Package 'tensorTS'

April 15, 2021

Type Package

Title Factor Models and Autoregressive Models for Time Series

Version 0.1.0

Author Zebang Li and Ruofan Yu and Yuefeng Han and Han Xiao and Rong Chen and Dan Yang

Maintainer Zebang Li <zl326@stat.rutgers.edu>

Description Factor and Autoregressive Models for Tensor Time Series

License GPL (>= 2)

Encoding UTF-8

LazyData true

Depends tensor,
rTensor,
MASS,
expm,
abind

RoxygenNote 7.1.1

VignetteBuilder knitr

R topics documented:

`matAR.RR.est` *Reduced Rank MAR(1) Iterative Estimation for Matrix-Valued Time Series*

Description

Reduced rank one-term MAR(1) iterative estimation for matrix-valued time series, including the reduced rank one-term MAR(1) iterative estimation (RRLSE), and reduced rank one-term MAR(1) iterative estimation with Kronecker structured covariance (RRMLE), as determined by the value of `method`.

Usage

```
matAR.RR.est(xx, method=c("RRLSE","RRMLE"), A1.init=NULL, A2.init=NULL,Sigl.init=NULL,
Sigr.init=NULL,k1=NULL, k2=NULL, niter=100,tol=1e-6)
```

Arguments

`xx`	$T \times d_1 \times d_2$ matrix-valued time series, T is the length of the series.
`method`	character string, specifying the method of the estimation to be used.
	`"RRLSE"`, Least squares.
	`"RRMLE"`, MLE under a separable cov(vec(E_t)).
`Sigl.init`	only if `method=RRMLE`, initial value of Sigl. The default is the identity matrix.
`Sigr.init`	only if `method=RRMLE`, initial value of Sigr. The default is the identity matrix.
`k1`	rank of A_1, a positive integer.
`k2`	rank of A_2, a positive integer.
`niter`	maximum number of iterations if error stays above `tol`.
`tol`	relative Frobenius norm error tolerance.
`A1.init`	initial value of A_1. The default is the identity matrix.
`A2.init`	initial value of A_2. The default is the identity matrix.

Details

One-term matrix autoregressive model (order one) has the following form:

$$X_t = A_1 X_{t-1} A_2' + E_t,$$

where A_i are $d_i \times d_i$ coefficient matrices, $i = 1, 2$, and E_t is a matrix white noise. Note that only the lag-1 term X_{t-1} appears on the right hand side, so we refer to it as an order-1 model. We also consider a special form of Cov(vec(E_t)),

$$\mathrm{Cov}(\mathrm{vec}(E_t)) = \Sigma_r \otimes \Sigma_l$$

For matrix reduced-rank model, we assume that $\mathrm{rank}(A_i) = k_i \leq d_i$ for $i = 1, 2$.

Value

return a list containing the following:

`A1` estimator of A_1, a d_1 by d_1 matrix

`A2` estimator of A_2, a d_2 by d_2 matrix

`Sigl` only if `method=MLE`, when $\mathrm{Cov}(\mathrm{vec}(E_t)) = \Sigma_r \otimes \Sigma_l$.

`Sigr` only if `method=MLE`, when $\mathrm{Cov}(\mathrm{vec}(E_t)) = \Sigma_r \otimes \Sigma_l$.

`res` residuals.

`Sig` covariance matrix cov(vec(E_t))

`cov` covariance matrix $\hat{A}_1$ and $\hat{A}_2$. If `method=RRLSE` or `method=RRMLE`, then it is a list containing

`Sigma` asymptotic covariance matrix of (vec($\hat{A}_1$),vec($\hat{A}_2^T$)).

`Theta1.u`, `Theta1.v` asymptotic covariance matrix of vec($\hat{U}_1$), vec($\hat{V}_1$).

`Theta2.u`, `Theta2.v` asymptotic covariance matrix of vec($\hat{U}_2$), vec($\hat{V}_2$).

`sd` standard errors of $\hat{A}_1$ and $\hat{A}_2$, returned in a list aligned with $\hat{A}_1$ and $\hat{A}_2$.

`niter` number of iterations.

`BIC` value of the extended Bayesian information criterion.

References

Reduced Rank Autoregressive Models for Matrix Time Series, by Han Xiao, Yuefeng Han, Rong Chen and Chengcheng Liu.

Examples

```
dim <- c(3,3)
xx <- tenAR.sim(t=500, dim, R=2, P=1, rho=0.5, cov='iid')
est <- matAR.RR.est(xx, method="RRLSE", k1=1, k2=1)
```

`matAR1.RR.se` *Asymptotic Covariance Matrix of One-Term Reduced rank MAR(1) Model*

Description

Asymptotic covariance matrix of one-term reduced rank MAR(1) model. If `Sigma1` and `Sigma2` is provided in input, we assume a separable covariance matrix, Cov(vec(E_t)) = $\Sigma_2 \otimes \Sigma_1$.

Usage

```
matAR1.RR.se(A1,A2,k1,k2,Sigma.e=NULL,Sigma1=NULL,Sigma2=NULL,RU1=diag(k1),
RV1=diag(k1),RU2=diag(k2),RV2=diag(k2),mpower=100)
```

Arguments

A1	left coefficient matrix
A2	right coefficient matrix
k1	rank of A1
k2	rank of A2
Sigma.e	Sigma.e = Cov(vec(E_t)): covariance matrix of dimension $(d_1 d_2) \times (d_1 d_2)$
Sigma1, Sigma2	Cov(vec(E_t)) = $\Sigma_2 \otimes \Sigma_1$. If these parameters provided, we assume a separable covariance matrix, Cov(vec(E_t)) = $\Sigma_2 \otimes \Sigma_1$.
RU1, RV1, RU2, RV2:	orthogonal rotations of U_1, V_1, U_2, V_2, e.g., new_U1=U1 RU1
mpower	truncate the VMA(∞) representation of vec(X_t) at mpower for the purpose of calculating the autocovariances. The default is 100.

Value

a list containing the following:

Sigma asymptotic covariance matrix of (vec($\hat{A}_1$),vec($\hat{A}_2^T$)).

Theta1.u asymptotic covariance matrix of vec($\hat{U}_1$).

Theta1.v asymptotic covariance matrix of vec($\hat{V}_1$).

Theta2.u asymptotic covariance matrix of vec($\hat{U}_2$).

Theta2.v asymptotic covariance matrix of vec($\hat{V}_2$).

References

Han Xiao, Yuefeng Han, Rong Chen and Chengcheng Liu, Reduced Rank Autoregressive Models for Matrix Time Series.

---

mplot *Plot Matrix-Valued Time Series*

Description

Plot matrix-valued time series, can be also used to plot tensor-valued time series by given mode.

Usage

```
mplot(x)
```

Arguments

xx	$T \times d_1 \times d_2$ matrix-valued time series. Note that the number of mode is 3, where the first mode is time.

Examples

```
dim <- c(3,3,3)
xx <- tenAR.sim(t=500, dim, R=2, P=1, rho=0.5, cov='iid')
mplot(xx[,,1])
```

tenAR.est *Estimation for Autoregressive Model of Tensor-Valued Time Series*

Description

Estimation function for tensor-valued time series. Projection method (PROJ), the Iterated Least Squares method (LSE), MLE under a Kronecker structured covariance matrix (MLE) and stacked vector AR(1) model (VAR), as determined by the value of `method`.

Usage

```
tenAR.est(xx, method=c("PROJ","LSE","MLE","VAR"), R=1, P=1, init.A=NULL,init.sig=NULL,
niter=500, tol=1e-6, print.true=FALSE)
```

Arguments

`xx`	$T \times d_1 \times \cdots \times d_K$ tensor-valued time series, T is the length of the series.
`R`	Kronecker rank for each lag.
`P`	Autoregressive order.
`method`	character string, specifying the type of the estimation method to be used. `"PROJ"`, Projection method. `"LSE"`, Least squares. `"MLE"`, MLE under a separable cov(vec(E_t)). `"VAR"`, VAR(1) model for the vectorized tensor series.
`init.A`	initial values of coefficient matrices $A_1^{(ir)}, \cdots, A_K^{(PR)}$ in estimation algorithms, which is a multi-layer list such that the first level denotes orders, the second denotes terms, the third denotes modes. See our example code in the following. By default, we use projection estimators as initial values.
`init.sig`	only if `method=MLE`, a list of initial values of $\Sigma_1, \ldots, \Sigma_K$. The default are identity matrices.
`niter`	maximum number of iterations if error stays above `tol`.
`tol`	error tolerance in terms of the Frobenius norm.

Details

Tensor autoregressive model (order one) has the following form:

$$X_t = \sum_{r=1}^{R} X_{t-1} \times_1 A_1^{(r)} \times_2 \cdots \times_K A_K^{(r)} + E_t,$$

where $A_k^{(r)}$ are $d_k \times d_k$ coefficient matrices, $k = 1, \cdots, K$, and E_t is a tensor white noise. R is the number of terms. Note that only the lag-1 term X_{t-1} appears on the right hand side, so we refer to it as an order-1 model. For order-p model we have the form:

$$X_t = \sum_{i=1}^{P} \sum_{r=1}^{R_i} X_{t-i} \times_1 A_1^{(ir)} \times_2 \cdots \times_K A_K^{(ir)} + E_t.$$

We also consider a special form of $\mathrm{Cov}(\mathrm{vec}(E_t))$,

$$\mathrm{Cov}(\mathrm{vec}(E_t)) = \Sigma_K \otimes \Sigma_{K-1} \otimes \cdots \otimes \Sigma_1,$$

Value

return a list containing the following:

`A` a list of estimated coefficient matrices $A_1^{(ir)}, \cdots, A_K^{(PR)}$. It is a multi-layer list, the first level denotes orders, the second denotes terms, the third denotes modes. See our example code in the following.

`sigma` only if `method=MLE`, a list of estimated $\Sigma_1, \ldots, \Sigma_K$.

`res` residuals

`Sig` covariance matrix cov(vec(E_t)).

`cov` grand covariance matrix of all estimated entries of $A_1^{(ir)}, \cdots, A_K^{(PR)}$

`sd` standard errors of the coefficient matrices $A_1^{(ir)}, \cdots, A_K^{(PR)}$, returned as a list aligned with `A`.

`niter` number of iterations.

`BIC` value of extended Bayesian information criterion.

References

Rong Chen, Han Xiao, and Dan Yang. "Autoregressive models for matrix-valued time series". Journal of Econometrics, 2020.

Examples

```
dim <- c(2,2,2)
xx <- tenAR.sim(t=500, dim,R=2,P=1,rho=0.5, cov='iid')
est <- tenAR.est(xx, R=2, P=1, method="LSE")
A <- est$A # A is a multi-layer list

length(A) == 1 # TRUE, since the order P = 1
length(A[[1]]) == 2 # TRUE, since the number of terms R = 2
length(A[[1]][[1]]) == 3 # TRUE, since the mode K = 3
```

`tenAR.predict` *Predictions for Tensor Autoregressive Models*

Description

'tenAR.predict' is a function for predictions from the results of model fitting functions. The function invokes particular methods which depend on the class of the first argument. If `rolling=TRUE`, this would be a rolling forecast, specifically, using given model parameters object, start from the last time point X_t in data, we obtain the one step ahead prediction X_{t+1}. Then fit the corresponding model using all available data and X_{t+1} to obtain X_{t+2}. Repeat this process to $X_{t+n.head}$. Our function is similar to the usage of classical 'predict.ar' in package "stats".

Usage

```
tenAR.predict(object, xx, n.head, method, rolling=FALSE)
```

Arguments

object	a fit from TenAR(P) model
n.head	number of steps ahead at which to predict
method	method used by rolling forecast
rolling	TRUE or FALSE, rolling forecast, is FALSE by default
data	data to which to apply the prediction

Value

predicted value

See Also

'predict.ar' or 'predict.arima'

Examples

```
dim <- c(2,2,2)
xx <- tenAR.sim(t=500, dim,R=2,P=1,rho=0.5, cov='iid')
est <- tenAR.est(xx, R=1, P=1, method="LSE")
pred <- tenAR.predict(est, xx, n.head = 5)
# rolling forcast
pred.rolling <- tenAR.predict(est, xx, n.head = 5, method="LSE", rolling=TRUE)
```

---

tenAR.sim *Generate TenAR(p) tensor time series*

Description

Simulate from the TenAR(p) model.

Usage

```
tenAR.sim(t, dim, R, P, rho, cov)
```

Arguments

t	length of output series. A strictly positive integer.
dim	dimension of the tensor at each time.
R	Kronecker rank for each lag.
P	autoregressive order.
rho	spectral radius of coefficient matrix Φ.
cov	covariance matrix of the error term: diagonal ("iid"), separable ("mle"), random ("svd").

Value

A tensor-valued time series generated by TenAR(p) model.

See Also

tenFM.sim

Examples

```
dim <- c(3,3,3)
xx <- tenAR.sim(t=500, dim, R=2, P=1, rho=0.5, cov='iid')
```

---

tenFM.est *Estimation for Tucker structure Factor Models of Tensor-Valued Time Series*

Description

Estimation function for Tucker structure factor models of tensor-valued time series. Two unfolding methods of the auto-covariance tensor, Time series Outer-Product Unfolding Procedure (TOPUP), Time series Inner-Product Unfolding Procedure (TIPUP), are included, as determined by the value of method.

Usage

```
tenFM.est(x,r,h0=1,method='TIPUP',iter=TRUE,vmax=FALSE,tol=1e-4,maxiter=100)
```

Arguments

x	$T \times d_1 \times \cdots \times d_K$ tensor-valued time series.
r	input rank of the factor tensor.
h0	the number of lags used in auto-covariance tensor.
method	character string, specifying the type of the estimation method to be used. "TIPUP", TIPUP method. "TOPUP", TOPUP method.
iter	boolean, specifying using an iterative approach or an non-iterative approach.
vmax	boolean, specifying using varimax rotation on the factor matrix or not.
tol	tolerance in terms of the Frobenius norm.
maxiter	maximum number of iterations if error stays above tol.
dim	dimension of the coefficient matrices.

Details

Tensor factor model with Tucker structure has the following form,

$$X_t = F_t \times_1 A_1 \times_2 \cdots \times_K A_k + E_t,$$

where A_k is the deterministic loading matrix of size $d_k \times r_k$ and $r_k \ll d_k$, the core tensor F_t itself is a latent tensor factor process of dimension $r_1 \times \cdots \times r_K$, and the idiosyncratic noise tensor E_t is uncorrelated (white) across time. Two estimation approaches, named TOPUP and TIPUP, are studied. Time series Outer-Product Unfolding Procedure (TOPUP) are based on

$$\mathrm{TOPUP}_k(X_{1:T}) = \left(\sum_{t=h+1}^{T} \frac{\mathrm{mat}_k(X_{t-h}) \otimes \mathrm{mat}_k(X_t)}{T-h},\ h = 1, ..., h_0 \right),$$

where h_0 is a predetermined positive integer, $\otimes$ is tensor product. Note that $\mathrm{TOPUP}_k(\cdot)$ is a function mapping a tensor time series to a order-5 tensor. Time series Inner-Product Unfolding Procedure (TIPUP) replaces the tensor product in TOPUP with the inner product:

$$\mathrm{TIPUP}_k(X_{1:T}) = \mathrm{mat}_1\left(\sum_{t=h+1}^{T} \frac{\mathrm{mat}_k(X_{t-h})\mathrm{mat}_k^\top(X_t)}{T-h},\ h = 1, ..., h_0\right).$$

Value

return a list containing the following:

`Ft` estimated factor processes of dimension $T \times r_1 \times r_2 \times \cdots \times r_k$.

`Ft.all` Summation of factor processes over time, of dimension $r_1, r_2, \cdots, r_k$.

`Q` a list of estimated factor loading matrices $Q_1, Q_2, \cdots, Q_K$.

`x.hat` fitted signal tensor, of dimension $T \times d_1 \times d_2 \times \cdots \times d_k$.

`niter` number of iterations.

`fnorm.resid` Frobenius norm of residuals, divide the Frobenius norm of the original tensor.

References

Chen, Rong, Dan Yang, and Cun-Hui Zhang. "Factor models for high-dimensional tensor time series." Journal of the American Statistical Association just-accepted (2021): 1-59.

Han, Yuefeng, Rong Chen, Dan Yang, and Cun-Hui Zhang. "Tensor factor model estimation by iterative projection." arXiv preprint arXiv:2006.02611 (2020).

Examples

```
dims <- c(16,18,20) # dimensions of tensor time series
t <- 100
r <- c(2,2,2)  # dimensions of factor series
ar1.coef <- array(seq(0.5,0.8,length.out=prod(r)),r)
F.dims <- dim(ar1.coef)
Ft <- array(NA,dim=c(t,F.dims[1],F.dims[2],F.dims[3]))
for(ir1 in 1:F.dims[1]){
 for(ir2 in 1:F.dims[2]){
   for(ir3 in 1:F.dims[3]){
     Ft[,ir1,ir2,ir3] <- arima.sim(n=t, model=list(ar=ar1.coef[ir1,ir2,ir3]))
   }
 }
}
lambda <- sqrt(prod(dims))
x <- tenFM.sim(Ft,dims=dims,lambda=lambda,A=NULL,cov='iid') # generate t*dims tensor time series
result <- tenFM.est(x,r,method='TOPUP')  # Estimation
Ft <- result$Ft
```

tenFM.rank *Rank Determination for Tensor Factor Models with Tucker Structure*

Description

Function for rank determination of tensor factor models with Tucker Structure, Two unfolding methods of the auto-covariance tensor, Time series Outer-Product Unfolding Procedure (TOPUP), Time series Inner-Product Unfolding Procedure (TIPUP), are included, as determined by the value of `method`. Different penalty functions for the information criterion (BIC) and the eigen ratio criterion (ER) can be used, which should be specified by the value of `rank` and `penalty`. The information criterion resembles BIC in the vector factor model literature. And the eigen ratio criterion is similar to the eigenvalue ratio based methods in the vector factor model literature.

Usage

```
tenFM.rank(x,r,h0=1,rank='BIC',method='TIPUP',inputr=FALSE,iter=TRUE,penalty=1,delta1=0,tol=1e-4
```

Arguments

`x`	$T \times d_1 \times \cdots \times d_K$ tensor-valued time series.
`r`	input rank of the factor tensor.
`h0`	the number of lags used in auto-covariance tensor.
`rank`	character string, specifying the type of the rank determination method to be used. `"BIC"`, information criterion. `"ER"`, eigen ratio criterion.
`method`	character string, specifying the type of the factor estimation method to be used. `"TIPUP"`, TIPUP method. `"TOPUP"`, TOPUP method.
`inputr`	boolean, if TRUE, use input rank for each iteration; if FLASE, update the rank r in each iteration.
`iter`	boolean, specifying using an iterative approach or an non-iterative approach.
`penalty`	takes value in 1,2,3,4,5, decide which penalty function to use for each tesnor mode k. When `rank`= 'BIC': if `penalty`=1, $g_1 = \frac{h_0 d^{2-2\nu}}{T} \log(\frac{dT}{d+T})$; if `penalty`=2, $g_2 = h_0 d^{2-2\nu}(\frac{1}{T} + \frac{1}{d}) \log(\frac{dT}{d+T})$;, if `penalty`=3, $g_3 = \frac{h_0 d^{2-2\nu}}{T} \log(\min(d, T))$; if `penalty`=4, $g_4 = h_0 d^{2-2\nu}(\frac{1}{T} + \frac{1}{d}) \log(\min(d, T))$; if `penalty`=5, $g_5 = h_0 d^{2-2\nu}(\frac{1}{T} + \frac{1}{d}) \log(\min(d_k, T))$. When `rank`= 'ER': if `penalty`=1, $h_1 = c_0 h_0$; if `penalty`=2, $h_2 = \frac{h_0 d^2}{T^2}$; if `penalty`=3, $h_3 = \frac{h_0 d^2}{T^2 d_k^2}$; if `penalty`=4, $h_4 = \frac{h_0 d^2}{T^2 d_k^2} + \frac{h_0 d_k^2}{T^2}$;

if penalty=5, $h_5 = \frac{h_0 d^2}{T^2 d_k^2} + \frac{h_0 d d_k^2}{T^2}$.

delta1	weakest factor strength, a tuning parameter used for BIC method only, default value is 0.
tol	tolerance in terms of the Frobenius norm.
maxiter	maximum number of iterations if error stays above tol.

Value

return a list containing the following:

path a $K \times (\text{niter} + 1)$ matrix of the estimated Tucker rank of the factor process as a path of the maximum number of iteration (niter) used. The i-th column is the estimated rank $\hat{r}_1, \hat{r}_2, \cdots, \hat{r}_K$ at $(i-1)thiteration$.

factor.num final solution of the estimated Tucker rank of the factor process $\hat{r}_1, \hat{r}_2, \cdots, \hat{r}_K$.

References

Han, Yuefeng, Cun-Hui Zhang, and Rong Chen. "Rank Determination in Tensor Factor Model." Available at SSRN 3730305 (2020).

Examples

```
dims <- c(16,18,20) # dimensions of tensor time series
t <- 100
r <- c(2,2,2)  # dimensions of factor series
ar1.coef <- array(seq(0.5,0.8,length.out=prod(r)),r)
F.dims <- dim(ar1.coef)
Ft <- array(NA,dim=c(t,F.dims[1],F.dims[2],F.dims[3]))
for(ir1 in 1:F.dims[1]){
 for(ir2 in 1:F.dims[2]){
   for(ir3 in 1:F.dims[3]){
     Ft[,ir1,ir2,ir3] <- arima.sim(n=t, model=list(ar=ar1.coef[ir1,ir2,ir3]))
   }
 }
}
lambda <- sqrt(prod(dims))
x <- tenFM.sim(Ft,dims=dims,lambda=lambda,A=NULL,cov='iid') # generate t*dims tensor time series
rank <- tenFM.rank(x,r,rank='BIC',method='TIPUP')  # Estimate the rank
```

---

tenFM.sim — *Generate tensor time series from given factor process and factor loading matrices*

Description

Simulate tensor time series X_t from given factor process F_t. The factor process F_t can be generate from the function tenAR.sim.

Usage

```
tenFM.sim(Ft,dims=NULL,lambda=1,A=NULL,cov='iid',rho=0.2)
```

Arguments

Ft	input of the factor process, of dimension $T \times r_1 \times r_2 \times \cdots \times r_k$. It can be TenAR(p) tensor time series generated from the function tenAR.sim.
dims	dimensions of the output tensor at each time, $d_1 \times d_2 \cdots \times d_K$.
lambda	signal strength parameter of the tensor factor models, see Details section for more information.
A	a list of the factor loading matrices $A_1, A_2, \cdots, A_K$. The default is random orthogonal matrices A_k of dimension $d_k \times r_k$.
cov	covariance matrix of the error tensor: identity ("iid"), separable Kronecker structure ("separable"), random ("random").
rho	a parameter only for "separable" covariance matrix of the error tensor. It is the off-diagonal element of the error matrices, with the diagonal being 1.

Details

To simulate tensor time series X_t, We consider the following model,

$$X_t = \lambda F_t \times_1 A_1 \times_2 \cdots \times_K A_k + E_t,$$

where A_k is the deterministic loading matrix of size $d_k \times r_k$ and $r_k \ll d_k$, the core tensor F_t itself is a latent tensor factor process of dimension $r_1 \times \cdots \times r_K$, $lambda$ is an additional signal strength parameter, and the idiosyncratic noise tensor E_t is uncorrelated (white) across time. In this function, the default A_k are orthogonal matrices.

Value

A tensor-valued time series of dimension $T \times d_1 \times d_2 \cdots \times d_K$.

See Also

tenAR.sim

Examples

```
dims <- c(16,18,20) # dimensions of tensor time series
t <- 100
r <- c(2,2,2)  # dimensions of factor series
ar1.coef <- array(seq(0.5,0.8,length.out=prod(r)),r)
F.dims <- dim(ar1.coef)
Ft <- array(NA,dim=c(t,F.dims[1],F.dims[2],F.dims[3]))
for(ir1 in 1:F.dims[1]){
 for(ir2 in 1:F.dims[2]){
   for(ir3 in 1:F.dims[3]){
     Ft[,ir1,ir2,ir3] <- arima.sim(n=t, model=list(ar=ar1.coef[ir1,ir2,ir3]))
   }
 }
}
lambda <- sqrt(prod(dims))
# generate t*dims tensor time series with iid error covaraince structure
x <- tenFM.sim(Ft,dims=dims,lambda=lambda,A=NULL,cov='iid')
# generate t*dims tensor time series with separable error covaraince structure
x <- tenFM.sim(Ft,dims=dims,lambda=lambda,A=NULL,cov='separable',rho=0.2)
```

```
# generate t*dims tensor time series with random error covaraince structure
dims <- c(30,20) # dimensions of tensor time series
t <- 100
r <- c(3,3)  # dimensions of factor series
ar1.coef <- array(seq(0.5,0.8,length.out=prod(r)),r)
F.dims <- dim(ar1.coef)
Ft <- array(NA,dim=c(t,F.dims[1],F.dims[2]))
for(ir1 in 1:F.dims[1]){
 for(ir2 in 1:F.dims[2]){
   Ft[,ir1,ir2] <- arima.sim(n=t, model=list(ar=ar1.coef[ir1,ir2]))
 }
}
lambda <- sqrt(prod(dims))
x <- tenFM.sim(Ft,dims=dims,lambda=lambda,A=NULL,cov='random')
```

Index