT(A, Target=SPA)

 # using random starts
 data("WansbeekMeijer", package="GPArotation")
fa.unrotated <- factanal(factors = 3, covmat=NetherlandsTV, normalize=TRUE, rotation="none")
 # single rotation with a random start
 oblimin(loadings(fa.unrotated), Tmat=Random.Start(3))
 oblimin(loadings(fa.unrotated), randomStarts=1)
 # multiple random starts
 oblimin(loadings(fa.unrotated), randomStarts=100)

 # assessing local minima for box26 data
 data(Thurstone, package = "GPArotation")
 infomaxQ(box26, normalize = TRUE, randomStarts = 150)
 geominQ(box26, normalize = TRUE, randomStarts = 150)
 # for detailed investigation of local minima, consult package 'fungible'
 # library(fungible)
 # faMain(urLoadings=box26, rotate="geominQ", rotateControl=list(numberStarts=150))
 # library(psych) # package 'psych' with random starts:
 # faRotations(box26, rotate = "geominQ", hyper = 0.15, n.rotations = 150)
```

Thurstone *Example Data from Thurstone*

Description

box20 and box26 are initial factor loading matrices.

Usage

```
data(Thurstone)
```

Format

The objects box20 and box26 are matrices.

Details

The objects box20 and box26 are loaded from the data file Thurstone.

Source

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

See Also

GPForth, Harman, WansbeekMeijer

vgQ *Rotations*

Description

vgQ routines to compute value and gradient of the criterion (not exported from 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=.01)
vgQ.bigeomin(L, delta=.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, .5=Biquartimin, 1=Covarimin.
Target	rotation target for objective calculation.
W	weighting of each element in target and in irls.
k	number of close to zero loadings.
delta	constant added to Λ^2 in objective calculation.
kappa	see details.

Details

The vgQ.* versions of the code are called by the optimization routine and would typically not be used directly, so these methods are not exported from the package NAMESPACE. (They simply return the function value and gradient for a given rotation matrix.) These functions can be printed

through,for example, `GPArotation:::vgQ.oblimin` to view the function `vgQ.oblimin`. The `T` or `Q` ending on function names should be omitted for the `vgQ.*` versions of the code so, for example, use `GPArotation:::vgQ.target` to view the target criterion calculation which is used in both orthogonal and oblique rotation.

`vgQ.oblimin`	orthogonal or oblique	oblimin family
`vgQ.quartimin`	oblique	
`vgQ.target`	orthogonal or oblique	target rotation
`vgQ.pst`	orthogonal or oblique	partially specified target rotation
`vgQ.oblimax`	oblique	
`vgQ.entropy`	orthogonal	minimum entropy
`vgQ.quartimax`	orthogonal	
`vgQ.varimax`	orthogonal	
`vgQ.simplimax`	oblique	
`vgQ.bentler`	orthogonal or oblique	Bentler's invariant pattern simplicity criterion
`vgQ.tandemI`	orthogonal	Tandem principle I criterion
`vgQ.tandemII`	orthogonal	Tandem principle II criterion
`vgQ.geomin`	orthogonal or oblique	
`vgQ.bigeomin`	orthogonal or oblique	
`vgQ.cf`	orthogonal or oblique	Crawford-Ferguson family
`vgQ.cubimax`	orthogonal	
`vgQ.infomax`	orthogonal or oblique	
`vgQ.mccammon`	orthogonal	McCammon minimum entropy ratio
`vgQ.varimin`	orthogonal	varimin criterion
`vgQ.bifactor`	orthogonal or oblique	bifactor/biquartimin rotation
`vgQ.lp.wls`	orthoginal or oblique	iterative reweighted least squares for L^p rotation

See rotations for use of arguments.

New rotation methods can be programmed with a name `"vgQ.newmethod"`. The inputs are the matrix `L`, and optionally any additional arguments. The output should be 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 (which includes elements used by `GPForth` and `GPFoblq`) with:

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

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.

See Also

rotations

Examples

```
GPArotation:::vgQ.oblimin
getAnywhere(vgQ.oblimax)
```

WansbeekMeijer *Factor Example from Wansbeek and Meijer*

Description

Netherlands TV viewership example p 171, Wansbeek and Meijer (2000)

Usage

```
data(WansbeekMeijer)
```

Format

The object NetherlandsTV is a correlation matrix.

Details

The object NetherlandsTV is loaded from the data file WansbeekMeijer.

Source

Tom Wansbeek and Erik Meijer (2000) *Measurement Error and Latent Variables in Econometrics*, Amsterdam: North-Holland.

See Also

GPForth, Thurstone, Harman

Index