Package 'tensorTS'

May 8, 2022

Type Package

Title Factor and Autoregressive Models for Tensor Time Series

Version 1.0.0

Description Factor and autoregressive models for matrix and tensor valued time series. We provide functions for estimation, simulation and prediction. The models are discussed in Li et al (2021) <arXiv:2110.00928>, Chen et al (2020) <DOI:10.1080/01621459.2021.1912757>, Chen et al (2020) <DOI:10.1016/j.jeconom.2020.07.015>, and Xiao et al (2020) <arXiv:2006.02611>.

License GPL (>= 2)

Encoding UTF-8

Depends tensor, rTensor, expm

Imports methods, stats, MASS, abind, Matrix, pracma, graphics

URL https://github.com/zebang/tensorTS

BugReports https://github.com/ZeBang/tensorTS/issues

RoxygenNote 7.1.2

NeedsCompilation no

Author Zebang Li [aut, cre],
Ruofan Yu [aut],
Rong Chen [aut],
Yuefeng Han [aut],
Han Xiao [aut],
Dan Yang [aut]

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

Repository CRAN

Date/Publication 2022-05-08 15:10:02 UTC

R topics documented:

`matAR.RR.est` *Estimation for Reduced Rank MAR(1) Model*

Description

Estimation of the reduced rank MAR(1) model, using least squares (RRLSE) or MLE (RRMLE), as determined by the value of `method`.

Usage

```
matAR.RR.est(xx, method, A1.init=NULL, A2.init=NULL,Sig1.init=NULL,Sig2.init=NULL,
k1=NULL, k2=NULL, niter=200,tol=1e-4)
```

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)).
`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.
`Sig1.init`	only if `method=RRMLE`, initial value of Σ_1. The default is the identity matrix.
`Sig2.init`	only if `method=RRMLE`, initial value of Σ_2. 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.

Details

The reduced rank MAR(1) model takes the form:

$$X_t = A_1 X_{t-1} A_2^\top + E_t,$$

where A_i are $d_i \times d_i$ coefficient matrices of ranks $\mathrm{rank}(A_i) = k_i \leq d_i$, $i = 1, 2$. For the MLE method we also assume

$$\mathrm{Cov}(\mathrm{vec}(E_t)) = \Sigma_2 \otimes \Sigma_1$$

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.

`loading` a list of estimated U_i, V_i, where we write $A_i = U_i D_i V_i$ as the singular value decomposition (SVD) of A_i, $i = 1, 2$.

`Sig1` only if `method=MLE`, when $\mathrm{Cov}(\mathrm{vec}(E_t)) = \Sigma_2 \otimes \Sigma_1$.

`Sig2` only if `method=MLE`, when $\mathrm{Cov}(\mathrm{vec}(E_t)) = \Sigma_2 \otimes \Sigma_1$.

`res` residuals.

`Sig` sample covariance matrix of the residuals vec($\hat{E}_t$).

`cov` a list containing

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

`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.A1` element-wise standard errors of $\hat{A}_1$, aligned with `A1`.

`sd.A2` element-wise standard errors of $\hat{A}_2$, aligned with `A2`.

`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

```
set.seed(333)
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)
```

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

Description

Asymptotic covariance matrix of the 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

```
matAR.RR.se(A1,A2,k1,k2,method,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 A_1.
k2	rank of A_2.
method	character string, specifying the method of the estimation to be used.
	"RRLSE", Least squares.
	"RRMLE", MLE under a separable cov(vec(E_t)).
Sigma.e	only if method = "RRLSE". Cov(vec(E_t)) = Sigma.e: covariance matrix of dimension $(d_1 d_2) \times (d_1 d_2)$
Sigma1, Sigma2	only if method = "RRMLE". Cov(vec(E_t)) = $\Sigma_2 \otimes \Sigma_1$. Σ_i is $d_i \times d_i$, $i = 1, 2$.
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 a given slice of a tensor-valued time series.

Usage

```
mplot(xx)
```

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.

Value

a figure.

Examples

```
dim <- c(3,3,3)
xx <- tenAR.sim(t=50, dim, R=2, P=1, rho=0.5, cov='iid')
mplot(xx[1:30,,,1])
```

---

mplot.acf *Plot ACF of Matrix-Valued Time Series*

Description

Plot ACF of matrix-valued time series, can be also used to plot ACF of a given slice of a tensor-valued time series.

Usage

```
mplot.acf(xx)
```

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.

Value

a figure.

Examples

```
dim <- c(3,3,3)
xx <- tenAR.sim(t=50, dim, R=2, P=1, rho=0.5, cov='iid')
mplot.acf(xx[1:30,,,1])
```

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

Description

Estimation function for tensor autoregressive models. Methods include projection (PROJ), Least Squares (LSE), maximum likelihood estimation (MLE) and vector autoregressive model (VAR), as determined by the value of `method`.

Usage

```
tenAR.est(xx,R=1,P=1,method="LSE",init.A=NULL,init.sig=NULL,niter=150,tol=1e-6)
```

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, a vector for $P > 1$, a positive integer, it assumes same number of terms in each lag.
`P`	Autoregressive order, a positive integer.
`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(P) model for the $\mathrm{vec}(E_t)$.
`init.A`	initial values of coefficient matrices $A_k^{(ir)}$ in estimation algorithms, which is a multi-layer list such that the first layer for the lag $1 \le i \le P$, the second the term $1 \le r \le R$, and the third the mode $1 \le k \le K$. See "Details". By default, we use PROJ 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 (of autoregressive order one) has the 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 Kronecker rank. The model of autoregressive order P takes the form

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

For the "MLE" method, we also assume,

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

Value

return a list containing the following:

`A` a list of estimated coefficient matrices $A_k^{(ir)}$. It is a multi-layer list, the first layer for the lag $1 \le i \le P$, the second the term $1 \le r \le R$, and the third the mode $1 \le k \le K$. See "Details".

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

`res` residuals

`Sig` sample covariance matrix of the residuals vec($\hat{E}_t$).

`cov` grand covariance matrix of all estimated entries of $A_k^{(ir)}$

`sd` standard errors of the coefficient matrices $A_k^{(ir)}$, 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.

Zebang Li, Han Xiao. "Multi-linear tensor autoregressive models". arxiv preprint arxiv:2110.00928 (2021).

Examples

```
set.seed(333)

# case 1: tensor-valued time series

dim <- c(2,2,2)
xx <- tenAR.sim(t=100, dim, R=2, P=1, rho=0.5, cov='iid')
est <- tenAR.est(xx, R=2, P=1, method="LSE") # two-term tenAR(1) model
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

# est <- tenAR.est(xx, R=c(1,2), P=2, method="LSE") # tenAR(2) model with R1=1, R2=2

# case 2: matrix-valued time series

dim <- c(2,2)
xx <- tenAR.sim(t=100, dim, R=2, P=1, rho=0.5, cov='iid')
est <- tenAR.est(xx, R=2, P=1, method="LSE") # two-term MAR(1) model
```

```
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]]) == 2 # TRUE, since the mode K = 2
```

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

Description

Prediction based on the tensor autoregressive model or reduced rank MAR(1) model. If `rolling = TRUE`, returns the rolling forecasts.

Usage

```
tenAR.predict(object, n.ahead = 1, xx = NULL, rolling = FALSE, n0 = NULL)
```

Arguments

`object`	a model object returned by `tenAR.est()`.
`n.ahead`	prediction horizon.
`xx`	$T' \times d_1 \times \cdots \times d_K$ new tensor time series to be used for prediction. Must have at least `n.ahead` length.
`rolling`	TRUE or FALSE, rolling forecast, is FALSE by default.
`n0`	only if `rolling = TRUE`, the starting point of rolling forecast.

Value

a tensor time series of length `n.ahead` if `rolling = FALSE`;

a tensor time series of length $T' - n_0 - n.ahead + 1$ if `rolling = TRUE`.

See Also

'predict.ar' or 'predict.arima'

Examples

```
set.seed(333)
dim <- c(2,2,2)
t = 20
xx <- tenAR.sim(t, dim, R=2, P=1, rho=0.5, cov='iid')
est <- tenAR.est(xx, R=1, P=1, method="LSE")
pred <- tenAR.predict(est, n.ahead = 1, xx = xx)
# rolling forcast
n0 = t - min(50,t/2)
pred.rolling <- tenAR.predict(est, n.ahead = 5, xx = xx, rolling=TRUE, n0)
```

```
# prediction for reduced rank MAR(1) model
dim <- c(2,2)
t = 20
xx <- tenAR.sim(t, dim, R=1, P=1, rho=0.5, cov='iid')
est <- matAR.RR.est(xx, method="RRLSE", k1=1, k2=1)
pred <- tenAR.predict(est, n.ahead = 1)
# rolling forcast
n0 = t - min(50,t/2)
pred.rolling <- tenAR.predict(est, n.ahead = 5, rolling=TRUE, n0=n0)
```

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

Description

Simulate from the TenAR(p) model.

Usage

```
tenAR.sim(t, dim, R, P, rho, cov, A = NULL, Sig = NULL)
```

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").
`A`	coefficient matrices. If not provided, they are randomly generated according to given `dim`, `R`, `P` and `rho`. It is a multi-layer list, the first layer for the lag $1 \leq i \leq P$, the second the term $1 \leq r \leq R$, and the third the mode $1 \leq k \leq K$. See "Details" of `tenAR.est`.
`Sig`	only if `cov=mle`, a list of initial values of $\Sigma_1, \ldots, \Sigma_K$. The default are identity matrices.

Value

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

See Also

`tenFM.sim`

Examples

```
set.seed(123)
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,tol=1e-4,maxiter=100)
```

Arguments

`x`	$T \times d_1 \times \cdots \times d_K$ tensor-valued time series.
`r`	input rank of 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.
`tol`	tolerance in terms of the Frobenius norm.
`maxiter`	maximum number of iterations if error stays above `tol`.

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 an 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

returns 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 (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

```
set.seed(333)
dims <- c(16,18,20) # dimensions of tensor time series
r <- c(3,3,3)  # dimensions of factor series
Ft <- tenAR.sim(t=100, dim=r, R=1, P=1, rho=0.9, cov='iid')
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,h0=1,iter=TRUE,method='TIPUP')  # 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 (IC) 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='IC',method='TIPUP',inputr=FALSE,iter=TRUE,penalty=1,
delta1=0,tol=1e-4,maxiter=100)
```

Arguments

x	$T \times d_1 \times \cdots \times d_K$ tensor-valued time series.
r	initial guess of the rank of 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. "IC", 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, always use initial guess rank r in each iteration; if FLASE, the rank will be updated in each iteration.
iter	boolean, specifying using an iterative approach or a non-iterative approach.
penalty	takes value in 1,2,3,4,5, decides which penalty function to use for each tesnor mode k. Here ν is a tuning parameter defined in the argument "delta1", and $d = \prod_{i=1}^{K} d_k$. When rank= 'IC': 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 IC method only
tol	tolerance in terms of the Frobenius norm.
maxiter	maximum number of iterations if error stays above tol.

Details

Let W be a $p \times p$ symmetric and non-negative definite matrix and $\widehat{W}$ be its sample version, $\hat{\lambda}_j$ be the eigenvalues of $\widehat{W}$ such that $\hat{\lambda}_1 \geq \hat{\lambda}_2 \geq \cdots \hat{\lambda}_p$. The rank determination methods using the

information criterion ("IC") and the eigen ratio criterion ("ER") are defined as follows:

$$IC(\widehat{W}) = \text{argmin}_{0\le m\le m^*}\left\{\sum_{j=m+1}^{p}\hat{\lambda}_j + mg(\widehat{W})\right\},$$

$$ER(\widehat{W}) = \text{argmin}_{0\le m\le m^*}\left\{\frac{\hat{\lambda}_{m+1}+h(\widehat{W})}{\hat{\lambda}_m+h(\widehat{W})}\right\},$$

where m^* is a predefined upper bound, g and h are some appropriate positive penalty functions. We have provided 5 choices for g and h; see more details in the argument "penalty". For non-iterative TOPUP and TIPUP methods, $\widehat{W}$ is $\text{mat}_1(\text{TOPUP}_k(X_{1:T}))\text{mat}_1(\text{TOPUP}_k(X_{1:T}))^\top$ or $(\text{TIPUP}_k(X_{1:T}))(\text{TIPUP}_k(X_{1:T}))^\top$, for each tensor mode k, $1\le k\le K$, where $\text{TOPUP}_k(X_{1:T})$ and $\text{TIPUP}_k(X_{1:T})$ are defined in the Details section of the function `tenFM.est`. For iterative TOPUP and TIPUP methods, we refer to the literature in the References section for more information.

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 first row is the estimated rank under non-iterative approach, the $i+1$-th row is the estimated rank $\hat{r}_1,\hat{r}_2,\cdots,\hat{r}_K$ at (i)-th iteration.

`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

```
set.seed(333)
dims <- c(16,18,20) # dimensions of tensor time series
r <- c(3,3,3)  # dimensions of factor series
Ft <- tenAR.sim(t=100, dim=r, R=1, P=1, rho=0.9, cov='iid')
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=c(4,4,4),h0=1,rank='IC',iter=TRUE,method='TIPUP')  # Estimate the rank
```

`tenFM.sim` *Generate Tensor Time series using given Factor Process and Factor Loading Matrices*

Description

Simulate tensor time series X_t using a given factor process F_t. The factor process F_t can be generated by 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 by 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

Simulate from the 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 $r_1 \times \cdots \times r_K$, λ is an additional signal strength parameter, and the idiosyncratic noise tensor E_t is uncorrelated (white) across time. In this function, by 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

```
set.seed(333)
dims <- c(16,18,20) # dimensions of tensor time series
r <- c(3,3,3)  # dimensions of factor series
Ft <- tenAR.sim(t=100, dim=r, R=1, P=1, rho=0.9, cov='iid')
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)
```

Index