Package ‘GPArotation’

November 16, 2023

Version 2023.11-1
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:
   "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 2023-11-16 15:34:00 UTC

R topics documented:

00.GPArotation .................................................. 2
echelon ........................................................... 4
eiv ................................................................. 6
GPA ................................................................. 8
Harman ............................................................ 12
Random.Start .................................................... 12
rotations ......................................................... 14
Thurstone ......................................................... 18
vgQ ............................................................... 19
WansbeekMeijer .................................................. 22
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

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>GPFRSorth</td>
<td>Orthogonal rotation with random starts</td>
</tr>
<tr>
<td>GPFRSorth</td>
<td>Oblique rotation with random starts</td>
</tr>
</tbody>
</table>

Gradient Projection Rotation Algorithms (code unchanged since 2008)

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>GPForth</td>
<td>Orthogonal rotation function</td>
</tr>
<tr>
<td>GPForth</td>
<td>Oblique rotation function</td>
</tr>
</tbody>
</table>

Utility functions

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Random.Start</td>
<td>Generate random a starting matrix</td>
</tr>
<tr>
<td>NormalizingWeight</td>
<td>Kaiser normalization (not exported from NAMESPACE)</td>
</tr>
</tbody>
</table>
Rotations

- 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
- 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

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.cf: Crawford-Ferguson Family vgQ
Data sets included in the GPArotation package

- **Harman**: Initial factor loading matrix for Harman’s 8 physical variables
- **Thurstone**: box20 and box26 initial factor loadings matrices.
- **WansbeekMeijer**: Netherlands TV viewership.

**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


Theory of gradient projection algorithms may be found in:


**See Also**

- `GPFRSorth`, `GPFRSoblq`, `rotations`, `vgQ`

---

**echelon**

**Echelon Rotation**

**Description**

Rotate to an echelon parameterization.

**Usage**

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

Arguments

L
reference

A factor loading matrix
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 $\text{t(chol(L[reference,] %*% t(L[reference,])))}$. 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 \texttt{factanal}. An orthogonal solution is assumed (so $\Phi$ 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 list (which includes elements used by \texttt{factanal}) with:

loadings The new loadings matrix.
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


See Also

eiv, rotations, GPForth, GPFoblq

Examples

data("WansbeekMeijer", package="GPArotation")
fa.unrotated <- factanal(factors = 2, covmat=NetherlandsTV, rotation="none")
f.a.ech <- echelon(fa.unrotated$loadings)
f.a.ech2 <- factanal(factors = 2, covmat=NetherlandsTV, rotation="echelon")
cbind(loadings(fa.unrotated), loadings(f.a.ech), loadings(f.a.ech2))
f.a.ech3 <- echelon(fa.unrotated$loadings, reference=6:7)
cbind(loadings(fa.unrotated), loadings(f.a.ech), loadings(f.a.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. $\Phi$ 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 $(I, (L_2L_1^{-1})^T)^T$ and $\Phi = L_1L_1^T$, 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 straightforward 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 list (which includes elements used by `factanal`) with:

- `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


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 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: a list of methodArgs arguments passed to the rotation objective.

methodArgs: 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 $A \times \text{solve}(\text{t}(\text{Th}))$. In case of orthogonal rotation, the factors the rotation matrix $T_{\text{mat}}$ is orthonormal, and the rotation simplifies to $A \times \text{Th}$. The rotation matrix $T_{\text{h}}$ 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 $T_{\text{mat}}$. 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 attempted 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 $T_{\text{mat}} = \text{Random.Start}(\text{ncol}(A))$.

The argument method can be used to specify a string indicating the rotation objective. Oblique rotations 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, 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") require one or more additional arguments. See reference link(rotations) for details and default values, if applicable.

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 columns 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


See Also

`Random.Start`, `factanal`, `oblimin`, `quartimin`, `targetT`, `targetQ`, `pstT`, `pстQ`, `oblmax`, `entropy`, `quartimax`, `Varimax`, `varimax`, `simplicmax`, `bentlerT`, `bentlerQ`, `tandemI`, `tandemII`, `geominT`, `geominQ`, `cft`, `cfQ`, `equamax`, `parsimax`, `infomaxT`, `infomaxQ`, `mccammon`, `varimin`, `bifactorT`, `bifactorQ`, `promax`

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

Random.Start}

`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**

```r
data(Harman)
```

**Format**

The object Harman8 is a matrix.

**Details**

The object Harman8 is loaded from the data file Harman.

**Source**


**See Also**

`GPForth`, `Thurstone`, `WansbeekMeijer`

---

### Random.Start

**Generate a Random Orthogonal Rotation**

**Description**

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

**Usage**

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

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.

Value

An orthogonal matrix.

Author(s)

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

See Also

GPFRSorth, GPFRSoblq, GPForth, GPFoblq, oblimin

Examples

Random.Start <- function(k,orthogonal=TRUE){
  # routine for generating orthogonal or oblique random matrix
  mat <- matrix(rnorm(k*k),k)
  if (orthogonal){
    ans <- qr.Q(qr(mat))
  } else{
    ans <- mat %*% diag(1/sqrt(diag(crossprod(mat))))
  }
  ans
}

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

### 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 Lambda^2 in objective calculation.
- **kappa**
  - see details.
- **normalize**
  - parameter passed to optimization routine (GPForth or GPForblq).
- **eps**
  - parameter passed to optimization routine (GPForth or GPForblq).
- **maxit**
  - parameter passed to optimization routine (GPForth or GPForblq).
- **randomStarts**
  - parameter passed to optimization routine (GPFRSorth or GPFRSoblq).
- **L**
  - provided for backward compatibility in target rotations only. Use A going forward.

### 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
rotations

- geominQ: 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)

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.

**Value**

A list (which 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**


See Also

GPFRSorth, GPFRSoblq, WansbeekMeijer, eiv, echelon, vgQ, vgQ.oblimin, vgQ.quartimin, vgQ.target, vgQ.pst, vgQ.oblimax, vgQ.entropy, vgQ.quartimax, vgQ.varimax, vgQ.simplimax, vgQ.bentler, vgQ.tandemI, vgQ.tandemII, vgQ.geomin, vgQ.cf, vgQ.infomax, vgQ.mccammon, vgQ.bifactor, vgQ.varimin, factanal, varimax

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, -.038,.087, -.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, -.0291, -.0314, -.0377, .397, .294, .428, -.0075, .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**


**See Also**

GPForth, Harman, WansbeekMeijer
**Description**

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

**Usage**

```r
gvQ.oblimin(L, gam=0)
gvQ.quartimin(L)
gvQ.target(L, Target=NULL)
gvQ.pst(L, W=NULL, Target=NULL)
gvQ.oblimax(L)
gvQ.entropy(L)
gvQ.quartimax(L)
gvQ.varimax(L)
gvQ.simplimax(L, k=nrow(L))
gvQ.bentler(L)
gvQ.tandemI(L)
gvQ.tandemII(L)
gvQ.geomin(L, delta=.01)
gvQ.cf(L, kappa=0)
gvQ.infomax(L)
gvQ.mccammon(L)
gvQ.varimin(L)
gvQ.bifactor(L)
```

**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.
- `k`: number of close to zero loadings.
- `delta`: constant added to Lambda^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.) You can print these functions, but the package name needs to be specified since they are not exported. For example, use 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.
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.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

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

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


See Also

GPForth, Thurstone, Harman
Index

* datasets
  Harman, 12
  Thurstone, 18
  WansbeekMeijer, 22

* multivariate
  echelon, 4
  eiv, 6
  GPA, 8
  Random.Start, 12
  rotations, 14
  vgQ, 19

* package
  Ø0.GPArotation, 2

* rotation
  echelon, 4
  eiv, 6
  GPA, 8
  Random.Start, 12
  rotations, 14
  vgQ, 19
  Ø0.GPArotation, 2
  bentlerQ, 3, 10, 22
  bentlerQ(rotations), 14
  bentlerT, 3, 10, 22
  bentlerT(rotations), 14
  bifactorQ, 3, 10, 22
  bifactorQ(rotations), 14
  bifactorT, 3, 10, 22
  bifactorT(rotations), 14
  box20 (Thurstone), 18
  box26 (Thurstone), 18
  cfQ, 3, 10, 22
  cfQ(rotations), 14
  cfT, 3, 10, 22
  cfT(rotations), 14
  echelon, 4, 8, 17
  eiv, 6, 6, 17
  entropy, 3, 10, 22
  entropy(rotations), 14
  equamax, 3, 10, 22
  equamax(rotations), 14
  factanal, 9, 10, 17
  geominQ, 3, 10, 22
  geominQ(rotations), 14
  geominT, 3, 10, 22
  geominT(rotations), 14
  GPA, 8
  GPArotation(Ø0.GPArotation), 2
  GPArotation-package(Ø0.GPArotation), 2
  GPArotation.Intro(Ø0.GPArotation), 2
  GPForb, 6, 8, 13, 22
  GPForb(GPA), 8
  GPForth, 2, 6, 8, 12, 13, 18, 22
  GPForth(GPA), 8
  GPFRSoblq, 4, 13, 17
  GPFRSoblq(GPA), 8
  GPFRSorth, 2, 4, 13, 17
  GPFRSorth(GPA), 8
  Harman, 4, 12, 18, 22
  Harman8(Harman), 12
  infomaxQ, 3, 10, 22
  infomaxQ(rotations), 14
  infomaxT, 3, 10, 22
  infomaxT(rotations), 14
  mccammon, 3, 10, 22
  mccammon(rotations), 14

  NetherlandsTV(WansbeekMeijer), 22
  NormalizingWeight, 2
  oblimax, 3, 10, 22
  oblimax(rotations), 14
  oblimin, 3, 10, 13, 22

23
<table>
<thead>
<tr>
<th>Method</th>
<th>Page Numbers</th>
</tr>
</thead>
<tbody>
<tr>
<td>oblimin (rotations)</td>
<td>14</td>
</tr>
<tr>
<td>parsimax</td>
<td>3, 10, 22</td>
</tr>
<tr>
<td>promax</td>
<td>10</td>
</tr>
<tr>
<td>pstQ</td>
<td>3, 10, 22</td>
</tr>
<tr>
<td>pstQ (rotations)</td>
<td>14</td>
</tr>
<tr>
<td>pstT</td>
<td>3, 10, 22</td>
</tr>
<tr>
<td>quartimax</td>
<td>3, 10, 22</td>
</tr>
<tr>
<td>quartimin</td>
<td>3, 10, 22</td>
</tr>
<tr>
<td>Random.Start</td>
<td>2, 9, 10, 12</td>
</tr>
<tr>
<td>rotations</td>
<td>4, 6, 8, 9, 14, 21, 22</td>
</tr>
<tr>
<td>simplimax</td>
<td>3, 10, 22</td>
</tr>
<tr>
<td>tandemI</td>
<td>3, 10, 22</td>
</tr>
<tr>
<td>tandemII</td>
<td>3, 10, 22</td>
</tr>
<tr>
<td>targetQ</td>
<td>3, 10, 22</td>
</tr>
<tr>
<td>targetT</td>
<td>3, 10, 22</td>
</tr>
<tr>
<td>Thurstone</td>
<td>4, 12, 18, 22</td>
</tr>
<tr>
<td>Varimax</td>
<td>3, 10, 22</td>
</tr>
<tr>
<td>Varimax (rotations)</td>
<td>14</td>
</tr>
<tr>
<td>varimax</td>
<td>10, 17</td>
</tr>
<tr>
<td>varimin</td>
<td>3, 10, 22</td>
</tr>
<tr>
<td>varimin (rotations)</td>
<td>14</td>
</tr>
<tr>
<td>vgQ</td>
<td>4, 9, 17, 19</td>
</tr>
<tr>
<td>vgQ.bentler</td>
<td>3, 17</td>
</tr>
<tr>
<td>vgQ.bifactor</td>
<td>4, 17</td>
</tr>
<tr>
<td>vgQ.cf</td>
<td>3, 17</td>
</tr>
<tr>
<td>vgQ.entropy</td>
<td>3, 17</td>
</tr>
<tr>
<td>vgQ.geomin</td>
<td>3, 17</td>
</tr>
<tr>
<td>vgQ.infomax</td>
<td>4, 17</td>
</tr>
<tr>
<td>vgQ.mccammon</td>
<td>4, 17</td>
</tr>
<tr>
<td>vgQ.oblimax</td>
<td>3, 17</td>
</tr>
<tr>
<td>vgQ.oblimin</td>
<td>3, 17</td>
</tr>
<tr>
<td>vgQ.pst</td>
<td>3, 17</td>
</tr>
<tr>
<td>vgQ.quartimax</td>
<td>3, 17</td>
</tr>
<tr>
<td>vgQ.quartimin</td>
<td>3, 17</td>
</tr>
<tr>
<td>vgQ.simplimax</td>
<td>3, 17</td>
</tr>
<tr>
<td>WansbeekMeijer</td>
<td>4, 12, 17, 18, 22</td>
</tr>
</tbody>
</table>