Package 'GPArotateDF'

July 21, 2025

Version 2023.11-1

Title Derivative Free Gradient Projection Factor Rotation

Depends R (>= 2.0.0)

Description Derivative Free Gradient Projection Algorithms for Factor Rotation.
For more details see ?GPArotateDF. Theory for these functions can be found in
the following publications:
Jennrich (2004) <doi:10.1007/BF02295647>.
Bernaards and Jennrich (2005) <doi:10.1177/0013164404272507>.

Imports GPArotation

License GPL (>= 2)

NeedsCompilation no

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

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

Repository CRAN

Date/Publication 2023-11-24 22:10:20 UTC

Contents

`00.GPArotateDF` *Derivative-Free Gradient Projection Algorithms for Factor Rotation*

Description

Derivative-Free GPA Rotation for Factor Analysis

The GPArotateDF package contains functions for the rotation of factor loadings matrices without the need for deriving gradients. Both orthogonal and oblique rotation algorithms are available. 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 derivartive-free rotation method implemented in this package is described in Jennrich (2004). The GPArotation package is based on Bernaards and Jennrich (2005).

Package:	GPArotateDF
Depends:	R (>= 2.0.0)
License:	GPL Version 2.
Imports:	GPArotation

Index of functions:

Derivative-Free Gradient Projection Rotation Algorithms

`GPForth.df`	Orthogonal rotation function
`GPFoblq.df`	Oblique rotation function

Rotations

`cubimax.df`	Cubimax rotation
`fssQ.df`	Oblique Forced Simple Structure rotation
`fssT.df`	Orthogonal Forced Simple Structure rotation

ff routines to compute value of the criterion

`ff.bentler`	Bentler's Invariant Pattern Simplicity ff
`ff.cf`	Crawford-Ferguson Family ff
`ff.cubimax`	Cubimax ff
`ff.entropy`	Minimum Entropy ff
`ff.geomin`	Geomin ff
`ff.infomax`	Infomax ff

ff.oblimax	Oblimax ff
ff.pst	Partially Specified Target ff
ff.quartimax	Quartimax ff
ff.quartimin	Quartimin ff
ff.simplimax	Simplimax ff
ff.fss	Forced Simple Structure ff
ff.target	Target ff
ff.varimax	Varimax ff

Utility functions

NormalizingWeight	Kaiser normalization (not exported from NAMESPACE)

Author(s)

Coen A. Bernaards and Robert I. Jennrich

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. (2004). Derivative free gradient projection algorithms for rotation. *Psychometrika*, **69**, 475–480.

See Also

GPForth.df, GPFoblq.df

---

ff.rotationsDF *Rotations*

Description

Optimize factor loading rotation objective.

Usage

```
ff.bentler(L)
ff.cf(L, kappa=0)
ff.cubimax(L)
ff.entropy(L)
ff.geomin(L, delta=0.01)
ff.infomax(L)
```

```
ff.oblimax(L)
ff.pst(L, W=NULL, Target=NULL)
ff.quartimax(L)
ff.quartimin(L)
ff.simplimax(L, k=nrow(L))
ff.fss(L, kij=2)
ff.target(L, Target=NULL)
ff.varimax(L)
```

Arguments

L	a factor loading matrix
kappa	see details.
delta	constant added to Lambda^2 in objective calculation.
Target	rotation target for objective calculation.
W	weighting of each element in target.
k	number of close to zero loadings.
kij	minimum additional number of forced simple structure loadings in a pair of factors.

Details

These functions are used to optimize a rotation objective. The name need to be included in a call to GPForth.df or GPFoblq. Calling the functions itself computes the values but no rotation is performed.

Functions listed here are all exported through NAMESPACE, and primarily serve as examples for programming new rotation methods. New rotation methods can be programmed with a name ff.newmethod. The inputs are the matrix L, and optionally any additional arguments. The output should be a list with elements

f	the value of the criterion at L.
Method	a string indicating the criterion.

Please note that the function value f has to be minimized. If the rotation criterion is supposed to maximize, then use the negative of the criterion to miniize. Functions which are available are

ff.bentler	orthogonal or oblique	Bentler's invariant pattern simplicity criterion
ff.cf	orthogonal or oblique	Crawford-Ferguson family
ff.cubimax	orthogonal	
ff.entropy	orthogonal	minimum entropy
ff.fss	orthogonal or oblique	Forced Simple Structure (see Vignette)
ff.geomin	orthogonal or oblique	
ff.infomax	orthogonal or oblique	
ff.oblimax	oblique	
ff.pst	orthogonal or oblique	partially specified target rotation

ff.quartimax	orthogonal	
ff.quartimin	oblique	
ff.simplimax	oblique	
ff.target	orthogonal or oblique	target rotation
ff.varimax	orthogonal	

The argument kappa parameterizes the family for the Crawford-Ferguson method. If m is the number of factors and p is the number of items 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.

For the argument kij for Forced Simple Structure see rotationsDF.

Value

f	criterion function value.
method	A string indicating the rotation objective function.

Author(s)

Coen A. Bernaards and Robert I. Jennrich

References

Jennrich, R.I. (2004) Derivative free gradient projection algorithms for rotation, Psychometrika: **69**(3), 475–480.

See Also

GPForth.df, GPFoblq.df, fssQ.df, fssT.df, cubimax.df, rotationsDF, factanal

Examples

```
data("Harman", package="GPArotation")
qHarman  <- GPForth.df(Harman8, Tmat=diag(2), method="quartimax")

# define a new function as ff.newname for use with factanal
ff.expomax <- function(L)
{
  f <- -sum(diag(expm1(abs(L))))
  list(f = f, Method = "DF-Expomax")
}
GPForth.df(Harman8, method ="expomax")

expomax.df <- function(L, Tmat = diag(ncol(L)), normalize = FALSE, eps = 1e-5, maxit = 1000){
GPForth.df(L, Tmat=Tmat, method = "expomax", normalize = normalize, eps= eps, maxit = maxit)

}
expomax.df(Harman8, normalize = TRUE)
factanal(factors = 2, covmat = ability.cov, rotation = "expomax.df",
    control = list(rotate =c(normalize = TRUE)))
```

GPADF *Rotation Optimization*

Description

Derivative free gradient projection rotation optimization routine used by various rotation objective.

Usage

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

Details

Derivative free gradient projection rotation optimization routines can be used to rotate a loadings matrix. The rotation criteria in the GPArotation package require a derivative to operate. In certain cases, the derivative is complex or non-existent. The derivative free gradient projection method provides a numerical alternative to the GPArotation package. The functions in the package GPArotateDF follow most of the functionality and logic as in the GPArotation package. Please consult the documentation in GPArotation for further details.

The argument method can be used to specify a string indicating the rotation objective. GPFoblq defaults to "quartimin" and GPForth defaults to "varimax". Available rotation objective functions include "ff.bentler", "ff.cf", "ff.cubimax", "ff.entropy", "ff.fss", "ff.geomin", "ff.infomax", "ff.oblimax", "ff.pst", "ff.quartimax","ff.quartimin", "ff.simplimax", "ff.target", and "ff.varimax". Most of the rotation criteria are avaible in the GPArotation pacakage except for cubimax and Forced Simple Structure.

The rotation criteria are in the functions prefixed by "ff." that are used in the actual function call. The ff.* function call would typically not be used directly, but are needed for rotation. Since these are illustrative of computation, these are all exported from the package namespace. New criteria for use with derivative free GP rotation do require a function of the type ff.newCriterionName that provides value for complexity f, and name of method.

Some rotation criteria (including "simplimax", "pst", "target", "cf", "fss") require one or more additional arguments. Check GPArotation documentation for details or see ff.fss.

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.

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 GPForth.df.
G	The gradient of the objective function at the rotated loadings.

Author(s)

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

References

Jennrich, R.I. (2004). Derivative free gradient projection algorithms for rotation. *Psychometrika*, **69**, 475–480.

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

cubimax.df fssQ.df fssT.df ff.bentler, ff.cf, ff.cubimax, ff.entropy, ff.fss, ff.geomin, ff.infomax, ff.oblimax, ff.pst, ff.quartimax, ff.quartimin, ff.simplimax, ff.target, ff.varimax

Examples

```
# GPRSorth and rotation name
data("Harman", package = "GPArotation")
GPForth.df(Harman8, method = "quartimax")
GPForth.df(Harman8, method = "cubimax")
GPForth.df(Harman8, method = "varimax")
GPFoblq.df(Harman8, method = "quartimin")

# displaying results of factor analysis rotation output
origdigits <- options("digits")
Abor.unrotated <- factanal(factors = 2, covmat = ability.cov, rotation = "none")
Abor <- GPFoblq.df(loadings(Abor.unrotated), method = "quartimin")
Abor
print(Abor)
print(Abor, Table = TRUE)
print(Abor, digits = 2)
summary(Abor)
options(digits = origdigits$digits)
```

`rotationsDF` *Rotations*

Description

Optimize factor loading rotation objective.

Usage

```
cubimax.df(A, Tmat=diag(ncol(A)), normalize=FALSE, eps=1e-5, maxit=1000)
fssQ.df(A, Tmat=diag(ncol(A)), kij=2, normalize=FALSE, eps=1e-5, maxit=1000)
fssT.df(A, Tmat=diag(ncol(A)), kij=2, normalize=FALSE, eps=1e-5, maxit=1000)
```

Arguments

`A`	an initial factor loadings matrix to be rotated.
`Tmat`	initial rotation matrix.
`kij`	minimum additional number of forced simple structure loadings in a pair of factors.
`normalize`	parameter passed to optimization routine (`GPForth.df` or `GPFoblq.df`).
`eps`	parameter passed to optimization routine (`GPForth.df` or `GPFoblq.df`).
`maxit`	parameter passed to optimization routine (`GPForth.df` or `GPFoblq.df`).

Details

The functions listed here optimize a rotation objective. They can be used directly or the function name can be passed to factor analysis functions like `factanal`.

Available rotations are

`cubimax.df`	orthogonal	Cubimax
`fssQ.df`	oblique	Forced Simple Structure (see Vignette)
`fssT.df`	orthogonal	Forced Simple Structure (see Vignette)

The argument `kij` for Forced Simple Structure is the minimum number of forced simple structure loadings in a pair of factors, in addition to the number of factors itself. Meaningful values are integers (1, ..., items - factors)

Value

A list (which includes elements used by `factanal`) with:

`loadings`	Lh from `GPForth.df` or `GPFoblq.df`.
`Th`	Th from `GPForth.df` or `GPFoblq.df`.
`Table`	Table from `GPForth.df` or `GPFoblq.df`.
`method`	A string indicating the rotation objective function.
`orthogonal`	A logical indicating if the rotation is orthogonal.
`convergence`	Convergence indicator from `GPForth.df` or `GPFoblq.df`.
`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 GPForth.df.

Author(s)

Coen A. Bernaards and Robert I. Jennrich

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. (2004) Derivative free gradient projection algorithms for rotation, *Psychometrika*, **69**(3), 475–480.

See Also

`GPForth.df`, `GPFoblq.df`, `ff.cubimax`, `ff.fss`, `factanal`

Examples

```
data(ability.cov)
x <- factanal(factors = 3, covmat = ability.cov, rotation="none")
fssT.df(x$loadings, kij = 2)
fssQ.df(x$loadings, kij = 4)

# 3 different methods
data("WansbeekMeijer", package="GPArotation")
fa.unrotated  <- factanal(factors = 3, covmat=NetherlandsTV, rotation="none")
#
fa.varimax <- GPForth.df(loadings(fa.unrotated), method = "varimax", normalize = TRUE)
fa.cubimax <- cubimax.df(loadings(fa.unrotated), normalize = TRUE)
fa.quartimax <- GPForth.df(loadings(fa.unrotated), method = "quartimax", normalize = TRUE)
print(cbind(loadings(fa.varimax), loadings(fa.cubimax), loadings(fa.quartimax)), digits = 2)
```

Index