Package 'GPArotation'

February 19, 2015

Version 2014.11-1

Title GPA Factor Rotation

Description Gradient Projection Algorithm Rotation for Factor Analysis. See ?GPArotation.Intro for more details.

Depends R (>= 2.0.0)

LazyLoad yes

License GPL (>= 2) | file LICENSE

Author Coen Bernaards and Robert Jennrich

Maintainer Paul Gilbert<pgilbert.ttv9z@ncf.ca>

URL http://www.stat.ucla.edu/research/gpa

NeedsCompilation no

Repository CRAN

Date/Publication 2014-11-25 08:40:16

R topics documented:

`GPArotation-package` *GPA Rotation for Factor Analysis*

Description

GPArotation implements Gradient Projection Algorithms and several rotation objective functions for factor analysis.

Details

Package:	GPArotation
Depends:	R (>= 2.0.0)
License:	GPL Version 2.
URL:	http://www.stat.ucla.edu/research or http://www.stat.ucla.edu/research/gpa

The main optimization functions are `GPForth` and `GPFoblq`. Rotation objectives include `oblimin` and many others.

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 `http://www.stat.ucla.edu/research/gpa`.

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

`rotations`, `GPForth`, `GPFoblq`, `factanal`

00.GPArotation.Intro *GPA Rotation for Factor Analysis*

Description

See `GPArotation-package` (in the help system use package?GPArotation or ?"GPArotation-package") for an overview.

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

`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 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

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

```
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.
`method`	rotation objective criterian.
`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.
`methodArgs`	a list ofmethodArgs arguments passed to the rotation objective

Details

Gradient projection rotation optimization routines developed by Coen A. Bernaards and Robert I. Jennrich. 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`. For examples of the indirect use see the documention for rotations (such as `oblimin`).

`GPForth` is the main GP algorithm for orthogonal rotation. `GPFoblq` is the main GP algorithm for oblique rotation. Both algorithms require a 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. A rotation is defined as `A %*% t(solve(Tmat))`. For the orthogonal case `Tmat` is orthonormal so this simplifies to `A %*% Tmat`. The starting point for iterative optimization is given by the `Tmat` rotation of `A`. By default the initial rotation is the identity matrix. For some rotation criteria local optima may exist and it is recommended to check for this by starting with many different initial rotations. The function `Random.Start` will help to do this.

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 objectives are `"oblimin"`, `"quartimin"`, `"target"`, `"pst"`, `"oblimax"`, `"entropy"`, `"quartimax"`, `"varimax"`, `"simplimax"`, `"bentler"`, `"tandemI"`, `"tandemII"`, `"geomin"`, `"cf"`, `"infomax"` and `"mccammon"`. The string is prefixed with "vgQ." to give the actual function call. The `vgQ.*` function call would typically not be used directly, so these methods are not exported from the package namespace. You can print these functions to see the code for calculating a criterion, but since they are not exported the package name needs to be specified. For example, use `GPArotation:::vgQ.oblimin` to view the function `vgQ.oblimin`.

Some rotation criteria (including `"simplimax"`, `"pst"`, `"procrustes"`) require one or more additional arguments. For example, `"simplimax"` needs the number of 'close to zero loadings' which is given as the extra argument `k`. Check the rotation methods for details. (If a new rotation method is implemented and needs additional arguments then this is the way to pass them.)

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.
`Th`	The rotation matrix, Lh %*% 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.
`Gq`	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

Additional information is available at http://www.stat.ucla.edu/research or http://www.stat.ucla.edu/research/gpa 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

Random.Start, factanal, oblimin, quartimin, targetT, targetQ, pstT, pstQ, oblimax, entropy, quartimax, Varimax, varimax, simplimax, bentlerT, bentlerQ, tandemI, tandemII, geominT, geominQ, cfT, cfQ, infomaxT, infomaxQ, mccammon, promax

Examples

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

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

GPForth(loadings(fa.unrotated), method="varimax", normalize=TRUE)$loadings

TV <- GPFoblq(loadings(fa.unrotated), method="oblimin", normalize=TRUE)

print(TV)
print(TV, Table=TRUE)
summary(TV)
```

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

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

Description

Random orthogonal rotation to use as Tmat matrix to start 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.

Value

An orthogonal matrix.

Author(s)

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

See Also

GPForth, GPFoblq, oblimin

Examples

```
Global.min <- function(A,method,B=10){
   fv <- rep(0,B)
   seeds <- sample(1e+7, B)
   for(i in 1:B){
  cat(i," ")
  set.seed(seeds[i])
  gpout <- GPFoblq(A=A, Random.Start(ncol(A)), method=method)
  dtab <- dim(gpout$Table)
  fv[i] <- gpout$Table[dtab[1],2]
  cat(fv[i], "\n")
   }
   cat("Min is ",min(fv),"\n")
   set.seed(seeds[order(fv)[1]])
   ans <- GPFoblq(A=A, Random.Start(ncol(A)), method=method)
```

```
    ans
    }

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

Global.min(box26,"simplimax",10)
```

rotations	*Rotations*

Description

Optimize factor loading rotation objective.

Usage

```
oblimin(L, Tmat=diag(ncol(L)), gam=0, normalize=FALSE, eps=1e-5, maxit=1000)
quartimin(L, Tmat=diag(ncol(L)), normalize=FALSE, eps=1e-5, maxit=1000)
targetT(L, Tmat=diag(ncol(L)), Target=NULL, normalize=FALSE, eps=1e-5, maxit=1000)
targetQ(L, Tmat=diag(ncol(L)), Target=NULL, normalize=FALSE, eps=1e-5, maxit=1000)
pstT(L, Tmat=diag(ncol(L)), W=NULL, Target=NULL,
           normalize=FALSE, eps=1e-5, maxit=1000)
pstQ(L, Tmat=diag(ncol(L)), W=NULL, Target=NULL,
           normalize=FALSE, eps=1e-5, maxit=1000)
oblimax(L, Tmat=diag(ncol(L)), normalize=FALSE, eps=1e-5, maxit=1000)
entropy(L, Tmat=diag(ncol(L)), normalize=FALSE, eps=1e-5, maxit=1000)
quartimax(L, Tmat=diag(ncol(L)), normalize=FALSE, eps=1e-5, maxit=1000)
Varimax(L, Tmat=diag(ncol(L)), normalize=FALSE, eps=1e-5, maxit=1000)
simplimax(L, Tmat=diag(ncol(L)), k=nrow(L),
           normalize=FALSE, eps=1e-5, maxit=1000)
bentlerT(L, Tmat=diag(ncol(L)), normalize=FALSE, eps=1e-5, maxit=1000)
bentlerQ(L, Tmat=diag(ncol(L)), normalize=FALSE, eps=1e-5, maxit=1000)
tandemI(L, Tmat=diag(ncol(L)), normalize=FALSE, eps=1e-5, maxit=1000)
tandemII(L, Tmat=diag(ncol(L)), normalize=FALSE, eps=1e-5, maxit=1000)
geominT(L, Tmat=diag(ncol(L)), delta=.01,
           normalize=FALSE, eps=1e-5, maxit=1000)
geominQ(L, Tmat=diag(ncol(L)), delta=.01,
           normalize=FALSE, eps=1e-5, maxit=1000)
cfT(L, Tmat=diag(ncol(L)), kappa=0, normalize=FALSE, eps=1e-5, maxit=1000)
cfQ(L, Tmat=diag(ncol(L)), kappa=0, normalize=FALSE, eps=1e-5, maxit=1000)
infomaxT(L, Tmat=diag(ncol(L)), normalize=FALSE, eps=1e-5, maxit=1000)
infomaxQ(L, Tmat=diag(ncol(L)), normalize=FALSE, eps=1e-5, maxit=1000)
mccammon(L, Tmat=diag(ncol(L)), normalize=FALSE, eps=1e-5, maxit=1000)
bifactorT(L, Tmat=diag(ncol(L)), normalize=FALSE, eps=1e-5, maxit=1000)
bifactorQ(L, Tmat=diag(ncol(L)), normalize=FALSE, eps=1e-5, maxit=1000)

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.cf(L, kappa=0)
vgQ.infomax(L)
vgQ.mccammon(L)
vgQ.bifactor(L)
```

Arguments

`L`	a factor loading matrix
`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 L\^2 in 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).

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 `GPForth` or `GPFoblq` respectively.

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.

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 Criterion
tandemII	orthogonal	Tandem Criterion
geominT	orthogonal	
geominQ	oblique	
cfT	orthogonal	Crawford-Ferguson family
cfQ	oblique	Crawford-Ferguson family
infomaxT	orthogonal	
infomaxQ	oblique	
mccammon	orthogonal	McCammon minimum entropy ratio
bifactorT	orthogonal	Jennrich and Bentler bifactor rotation
bifactorQ	oblique	Jennrich and Bentler biquartimin rotation

Also included for convenience are two analytic rotations `eiv` and `echelon` which do not require `GPForth` or `GPFoblq`.

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.

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`	the value of the criterion at L.
`Gq`	the gradient at L.
`Method`	a string indicating the criterion.

Value

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

`loadings`	Lh from `GPForth` or `GPFoblq`.
`Th`	Th from `GPForth` or `GPFoblq`.

`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 `GPForth` or `GPFoblq`.
`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.

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.

Bifactor rotation, bifactorT and bifactorQ are called bifactor and biquartimin in Jennrich, R.I. and Bentler, P.M. (2011) Exploratory bi-factor analysis. *Psychometrika*, **76**.

A discussion of rotation objectives can be found in many references, for example,

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

See Also

`GPForth`, `GPFoblq`, `WansbeekMeijer`, `eiv`, `echelon`, `factanal`, `varimax`, `promax`

Examples

```
data(ability.cov)
factanal(factors = 2, covmat = ability.cov, rotation="oblimin")

data("Harman", package="GPArotation")
qHarman  <- GPForth(Harman8, Tmat=diag(2), method="quartimax")
qHarman2 <- quartimax(Harman8)

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

fa.varimax <- factanal(factors = 2, covmat=NetherlandsTV,
              rotation="varimax", control=list(rotate=list(normalize=TRUE)))
fa.oblimin <- factanal(factors = 2, covmat=NetherlandsTV,
              rotation="oblimin", control=list(rotate=list(normalize=TRUE)))

cbind(loadings(fa.unrotated), loadings(fa.varimax), loadings(fa.oblimin))
```

`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`

`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