Package ‘tsfa’

May 11, 2007

Title Time Series Factor Analysis

Description Extraction of Factors from Multivariate Time Series. See ?00tsfa-Intro for more details.

Depends R (>= 2.1.0), GPArotation (>= 2006.9-1), setRNG (>= 2004.4-1), tframe (>= 2007.5-1), dse1 (>= 2006.1-1), dse2 (>= 2006.1-1)

Suggests CDNmoney

Version 2007.05-1

Date 2007-05-07

LazyLoad yes

License GPL Version 2. See the LICENSE file for details.

Author Paul Gilbert and Erik Meijer <pgilbert@bank-banque-canada.ca>

Maintainer Paul Gilbert and Erik Meijer <pgilbert@bank-banque-canada.ca>

URL http://www.bank-banque-canada.ca/pgilbert

R topics documented:

`00.tsfa.Intro` *Time Series Factor Analysis (TSFA)*

Description

TSFA extends standard factor analysis (FA) to time series data. Rotations methods can be applied as in FA. A dynamic model of the factors is not assumed, but could be estimated separately using the extracted factors.

Details

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

`checkResiduals.TSFmodel`
Check Time Series Idiosyncratic Component

Description

The data is subtracted from the explained data (after differencing if `diff` is `TRUE`, the default) and the result is treated as a residual. Its covariance, the sum of the diagonal elements of the covariance, and the sum of the off-diagonal elements of the covariance are printed. The residual is then passed to the default method for `checkResiduals` which produces several diagonistic plots and (invisibly) returns statistics. See `checkResiduals` for more details. Calculation of partial autocorrelations can be problematic.

Some care should be taken interpreting the results. Factor estimation does not minimize residuals, it extracts common factors.

Usage

```
## S3 method for class 'TSFmodel':
checkResiduals(obj, data=obj$data, diff.=TRUE, ...)
```

Arguments

`obj`	TSFmodel object for which the idiosyncratic component should be examined (as if it were a residual).
`data`	data from which the idiosyncratic component should be calculated.
`diff.`	logical indicating if data and explained should be differenced.
`...`	arguments to be passed to checkResiduals default methods.

Author(s)

Paul Gilbert

See Also

checkResiduals, TSFmodel, estTSF.ML

Examples

```
data("CanadianMoneyData.asof.28Jan2005", package="CDNmoney")
data("CanadianCreditData.asof.28Jan2005", package="CDNmoney")

z <- tframed(tbind(
    MB2001,
    MB486 + MB452 + MB453 ,
    NonbankCheq,
    MB472 + MB473 + MB487p,
    MB475,
    NonbankNonCheq + MB454 + NonbankTerm + MB2046 + MB2047 + MB2048 +
    MB2057 + MB2058 + MB482),
    names=c("currency", "personal cheq.", "NonbankCheq",
    "N-P demand & notice", "N-P term", "Investment" )
  )

z <- tfwindow(tbind (z, ConsumerCredit, ResidentialMortgage,
                        ShortTermBusinessCredit, OtherBusinessCredit),
     start=c(1981,11), end=c(2004,11))

cpi <- 100 * M1total / M1real
popm <- M1total / M1PerCapita
scale <- tfwindow(1e8 /(popm * cpi), tf=tframe(z))

MBandCredit <- sweep(z, 1, scale, "*")
c4withML  <- estTSF.ML(MBandCredit, 4)

checkResiduals(c4withML, pac=FALSE)
```

---

distribution.factorsEstEval

Distribution of Time Series Factors Estimates

Description

Plot the distribution of the multiple estimates from EstEval, and possibly multiple EstEval objects.

Usage

```
## S3 method for class 'factorsEstEval':
distribution(obj, ..., bandwidth = "nrd0",
    cumulate=TRUE, graphs.per.page = 5, Title=NULL)
```

Arguments

`obj`	EstEval object.
`bandwidth`	bandwidth for distribution smoothing.
`cumulate`	logical indicating if the distribution across time and repititions should be plotted (TRUE) or a time series of standard deviation across repititions should be plotted (FALSE).
`graphs.per.page`	number of graphs on an output page.
`Title`	string indicating a title for the plot.
`...`	additional EstEval objects which will be plotted on the same graph.

Author(s)

Paul Gilbert

See Also

`distribution`, `EstEval`, `estTSF.ML`

Examples

```
data("CanadianMoneyData.asof.6Feb2004", package="CDNmoney")

### Construct data

cpi <- 100 * M1total / M1real
seriesNames(cpi) <- "CPI"
popm <- M1total / M1PerCapita
seriesNames(popm) <- "Population of Canada"

z <- tframed(tbind(
    MB2001,
    MB486 + MB452 + MB453 ,
    NonbankCheq,
    MB472 + MB473 + MB487p,
    MB475,
    NonbankNonCheq + MB454 + NonbankTerm + MB2046 + MB2047 + MB2048 +
    MB2057 + MB2058 + MB482),
    names=c("currency", "personal cheq.", "NonbankCheq",
    "N-P demand & notice", "N-P term", "Investment")
    )

z <- tfwindow(z, start=c(1986,1))
if( all(c(2003,12) ==end(z))) z <-tfwindow(z, end=c(2003,11))
MBcomponents <- 1e8 * z/matrix(tfwindow(popm * cpi,tf=tframe(z)),periods(z),6)

### Specify "true" parameters and factors

Omega <- diag(c(72.63, 1233, 87.33,
            629.4, 3968, 12163))

Boblq <- t(matrix(c(
   8.84,   5.20,
  23.82, -12.57,
```

```
     5.18,  -1.97,
    36.78,  16.94,
    -2.84,  31.02,
     2.60,  47.63), 2,6))

PhiOblq <- matrix(c( 1.0, 0.00949, 0.00949, 1.0),2,2)

etaBart <- MBcomponents %*% solve(Omega) %*% Boblq %*% (
            solve( t(Boblq) %*% solve(Omega) %*% Boblq ) )

DetaBart <- diff(etaBart, lag=1)
SDE      <- cov(DetaBart)
RR1 <- chol(SDE)      # upper triangular: SDE = RR1' RR1
RR2 <- chol(PhiOblq)  # ditto
PP  <- t(RR2) %*% solve(t(RR1))
Psi       <- 0.5 * Omega

etaTrue <- tframed(etaBart %*% t(PP), tf=tframe(MBcomponents))

### run Monte Carlo  N.B. replications would typically be much larger

require("dse2")

EE.ML5 <- EstEval(TSFmodel(Boblq, f=etaTrue, positive.measures=FALSE),
  replications=5, quiet=FALSE,
  simulation.args=list(Cov=Psi, noIC=TRUE),
  estimation="estTSF.ML", estimation.args=list(2, BpermuteTarget=Boblq),
  criterion ="TSFmodel")

distribution(factors(EE.ML5))
distribution(factors(EE.ML5), cumulate=FALSE)
distribution(diff(factors(EE.ML5)))
distribution(diff(factors(EE.ML5)), cumulate=FALSE)
```

`estFAmodel` *Estimate a Factor Model*

Description

Estimate an `FAmodel`.

Usage

```
estFAmodel(Sigma, p, n.obs=NA,
            est="factanal",
            estArgs=list(scores="none", control=list(opt=list(maxit=10000))),
            rotation=if(p==1) "none" else "quartimin", rotationArgs=NULL,
            GPFargs=list(Tmat=diag(p), normalize=TRUE, eps=1e-5, maxit=1000),
            BpermuteTarget=NULL,
            factorNames=paste("Factor", seq(p)),
            indicatorNames=NULL)
```

Arguments

`Sigma`	covariance of the data matrix.
`n.obs`	integer indication number of observations in the dataset.
`p`	integer indication number of factors to estimate.
`est`	name of the estimation function.
`estArgs`	list of aarguments passed to the estimation function.
`rotation`	character vector indicating the factor rotation method (see **GPArotation** for many options).
`rotationArgs`	list of arguments passed to the rotation method, specifying arguments for the rotation criteria. See `GPFoblq`.
`GPFargs`	list of arguments passed to `GPFoblq` or `GPForth` for rotation optimization
`BpermuteTarget`	matrix of loadings. If supplied, this is used to permute the order of estimated factors and change signs. (It is for comparison with other results.
`factorNames`	vector of strings indicating names of factor series.
`indicatorNames`	vector of strings indicating names of indicator series.

Details

The default `est` method and `quartimin` rotation give parameters using standard (quasi) ML factor analysis (on the correlation matrix and then scaled back). The function `factanal` with no rotation is used to find the initial (orthogonal) solution. Rotation is then done (by default with quartimin using `GPFoblq` optimization). `factanal` always uses the correlation matrix, so standardizing does not affect the solution.

If `rotation` is `"none"` the result of the `factanal` estimation is not rotated. In this case, to avoid confusion with a rotated solution, the factor covariance matrix `Phi` is returned as `NULL`. Another possibility for its value would be the identity matrix, but this is not calculated so `NULL` avoids confusion.

The arguments `rotation`, `rotationArgs` are used for rotation. The quartimin default uses GPArotation and its default `normalize=TRUE`, `eps=1e-5`, `maxit=1000`, and `Tmat=I` are passed through the rotation method to `GPFoblq`.

The estimated loadings, Bartlett predictor matrix, etc., are put in the returned `FAmodel` (see below). The Bartlett factor score coefficient matrix can be calculated as

$$(B'\Omega^{-1}B)^{-1}B'\Omega^{-1}x$$

or equivalently as

$$(B'\Sigma^{-1}B)^{-1}B'\Sigma^{-1}x,$$

The first is simpler because Ω is diagonal, but breaks down with a Heywood case, because Ω is then singular (one or more of its diagonal elements are zero). The second only requires nonsingularity of Σ. Typically, Σ is not singular even if Ω is singular. Σ is calculated from $B\Phi B' + \Omega$, where B, Φ, and Ω are the estimated values returned from `factanal` and rotated. The data covariance could also be used for Σ. (It returns the same result with this estimation method.)

The returned `FAmodel` object is a list containing

loadings the estimated loadings matrix.

Omega the covariance of the idiosyncratic component (residuals).

Phi the covariance of the factors.

LB the Bartlett predictor matrix.

LB.std the standardized Bartlett predictor matrix.

estConverged a logical indicating if estimation converged.

rotationConverged a logical indicating if rotation converged.

orthogonal a logical indicating if the rotation is orthogonal.

uniquenesses the uniquenesses.

call thearguments of the function call.

Value

A `FAmodel` object (see details).

Author(s)

Paul Gilbert and Erik Meijer

References

Gilbert, Paul D. and Meijer, Erik (2005) Time Series Factor Analaysis with an Application to Measuring Money. Research Report 05F10, University of Groningen, SOM Research School. Available from `http://som.eldoc.ub.rug.nl/reports/themeF/2005/05F10/`.

See Also

`estTSF.ML`, `rotations`, `factanal`

Examples

```
data("WansbeekMeijer", package="GPArotation")
fa.unrotated  <-  estFAmodel(NetherlandsTV, 2, n.obs=2150, rotation="none" )
fa.varimax <- estFAmodel(NetherlandsTV, 2, n.obs=2150, rotation="Varimax" )
fa.eiv     <- estFAmodel(NetherlandsTV, 2, n.obs=2150, rotation="eiv" )
fa.oblimin <- estFAmodel(NetherlandsTV, 2, n.obs=2150, rotation="oblimin" )

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

`estTSFmodel` *Estimate Time Series Factor Model*

Description

Estimate a `TSFmodel`.

Usage

```
estTSFmodel(y, p, diff.=TRUE,
            est="factanal",
            estArgs=list(scores="none", control=list(opt=list(maxit=10000))),
            rotation=if(p==1) "none" else "quartimin",
            rotationArgs=NULL,
            GPFargs=list(Tmat=diag(p),normalize=TRUE, eps=1e-5, maxit=1000),
            BpermuteTarget=NULL,
            factorNames=paste("Factor", seq(p)))
estTSF.ML(y, p, diff.=TRUE,
            rotation=if(p==1) "none" else "quartimin",
            rotationArgs=NULL,
            normalize=TRUE, eps=1e-5, maxit=1000, Tmat=diag(p),
            BpermuteTarget=NULL,
            factorNames=paste("Factor", seq(p)))
```

Arguments

`y`	a time series matrix.
`p`	integer indication number of factors to estimate.
`diff.`	logical indicating if model should be estimated with differenced data.
`est`	character vector indicating the factor estimation method (currently only factanal is supported).
`estArgs`	list passed to as arguments to the estimation function.
`rotation`	character vector indicating the factor rotation method (see **GPArotation** for options).
`rotationArgs`	list passed to the rotation method, specifying arguments for the rotation criteria.
`GPFargs`	list passed to `GPFoblq` or `GPForth`, possibly via the rotation method, specifying arguments for the rotation optimization. See `GPFoblq` and `GPForth`.
`normalize`	Passed to `GPFoblq`. `TRUE` means do Kaiser normalization before rotation and then undo it after completing rotatation. `FALSE` means do no normalization. See `GPFoblq` for other possibilities.
`eps`	passed to `GPFoblq`
`maxit`	passed to `GPFoblq`
`Tmat`	passed to `GPFoblq`
`BpermuteTarget`	matrix of loadings. If supplied, this is used to permute the order of estimated factors and change signs in order to compare properly.
`factorNames`	vector of strings indicating names to be given to factor series.

Details

The function `estTSF.ML` is a wrapper to `estTSFmodel`.

The function `estTSF.ML` estimates parameters using standard (quasi) ML factor analysis (on the correlation matrix and then scaled back). The function `factanal` with no rotation is used to find the initial (orthogonal) solution. Rotation, if specified, is then done with `GPFoblq`. `factanal` always uses the correlation matrix, so standardizing does not affect the solution.

If `diff.` is `TRUE` (the default) the indicator data is differenced before it is passed to `factanal`. This is necessary if the data is not stationary. The resulting Bartlett factor score coefficient matrix (rotated) is applied to the undifferenced data. See *Gilbert and Meijer (2005)* for a discussion of this approach.

If `rotation` is `"none"` the result of the `factanal` estimation is not rotated. In this case, to avoid confusion with a rotated solution, the factor covariance matrix `Phi` is returned as `NULL`. Another possibility for its value would be the identity matrix, but this is not calculated so `NULL` avoids confusion.

The arguments `rotation`, `methodArgs`, `normalize`, `eps`, `maxit`, and `Tmat` are passed to `GPFoblq`.

The estimated loadings, Bartlett factor score coefficient matrix and predicted factor scores are put in a `TSFmodel` which is part of the returned object. The Bartlett factor score coefficient matrix can be calculated as

$$(B'\Omega^{-1}B)^{-1}B'\Omega^{-1}x$$

or equivalently as

$$(B'\Sigma^{-1}B)^{-1}B'\Sigma^{-1}x,$$

The first is simpler because Ω is diagonal, but breaks down with a Heywood case, because Ω is then singular (one or more of its diagonal elements are zero). The second only requires nonsingularity of Σ. Typically, Σ is not singular even if Ω is singular. Σ is calculated from $B\Phi B' + \Omega$, where B, Φ, and Ω are the estimated values returned from `factanal` and rotated. The data covariance could also be used for Σ. (It returns the same result with this estimation method.)

The returned `TSFestModel` object is a list containing

model the estimated `TSFmodel`.

data the indicator data used in the estimation.

estimates a list of

- **estimation** a character string indicating the name of the estimation function.
- **diff.** the setting of the argument `diff`.
- **rotation** the setting of the argument `rotation`.
- **uniquenesses** the estimated uniquenesses.
- **BpermuteTarget** the setting of the argument `BpermuteTarget`.

Value

A `TSFestModel` object which is a list containing `TSFmodel`, the data, and some information about the estimation.

Author(s)

Paul Gilbert and Erik Meijer

References

Gilbert, Paul D. and Meijer, Erik (2005) Time Series Factor Analaysis with an Application to Measuring Money. Research Report 05F10, University of Groningen, SOM Research School. Available from `http://som.eldoc.ub.rug.nl/reports/themeF/2005/05F10/`.

See Also

TSFmodel, GPFoblq, rotations, factanal

Examples

```
data("CanadianMoneyData.asof.28Jan2005", package="CDNmoney")
data("CanadianCreditData.asof.28Jan2005", package="CDNmoney")

z <- tframed(tbind(
    MB2001,
    MB486 + MB452 + MB453 ,
    NonbankCheq,
    MB472 + MB473 + MB487p,
    MB475,
    NonbankNonCheq + MB454 + NonbankTerm + MB2046 + MB2047 + MB2048 +
    MB2057 + MB2058 + MB482),
    names=c("currency", "personal cheq.", "NonbankCheq",
    "N-P demand & notice", "N-P term", "Investment" )
  )

z <- tfwindow(tbind (z, ConsumerCredit, ResidentialMortgage,
                        ShortTermBusinessCredit, OtherBusinessCredit),
     start=c(1981,11), end=c(2004,11))

cpi <- 100 * M1total / M1real
popm <- M1total / M1PerCapita
scale <- tfwindow(1e8 /(popm * cpi), tf=tframe(z))

MBandCredit <- sweep(z, 1, scale, "*")
c4withML  <- estTSF.ML(MBandCredit, 4)
tfplot(ytoypc(factors(c4withML)),
       Title="Factors from 4 factor model (year-to-year growth rate)")
tfplot(c4withML, graphs.per.page=3)
summary(c4withML)
summary(TSFmodel(c4withML))
```

---

explained — *Calculate Explained Portion of Data*

Description

Calculate portion of the data (indicators) explained by the factors.

Usage

```
    explained(object, ...)
    ## S3 method for class 'TSFmodel':
    explained(object, f=factors(object),
                  names=seriesNames(object), ...)
    ## S3 method for class 'FAmodel':
    explained(object, f=factors(object),
                  names=indicatorNames(object), ...)
```

Arguments

`object`	A TSFmodel or TSFestModel.
`f`	Factor values to use with the model.
`names`	A vector of strings to use for the output series.
`...`	arguments passed to other methods.

Value

A time series matrix.

Author(s)

Paul Gilbert

See Also

`TSFmodel`, `predict`, `estTSF.ML`, `simulate`, `tfplot.TSFmodel`,

`factorNames` *Extract the Factors Names from an Object*

Description

Extract the factor (or series) names from an object.

Usage

```
factorNames(x)
## S3 method for class 'FAmodel':
factorNames(x)
## S3 method for class 'TSFfactors':
factorNames(x)
## S3 method for class 'EstEval':
factorNames(x)
## S3 method for class 'TSFmodel':
seriesNames(x)
```

Arguments

`x`	an object.

Value

character vector of names.

Author(s)

Paul Gilbert

See Also

`factors`, `nfactors`, `seriesNames`, `TSFmodel`,

factors — *Extract Time Series Factors from an Object*

Description

Extract time series factors from an object.

Usage

```
factors(x)
## S3 method for class 'TSFmodel':
factors(x)
## S3 method for class 'EstEval':
factors(x)
```

Arguments

x	an object.

Value

factor series.

Author(s)

Paul Gilbert

See Also

TSFmodel, estTSF.ML, simulate.TSFmodel

FAfitStats — *Summary Statistics for a TSFA Models*

Description

FAfitStats calculates various statistics for a TSFestModel or all possible (unrotated factanal) models for a data matrix. This function is also used by the summary method for a TSFestModel.

Usage

```
FAfitStats(object, ...)
## Default S3 method:
FAfitStats(object, diff. = TRUE,
           N=(nrow(object) - diff.),
           control=list(lower = 0.0001, opt=list(maxit=1000)), ...)
## S3 method for class 'TSFmodel':
FAfitStats(object, diff. = TRUE,
           N=(nrow(object$data) - diff.), ...)
```

Arguments

`object`	a time series matrix or TSFestModel.
`diff.`	logical indicating if data should be differenced.
`N`	sample size.
`control`	a list of arguments passed to `factanal`.
`...`	further arguments passed to other methods.

Details

In the case of the method for a `TSFmodel` the model parameters are extracted from the model and the result is a vector of various fit statistics (see below). (Calculations are done by the internal function `FAmodelFitStats`.)

Most of these statistics are described in *Wansbeek and Meijer* (2000, WM below). The sample size N is used in the calculation of these statistics. The default is the number of number of observations, as in WM. That is, the number of rows in the data matrix, minus one if the data is differenced. Many authors use $N-1$, which would be $N-2$ if the data is differenced. The exact calculations can be determined by examining the code: `print(tsfa:::FAmodelFitStats)`. The vector of statistics is:

chisq Chi-square statistic (see, for example, WM p298).

df degrees of freedom, which takes the rotational freedom into account (WM p169).

pval p-value

delta delta

RMSEA Root mean square error of approximation (WM p309).

RNI Relative noncentrality index (WM p307).

CFI Comparative fit index (WM p307).

MCI McDonald's centrality index.

GFI Goodness of fit index (Jöreskog and Sörbom, 1981, 1986, WM p305).

AGFI Adjusted GFI (Jöreskog and Sörbom, 1981, 1986).

AIC Akaike's information criterion (WM p309).

CAIC Consistent AIC(WM p310).

SIC Schwarz's Bayesian information criterion.

CAK Cudeck & Browne's rescaled AIC.

CK Cudeck & Browne's cross-validation index.

The information criteria account for rotational freedom. Some of these goodness of fit statistics should be used with caution, because they are not yet based on sound statistical theory. Future versions of tsfa will probably provide improved versions of these goodness-of-fit statistics.

In the case of the default method, which expects a matrix of data with columns for each indicator series, models are calculated with `factanal` for factors up to the Ledermann bound. No rotation is needed, since rotation does not affect the fit statistics. Values for the saturated model are also appended to facilitate a sequential comparison.

If `factanal` does not obtain a satisfactory solution it may produce an error "unable to optimize from these starting value(s)." This can sometimes be fixed by increasing the `opt, maxit` value in the `control` list.

The result for the default method is a list with elements

fitStats a matrix with rows as for a single model above, and a column for each possible number of factors.

seqfitStats a matrix with rows `chisq`, `df`, and `pval`, and columns indicating the comparative fit for an additional factor starting with the null (zero factor) model. (See also independence model, WM, p305)

The largest model can correspond to the saturated model, but will not if the Ledermann bound is not an integer, or even in the case of an integer bound but implicit contraints resulting in a Heywood case (see Dijkstra, 1992). In these situations it might make sense to remove the model corresponding to the largest integer, and make the last sequential comparison between the second to largest integer and the saturated solution. The code does not do this automatically.

Value

a vector or list of various fit statistics. See details.

Author(s)

Paul Gilbert and Erik Meijer

References

Dijkstra, T. K. (1992) On Statistical Inference with Parameter Estimates on the Boundary of the Parameter Space, *British Journal of Mathematical and Statistical Psychology*, **45**, 289–309.

Hu, L.-t., and Bentler, P. (1995) Evaluating model fit. In R. H. Hoyle (Ed.), *Structural equation modeling: Concepts, issues, and applications* (pp. 76–99). Thousand Oaks, CA: Sage.

Jöreskog, K. G., and Sörbom, D. (1981) *LISREL V user's guide*. Chicago: National Educational Resources.

Jöreskog, K. G., and Sörbom, D. (1986) LISREL VI: Analysis of linear structural relationships by maximum likelihood, instrumental variables, and least squares methods (User's Guide, 4th ed.). Mooresville, IN: Scientific Software.

Ogasawara, Haruhiko. (2001). Approximations to the Distributions of Fit Indexes for Misspecified Structural Equation Models. *Structural Equation Modeling*, **8**, 556–574.

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

See Also

`FAmodelFitStats`, `summary`, `summary.TSFmodel`, `summaryStats`, `LedermannBound`

Examples

```
data("CanadianMoneyData.asof.28Jan2005", package="CDNmoney")
data("CanadianCreditData.asof.28Jan2005", package="CDNmoney")

z <- tframed(tbind(
    MB2001,
    MB486 + MB452 + MB453 ,
    NonbankCheq,
    MB472 + MB473 + MB487p,
    MB475,
    NonbankNonCheq + MB454 + NonbankTerm + MB2046 + MB2047 + MB2048 +
    MB2057 + MB2058 + MB482),
```

```
    names=c("currency", "personal cheq.", "NonbankCheq",
    "N-P demand & notice", "N-P term", "Investment" )
  )

z <- tfwindow(tbind (z, ConsumerCredit, ResidentialMortgage,
                ShortTermBusinessCredit, OtherBusinessCredit),
     start=c(1981,11), end=c(2004,11))

cpi <- 100 * M1total / M1real
popm <- M1total / M1PerCapita
scale <- tfwindow(1e8 /(popm * cpi), tf=tframe(z))

MBandCredit <- sweep(z, 1, scale, "*")

FAfitStats(MBandCredit)

c4withML  <- estTSF.ML(MBandCredit, 4)
FAfitStats(c4withML)
```

FAmodelFitStats *Calculate Summary Statistics with given FA Model Parameters*

Description

Calculates various statistics with given Paramaters of an FA Model.

Usage

```
FAmodelFitStats(B, Phi, omega, S, N)
```

Arguments

B	loadings.
Phi	cov. matrix of factors.
omega	vector of error variances
S	sample covariance matrix.
N	sample size.

Details

This function is used by FAfitStats and would not normally be called by a user.

Value

a vector of various fit statistics.

Author(s)

Paul Gilbert and Erik Meijer

See Also

FAfitStats

FAmodel *Construct a Factor Model*

Description

The default method constructs a FAmodel. Other methods extract a FAmodel from an object.

Usage

```
 FAmodel(obj, ...)
 ## Default S3 method:
 FAmodel(obj,  Omega=NULL, Phi=NULL, LB=NULL, LB.std=NULL,
stats=NULL,  ...)
 ## S3 method for class 'FAmodel':
 FAmodel(obj, ...)
```

Arguments

`obj`	The loadings matrix (B) in the default (constructor) method. In other methods, an object from which the model should be extracted.
`Omega`	Covariance of the idiosyncratic term.
`Phi`	Covariance of the factors.
`LB`	Factor score predictor matrix.
`LB.std`	The standardized factor score predictor matrix.
`stats`	An optional list of statistics from model estimation.
`...`	arguments passed to other methods or stored in the object.

Details

The default method is the constructor for `FAmodel` objects. Other methods extract a `FAmodel` object from other objects that contain one. The loadings must be supplied to the default method. Omega, Phi, and LB are included when the object comes from an estimation method, but are not necessary when the object is being specified in order to simulate. The model is defined by

$$y_t = Bf_t + \varepsilon_t,$$

where the factors f_t have covariance Φ and ε_t have covariance Ω. The loadings matrix B is $M \times k$, where M is the number of indicator variables (the number of indicators in y) and k is the number of factors.

Value

A FAmodel.

Author(s)

Paul Gilbert

See Also

`TSFmodel`, `estFAmodel`

Examples

```
B <- t(matrix(c(0.9, 0.1,
                0.8, 0.2,
                0.7, 0.3,
                0.5, 0.5,
                0.3, 0.7,
                0.1, 0.9), 2,6))

z <- FAmodel(B)
z
loadings(z)
```

`LedermannBound` *Ledermann Bound for Number of Indicators*

Description

The Ledermann bound is given by the solution k for $(M - k)^2 \geq M + k$, where M is the number of indicator variables. The maximum possible number of factors is the largest integer smaller than or equal k.

Usage

```
LedermannBound(M)
```

Arguments

`M`	an integer indicating the number of indicator variables or a matrix of data, in which case ncol(M) is used as the number of indicator variables.

Value

The Ledermann bound, a positive real number.

Author(s)

Paul Gilbert and Erik Meijer

References

Tom Wansbeek and Erik Meijer (2000) *Measurement Error and Latent Variables in Econometrics*, Amsterdam: North-Holland. (note p169.)

See Also

`FAfitStats`

`loadings` *Extract the Loadings Matrix from an Object*

Description

Extract the loadings matrix from an object.

Usage

```
loadings(x)
## Default S3 method:
loadings(x)
## S3 method for class 'FAmodel':
loadings(x)
DstandardizedLoadings(x)
## S3 method for class 'TSFmodel':
DstandardizedLoadings(x)
```

Arguments

`x`	an object.

Details

`stats:::loadings` is defined as the default method for the generic which replaces it. (See `help(loadings, package="stats")` for more details.) The loadings matrix in `TSFmodel` and `TSFestModel` objects is similar to that described for the default, but calculated for a TSFA model. More details are provided in `estTSF.ML`

Value

a loadings matrix.

Author(s)

Paul Gilbert

See Also

`stats:::loadings`, `factors`, `factorNames`, `estTSF.ML`, `TSFmodel`,

nfactors — *Extract the Number of Factors from an Object*

Description

Extract the number of factors from an object.

Usage

```
nfactors(x)
## S3 method for class 'FAmodel':
nfactors(x)
## S3 method for class 'TSFfactors':
nfactors(x)
## S3 method for class 'EstEval':
nfactors(x)
```

Arguments

`x`	an object.

Value

an integer.

Author(s)

Paul Gilbert

See Also

`factors`, `factorNames`, `TSFmodel`,

permusign — *Internal Utility to Permute the Loadings Matrix.*

Description

Internal utility to permute the loadings matrix.

Usage

```
permusign(B, Btarget, Phi=NULL)
```

Arguments

`B`	proposed loadings matrix.
`Btarget`	target loadings matrix.
`Phi`	proposed Phi matrix.

Value

list with a permuted and sign changed loadings matrix and the corresponding `Phi` matrix.

Author(s)

Paul Gilbert and Erik Meijer

See Also

`factors`, `factorNames`, `TSFmodel`,

---

predict — *Predict Factor Scores from an Object.*

Description

Predict factor scores using the predictor from object.

Usage

```
## S3 method for class 'FAmodel':
predict(object,
       data = NULL, factorNames.=factorNames(object), ...)
## S3 method for class 'TSFmodel':
predict(object,
       data = object$data, factorNames.=factorNames(object), ...)
```

Arguments

`object`	an object from which a matrix (predictor) can be extracted to apply to the data.
`data`	data to which the predictor should be applied.
`factorNames.`	names to be given to the calculated predicted factor scores.
`...`	additional arguments currently unused.

Details

If `data` is not supplied then it is extacted from object if possible (which is normally the data the model was estimated with), and otherwise an error is indicated. The predicted factor scores are given by `data %*% t(LB)`, where LB is the factor score predictor matrix extracted from object. This is the Barlett factor score coefficient matrix if `TSFmodel` or `TSFestModel` objects were estimated with `estTSF.ML`.

Value

Predicted factor scores.

Author(s)

Paul Gilbert

See Also

`predict`, `factors`, `factorNames`, `TSFmodel`

simulate.TSFmodel *Simulate a Time Series Factor Model*

Description

Simulate a TSFmodel to generate time series data (indicators) using factors and loadings from the model.

Usage

```
## S3 method for class 'TSFmodel':
simulate(model, f=factors(model), Cov=model$Omega,
   sd=NULL, noise=NULL, rng=NULL, noise.model=NULL, ...)
```

Arguments

`model`	A TSFmodel.
`f`	Factors to use with the model.
`Cov`	covariance of the idiosyncratic term.
`sd`	see `makeTSnoise`.
`noise`	see `makeTSnoise`.
`rng`	see `makeTSnoise`.
`noise.model`	see `makeTSnoise`.
`...`	arguments passed to other methods.

Details

`simulate.TSFmodel` generates artifical data (indicators or measures) with a given `TSFmodel` (which has factors and loadings). The `obj` should be a `TSFmodel`. This might be a model constructed with `TSFmodel` or as returned by `estTSF.ML`.

The number of factor series is determined by the number of columns in the time series matrix `f` (the factors in the model object). This must also be the number of columns in the loadings matrix B (in the model object). The number of rows in the loadings matrix determines the number of indicator series generated (the number of columns in the matrix result). The number of rows in the time series factor matrix determines the number of periods in the indicator series generated (the number of rows in the matrix result).

`simulate` passes `Cov`, `sd`, `noise`, `rng`, and `noise.model` to `makeTSnoise` to generate the random idiosyncratic term ε_t, which will have the same dimension as the generated indicator series that are returned. ε_t will have random distribution determined by other arguments passed to `makeTSnoise`. Note that the covariance of the generated indicator series y_t is also influenced by the covariance of the factors f.

The calculation to give the generated artificial time series indicator data matrix y is

$$y_t = Bf_t + \varepsilon_t.$$

`simulate.TSFmodel` can use a `TSFmodel` that has only `B` and `f` specified, but in this case one of `Cov`, `sd`, `noise`, or `noise.model` must be specified as the default `Omega` from the model is not available.

Value

A time series matrix.

Author(s)

Paul Gilbert

See Also

TSFmodel, estTSF.ML, simulate, tfplot.TSFmodel, explained.TSFmodel

Examples

```
f <- matrix(c(2+sin(pi/100:1),5+3*sin(2*pi/5*(100:1))),100,2)
B <- t(matrix(c(0.9, 0.1,
                0.8, 0.2,
                0.7, 0.3,
                0.5, 0.5,
                0.3, 0.7,
                0.1, 0.9), 2,6))

z <- simulate(TSFmodel(B, f=f), sd=0.01)
tfplot(z)
```

---

summary.TSFmodel *summary.TSFmodel Method for Base Generic*

Description

Summary method for object in **tsfa**, such as the object returned by the estimation method `estTSF.ML`. See `FAfitStats` for details on the results from `summary.TSFmodel`.

Usage

```
## S3 method for class 'TSFmodel':
summary(object, ...)
## S3 method for class 'FAmodel':
summary(object, ...)
## S3 method for class 'TSFmodelEstEval':
summary(object, ...)
## S3 method for class 'summary.TSFmodel':
print(x, ...)
## S3 method for class 'summary.FAmodel':
print(x, ...)
## S3 method for class 'summary.TSFmodelEstEval':
print(x, digits = options()$digits, ...)
```

Arguments

`object`	an object to summarize.
`x`	an object to print.
`digits`	precision of printed numbers.
`...`	further arguments passed to other methods.

Value

a summary object.

Author(s)

Paul Gilbert and Erik Meijer

See Also

`estTSF.ML`, `FAfitStats`, `summary`

`summaryStats` *Summary Statistics Calculations*

Description

Calculates various statistics from a TSFmodelEstEval object returned by EstEval. This function is for use by the summary and tfplot methods and would not typically be called by a user.

Usage

```
summaryStats(object, ...)
## S3 method for class 'TSFmodelEstEval':
summaryStats(object, ...)
```

Arguments

`object`	a TSFestModel object to summarize.
`...`	further arguments passed to other methods.

Value

a list passed of statistics.

Author(s)

Paul Gilbert and Erik Meijer

See Also

`EstEval`, `summary.TSFmodelEstEval`, `tfplot.TSFmodelEstEval` tframeMethods *Time Series Factor Methods for tframe Generics*

Description

Plot or difference objects. See the generic descriptions.

Usage

```
## S3 method for class 'TSFmodel':
tframe(x)

## S3 method for class 'TSFmodel':
tfplot(x, ..., tf=tfspan(x, ...), start=tfstart(tf), end=tfend(tf),
              series = seq(nfactors(x)),
              Title = "Model factors",
              lty = 1:5, lwd = 1, pch = NULL, col = 1:6, cex = NULL,
              xlab = NULL, ylab = factorNames(x), xlim = NULL, ylim = NULL,
              graphs.per.page = 5,
              par=NULL, mar = par()$mar, reset.screen = TRUE)
## S3 method for class 'TSFfactors':
tfplot(x,..., tf=tfspan(x, ...), start=tfstart(tf), end=tfend(tf),
              series=seq(nfactors(x)),
              Title="Estimated factors (dashed) and true (solid)",
              lty = c("dashed", "solid"), lwd = 1, pch = NULL, col = 1:6, cex
              xlab=NULL, ylab=factorNames(x), xlim = NULL, ylim = NULL,
              graphs.per.page=5, par=NULL, mar=par()$mar, reset.screen=TRUE)
## S3 method for class 'TSFexplained':
tfplot(x,..., tf=tfspan(x, ...), start=tfstart(tf), end=tfend(tf),
              series=seq(nseries(x)),
              Title="Explained (dashed) and actual data (solid)",
              lty = c("dashed", "solid"), lwd = 1, pch = NULL, col = 1:6, cex
              xlab=NULL,
              ylab=seriesNames(x),
              xlim = NULL, ylim = NULL,
              graphs.per.page=5, par=NULL, mar=par()$mar, reset.screen=TRUE)
## S3 method for class 'TSFmodelEstEval':
tfplot(x, ...,  tf=NULL, start=tfstart(tf), end=tfend(tf),
              series=seq(nseries(factors(x))),
              Title="Monte Carlo Results",
              lty = c("solid", "dotdash", "dashed",  "dashed"), lwd = 1, pch
              col = c("black", "red", "red", "red"), cex = NULL,
              xlab=NULL,
              ylab=seriesNames(factors(x$truth)),
              xlim = NULL, ylim = NULL,
              graphs.per.page=5, par=NULL, mar=par()$mar, reset.screen=TRUE,
              diff.=FALSE,  percentChange.=FALSE,
              PCcentered.=FALSE, summary.=TRUE)

 ## S3 method for class 'TSFmodel':
 diff(x, ...)
```

```
## S3 method for class 'TSFexplained':
diff(x, ...)
## S3 method for class 'TSFfactors':
diff(x, ...)
## S3 method for class 'factorsEstEval':
diff(x, ...)
```

Arguments

`x`	an object.
`x`	a TSFmodel, TSFestModel, TSFexplained, or TSFfactors object for plotting or differencing.
`diff.`	logical indicating if differenced data should be plotted.
`percentChange.`	logical indicating if percent change data should be plotted.
`PCcentered.`	logical indicating if centered percent change data should be plotted.
`summary.`	logical indicating if mean and 1 SD bounds should be plotted in place of all estimates.
`tf`	See generic tfplot method
`start`	See generic tfplot method
`end`	See generic tfplot method
`series`	See generic tfplot method
`Title`	string to use for title of factors plot (but see details).
`lty`	See generic tfplot method
`lwd`	See generic tfplot method
`pch`	See generic tfplot method
`col`	See generic tfplot method
`cex`	See generic tfplot method
`xlab`	See generic tfplot method
`ylab`	See generic tfplot method
`xlim`	See generic tfplot method
`ylim`	See generic tfplot method
`graphs.per.page`	See generic tfplot method
`par`	See generic tfplot method
`mar`	See generic tfplot method
`reset.screen`	See generic tfplot method
`...`	other objects to plot (currently unused).

Details

The `Title` is not put on the plot if the global option PlotTitles is FALSE. This can be set with `options(PlotTitles=FALSE)`. This provides a convenient mechanism to omit all titles when the title may be added separately (e.g. in Latex).

Value

diff returns an object in which the time series data has been differenced. tfplot returns an invisible value but is executed mainly for the side-effect (plot).

Author(s)

Paul Gilbert

See Also

TSFmodel, estTSF.ML, simulate.TSFmodel, tfplot, diff, factors, explained, factorNames, TSFmodel

---

tsfa-package *Time Series Factor Analysis (TSFA)*

Description

TSFA extends standard factor analysis (FA) to time series data. Rotations methods can be applied as in FA. A dynamic model of the factors is not assumed, but could be estimated separately using the extracted factors.

Details

Package:	tsfa
Depends:	R (>= 2.0.0), GPArotation, setRNG (>= 2004.4-1), tframe (>= 2006.1-1), dse1 (>= 2006.1-1), dse2 (>= 2006.1-1)
Suggests:	CDNmoney
License:	GPL Version 2.
URL:	http://www.bank-banque-canada.ca/pgilbert

The main functions are:

```
DstandardizedLoadings   Extract standardized loadings from an object
loadings                Extractloadings from an object
estTSF.ML               Estimate a time series factor model
factors                 Extract time series factors from an object
FAmodelFitStats         Various fit statistics.
simulate                Simulate a time series factor model
summary                 Summary methods for \pkg{tsfa} objects
tfplot                  Plot methods for \pkg{tsfa} objects
TSFmodel                Construct a time series factor model
```

An overview of how to use the package is available in the vignette `tsfa` (source, pdf).

Author(s)

Paul Gilbert <pgilbert@bank-banque-canada.ca> and Erik Meijer <e.meijer@eco.rug.nl>

Maintainer: Paul Gilbert <pgilbert@bank-banque-canada.ca>

References

Gilbert, Paul D. and Meijer, Erik (2005) Time Series Factor Analaysis with an Application to Measuring Money. Research Report 05F10, University of Groningen, SOM Research School. Available from http://som.eldoc.ub.rug.nl/reports/themeF/2005/05F10/.

Gilbert, Paul D. and Meijer, Erik (2006) Money and Credit Factors. Bank of Canada Working Paper 2006-3, Available from http://www.bank-banque-canada.ca/en/res/wp/wp(y)_2006.html.

See Also

estTSF.ML, GPArotation, tframe, dse1, dse2

TSFmodel *Construct a Time Series Factor Model*

Description

The default method constructs a TSFmodel. Other methods extract a TSFmodel from an object.

Usage

```
TSFmodel(obj, ...)
## Default S3 method:
TSFmodel(obj, f=NULL, Omega = NULL, Phi=NULL, LB = NULL,
     positive.data=FALSE, names=NULL, ...)
## S3 method for class 'TSFmodel':
TSFmodel(obj, ...)
## S3 method for class 'FAmodel':
TSFmodel(obj, f=NULL, positive.data=FALSE, names=NULL, ...)
```

Arguments

obj	The loadings matrix (B) in the default (constructor) method. In other methods, an object from which the model should be extracted.
f	matrix of factor series.
Omega	Covariance of the idiosyncratic term.
Phi	Covariance of the factors.
LB	Factor score coefficient matrix.
positive.data	logical indicating if any resulting negative values should be set to zero.
names	vector of strings indicating names to be given to output series.
...	arguments passed to other methods or stored in the object.

Details

The default method is the constructor for `TSFmodel` objects. Other methods extract a `TSFmodel` object from other objects that contain one. The loadings and the factors must be supplied to the default method. Omega, Phi, and LB are included when the object comes from an estimation method, but are not necessary when the object is being specified in order to simulate. The model is defined by

$$y_t = Bf_t + \varepsilon_t,$$

where the factors f_t have covariance Φ and ε_t have covariance Ω. The loadings matrix B is $M \times k$, where M is the number of indicator variables (the number of series in y) and k is the number of factor series.

The estimation method `estTSF.ML` returns a `TSFmodel` as part of a `TSFestModel` that has additional information about the estimation.

Value

A TSFmodel.

Author(s)

Paul Gilbert

See Also

`simulate.TSFmodel`, `simulate`, `estTSF.ML`

Examples

```
f <- matrix(c(2+sin(pi/100:1),5+3*sin(2*pi/5*(100:1))),100,2)
B <- t(matrix(c(0.9, 0.1,
                0.8, 0.2,
                0.7, 0.3,
                0.5, 0.5,
                0.3, 0.7,
                0.1, 0.9), 2,6))

z <- TSFmodel(B, f=f)
tfplot(z)
```

Index