Package ‘tsfa’

February 20, 2015

Version 2014.2-2
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), dse (>= 2006.1-1), EvalEst (>= 2006.1-1)
Imports setRNG (>= 2004.4-1), tframe (>= 2011.3-1), tfplot (>= 2014.2-1)
Suggests CDNmoney, MASS
LazyLoad yes
License GPL-2
Author Paul Gilbert and Erik Meijer <pgilbert.ttv9z@ncf.ca>
Maintainer Paul Gilbert<pgilbert.ttv9z@ncf.ca>
URL http:OOtsanalysisNrMforgeNrMprojectOorg/
Repository CRAN
Repository/R-Forge/Project tsanalysis
Repository/R-Forge/Revision 161
Repository/R-Forge/DateTimeStamp 2014-02-19 15:01:27
Date/Publication 2014-02-24 16:41:41
NeedsCompilation no

R topics documented:

  tsfa-package .................................................. 2
  00.tsfa.Intro .................................................. 3
  checkResiduals.TSFmodel ...................................... 3
  distribution.factorsEstEval .................................. 5
  estFModel .................................................... 7
  estTSFmodel .................................................. 9
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
Suggests: CDNmoney
License: GPL Version 2.
URL: http://tsanalysis.r-forge.r-project.org/

The main functions are:

- extract standardized loadings from an object
- Extract loadings from an object
- Estimate a time series factor model
- Extract time series factors from an object
- Various fit statistics.
- Simulate a time series factor model
- Summary methods for \pkg{tsfa} objects
- Plot methods for \pkg{tsfa} objects
- Construct a time series factor model

An overview of how to use the package is available in the vignette \pkg{tsfa} (source, pdf).
Author(s)
Paul Gilbert <pgilbert.ttv9z@ncf.ca> and Erik Meijer <meijer@rand.org>
Maintainer: Paul Gilbert <pgilbert.ttv9z@ncf.ca>

References

See Also
estTSF.ML, GPArotation, tframe, dse

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.

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 diagnostic 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

```r
## 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

```r
if (require("CDNmoney")){
  data("CanadianMoneyData.asof.28Jan2005", package="CDNmoney")
  data("CanadianCreditData.asof.28Jan2005", package="CDNmoney")

  z <- tframed(tbind(
    MB2001,
    MB462 + 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

```r
## 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 repetitions should be plotted (TRUE) or a time series of standard deviation across repetitions 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

```r
if (require("CDNmoney")){
  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(
```
distribution.factorsEstEval


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=tfframe(z)), Tobs(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 <- tfframe(etabart %*% t(PP), tf=tfframe(MBcomponents))

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

require("EvalEst")

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))
---

**estFAmo**

Estimate a Factor Model

---

**Description**

Estimate an FAmo model.

**Usage**

```r
estFAmo(sigma, p, n.obs = NA, est = "factanal",
estArgs = list(scores = "none", control = list(opt = list(maxit = 1000))),
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 arguments 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 `GPFob1q`.
- `GPFargs` : list of arguments passed to `GPFob1q` 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 FAmode1 (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 \(\Omega\) is diagonal, but breaks down with a Heywood case, because \(\Omega\) is then singular (one or more of its diagonal elements are zero). The second only requires nonsingularity of \(\Sigma\). Typically, \(\Sigma\) is not singular even if \(\Omega\) is singular. \(\Sigma\) is calculated from \(B\Phi B' + \Omega\), where \(B, \Phi,\) and \(\Omega\) are the estimated values returned from factanal and rotated. The data covariance could also be used for \(\Sigma\). (It returns the same result with this estimation method.)

The returned FAmode1 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** the arguments of the function call.

Value

A FAmode1 object (see details).
estTSFmodel

Author(s)

Paul Gilbert and Erik Meijer

References


See Also

estTSF.ML, rotations, factanal

Examples

```r
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.eiv))
```

estTSFmodel

**Estimate Time Series Factor Model**

Description

Estimate a TSFmodel.

Usage

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

The returned \texttt{TSFestModel} object is a list containing

- \texttt{model} the estimated \texttt{TSFmodel}.
- \texttt{data} the indicator data used in the estimation.
- \texttt{estimates} a list of
  - \texttt{estimation} a character string indicating the name of the estimation function.
  - \texttt{diff} the setting of the argument \texttt{diff}.
  - \texttt{rotation} the setting of the argument \texttt{rotation}.
  - \texttt{uniquenesses} the estimated uniquenesses.
  - \texttt{BpermuteTarget} the setting of the argument \texttt{BpermuteTarget}.

\textbf{Value}

A \texttt{TSFestModel} object which is a list containing \texttt{TSFmodel}, the data, and some information about the estimation.

\textbf{Author(s)}

Paul Gilbert and Erik Meijer

\textbf{References}


\textbf{See Also}

\texttt{TSFmodel, GPFoblq, rotations, factanal}

\textbf{Examples}

```r
if (require("CDNmoney")){
data("CanadianMoneyData.asof.28Jan2005", package="CDNmoney")
data("CanadianCreditData.asof.28Jan2005", package="CDNmoney")

z <- tframed(tbind(MB2001, MB486 + MB452 + MB453)
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=dimnames(loadings(object))[[1]], ...)  

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.
factorNames

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**

```r
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.
- **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


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

$$y_t = B f_t + \varepsilon_t,$$

where the factors $f_t$ have covariance $\Phi$ and $\varepsilon_t$ have covariance $\Omega$. 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

```r
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))
```
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 to $k$.

**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**


**See Also**

FAfitStats
nfparsors

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.

predict

Predict Factor Scores from an Object.

Description

Predict factor scores using the predictor from object.
Usage

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

```r
# 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 extracted 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.EM`.

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

```r
## 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

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 time observations (periods) in the indicator series generated, that is, 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 $\varepsilon_t$, which will have the same dimension as the generated indicator series that are returned. $\varepsilon_t$ 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

$$y_t = B f_t + \varepsilon_t.$$  

`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 $\Omega_m$ 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

```r
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,
               0.7, 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)
```

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`

---

### TSFmodel

**Construct a Time Series Factor Model**

**Description**

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

**Usage**

```r
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

\[ y_t = B f_t + \varepsilon_t, \]

where the factors \( f_t \) have covariance \( \Phi \) and \( \varepsilon_t \) have covariance \( \Omega \). 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
```r
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

*Topic package
  ØØ.tsfa.Intro, 3
  tsfa-package, 2

*Topic ts
  checkResiduals.TSFmodel, 3
  distribution.factorsEstEval, 5
  estFAModel, 7
  estTSFModel, 9
  explained, 12
  factorNames, 13
  factors, 14
  FAFitStats, 14
  FAModel, 17
  LedermannBound, 19
  nfactors, 20
  predict, 20
  simulate.TSFModel, 21
  summary.TSFModel, 23
  TSFModel, 24
  ØØ.tsfa.Intro, 3
  checkResiduals, 3, 4
  checkResiduals.TSFModel, 3
  distribution, 5
  distribution.factorsEstEval, 5
  dse, 3
  EstEval, 5
  estFAModel, 7, 18
  estTSF.ML, 3–5, 9, 13, 14, 22–25
  estTSF.ML(estTSFModel), 9
  estTSFModel, 9
  explained, 12
  explained.TSFModel, 23
  factanal, 9, 11
  factorNames, 13, 20, 21
  factors, 13, 14, 20, 21

  FAfitStats, 14, 19, 23, 24
  FAModel, 17
  FAModelFitStats, 17
  GPArotation, 3
  GPFoblq, 8, 10, 11
  LedermannBound, 17, 19
  makeTSnoise, 22
  nfactors, 13, 20
  predict, 13, 20, 21
  print.summary.FAModel
    (summary.TSFModel), 23
  print.summary.TSFModel
    (summary.TSFModel), 23
  print.summary.TSFModelEstEval
    (summary.TSFModel), 23
  rotations, 9, 11
  seriesNames, 13
  seriesNames.TSFModel(factorNames), 13
  simulate, 13, 23, 25
  simulate.TSFModel, 14, 21, 25
  summary, 17, 24
  summary.FAModel (summary.TSFModel), 23
  summary.TSFModel, 17, 23
  summary.TSFModelEstEval
    (summary.TSFModel), 23
  summaryStats, 17
  tfplot.TSFModel, 13, 23
  tframe, 3
  tsfa-package, 2
  tsfa.Intro (tsfa-package), 2
  TSFModel, 4, 10, 11, 13, 14, 18, 20–23, 24