Package ‘timeFA’

June 26, 2020

Type Package

Title Factor Models and Autoregressive Models for Time Series

Version 0.1.0

Author@R person(``Zebang'', ``Li'', email = ``zebang.li@rutgers.edu'',
role = c(``aut'', ``cre''))

Description More about what it does (maybe more than one line)
Use four spaces when indenting paragraphs within the Description.

License GPL (>= 2)

Encoding UTF-8

LazyData true

Imports tensor,
rTensor,
MASS,
abind,
igraph

RoxygenNote 7.0.2

VignetteBuilder knitr

R topics documented:

`dynamic_A` *dynamic_A*

Description

Get the adjacency matrix for plotting

Usage

```
dynamic_A(x, factor_count, simple.flag, threshold)
```

Arguments

`x`	input the original estimated loading matrix
`factor_count`	The number of factors to use
`simple.flag`	if True, only eliminate the entries below threshold and make all row sums to be 1; if False, the approach further eliminates the entries of the rows that are very close to threshold value and only leaves the maximum entry of each row
`threshold`	A parameter to eliminate very small entries of the loading matrix

Value

The new loading matrix with all rows sum to be 1

em — *Permutation matrix em*

Description

Permutation matrix em.

Usage

```
em(m, n, i, j)
```

Value

Permutation matrix em #' @seealso `trearrange`

Examples

```
em(m,n,i,j)
```

generate — *Generate an AR(1) tensor time series*

Description

For test only, randomly generate matrices A1,A2,..Ak by iid standard gaussians with given dimensions, and then generate an AR(1) tensor time series.

Usage

```
generate(dim, t)
```

Arguments

`dim`	an array of dimentions of matrices $A_1, A_2, \cdots, A_k$
`t`	length of time

Value

a list containing the following:

`A` matrices $A_1, A_2, \cdots, A_k$

`X` AR(1) tensor time series

#' @seealso `run.test`

Examples

```
dim <- c(1,2,3)
T <- 100
xx <- generate(c(m1,m2,m3),T)
```

grouping.loading *grouping.loading*

Description

Get the group of loadings

Usage

```
grouping.loading(loading, ncluster, rowname, plot = T)
```

Arguments

`loading`	The estimated loading matrix
`ncluster`	The number of clusters to use, usually the dimension of the factor matrix
`rowname`	The name of the rows
`plot`	plot the clustering graph, defacult True

Value

Loading matrix after grouping

MAR *Matrix Method*

Description

Estimation function for matrix-valued time series, including the projection method, the Iterated least squares method and MLE under a structured covariance tensor, as determined by the value of `type`.

Usage

```
MAR(xx, type = c("projection", "LS", "MLE", "ar"))
```

Arguments

`xx`	T * p * q matrix-valued time series
`type`	character string, specifying the type of the estimation method to be used. `"projection"`, Projection method. `"LS"`, Iterated least squares. `"MLE"`, MLE under a structured covariance tensor. `"ar"`, Stacked vector AR(1) Model.

Value

a list containing the following:

`LL` estimator of LL, a p by p matrix

`RR` estimator of RR, a q by q matrix

`res` residual of the MAR(1)

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

`MAR.SE` *Asymptotic Covariance Matrix of* `MAR1.otimes`

Description

Asymptotic covariance Matrix of `MAR1.otimes` for given A, B and matrix-valued time series xx, see Theory 3 in paper.

Usage

```
MAR.SE(xx, B, A, Sigma)
```

Arguments

`xx`	T * p * q matrix-valued time series
`B`	q by q matrix in MAR(1) model
`A`	p by p matrix in MAR(1) model
`Sig`	covariance matrix cov(vec(E_t)) in MAR(1) model

Value

asmptotic covariance matrix

Examples

```
# given T * p * q time series xx
out2=MAR1.LS(xx)
FnormLL=sqrt(sum(out2$LL))
xdim=p;ydim=q
out2Xi=MAR.SE(xx.nm,out2$RR*FnormLL,out2$LL/FnormLL,out2$Sig)
out2SE=sqrt(diag(out2Xi))
SE.A=matrix(out2SE[1:xdim^2],nrow=xdim)
SE.B=t(matrix(out2SE[-(1:xdim^2)],nrow=ydim))
```

MAR1.LS — *Least Squares Iterative Estimation*

Description

Iterated least squares estimation in the model $X_t = LL * X_{t-1} * RR + E_t$.

Usage

```
MAR1.LS(xx, niter = 50, tol = 1e-06, print.true = FALSE)
```

Arguments

`xx`	T * p * q matrix-valued time series
`niter`	maximum number of iterations if error stays above `tol`
`tol`	relative Frobenius norm error tolerance
`print.true`	printe LL and RR

Value

a list containing the following:

`LL` estimator of LL, a p by p matrix

`RR` estimator of RR, a q by q matrix

`res` residual of the MAR(1)

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

`dis` Frobenius norm difference of last update

`niter` number of iterations

MAR1.otimes — *MLE under a structured covariance tensor*

Description

MAR(1) iterative estimation with Kronecker covariance structure: $X_t = LL * X_{t-1} * RR + E_t$ such that $\Sigma = cov(vec(E_t)) = \Sigma_r \otimes \Sigma_l$.

Usage

```
MAR1.otimes(
  xx,
  LL.init = NULL,
  Sigl.init = NULL,
  Sigr.init = NULL,
  niter = 50,
  tol = 1e-06,
  print.true = FALSE
)
```

Arguments

`xx`	T * p * q matrix-valued time series
`LL.init`	initial value of LL
`Sigl.init`	initial value of Sigl
`Sigr.init`	initial value of Sigr
`niter`	maximum number of iterations if error stays above `tol`
`tol`	relative Frobenius norm error tolerance
`print.true`	printe LL and RR

Value

a list containing the following:

`LL` estimator of LL, a p by p matrix

`RR` estimator of RR, a q by q matrix

`res` residual of the MAR(1)

`Sigl` one part of structured covariance matrix $\Sigma = \Sigma_r \otimes \Sigma_l$

`Sigr` one part of structured covariance matrix $\Sigma = \Sigma_r \otimes \Sigma_l$

`dis` Frobenius norm difference of the final update step

`niter` number of iterations

`MAR1.projection` *Projection Method*

Description

MAR(1) one step projection estimation in the model $X_t = LL * X_{t-1} * RR + E_t$.

Usage

```
MAR1.projection(xx)
```

Arguments

`xx`	T * p * q matrix-valued time series

Value

a list containing the following:

`LL` estimator of LL, a p by p matrix

`RR` estimator of RR, a q by q matrix

`res` residual of the MAR(1)

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

matrix_factor *matrix_factor*

Description

The main estimation function

Usage

```
matrix_factor(Yt, inputk1, inputk2, iscentering = 1, hzero = 1)
```

Arguments

`Yt`	Time Series data for a matrix
`inputk1`	The pre-determined row dimension of the factor matrix
`inputk2`	The pre-determined column dimension of the factor matrix
`iscentering`	The data is subtracted by its mean value
`hzero`	Pre-determined parameter

Value

a list containing the following:

`eigval1` estimated row dimension of the factor matrix

`eigval2` estimated column dimension of the factor matrix

`loading1` estimated left loading matrix

`loading2` estimated right loading matrix

`Ft` Estimated factor matrix with pre-determined number of dimensions

`Ft.all` Sum of Ft

`Et` The estimated residual, by subtracting estimated signal term from the data

Examples

```
A <- 1:180
dim(A) <- c(3,3,20)
out = matrix_factor(A,3,3)
eig1 = out$eigval1
loading1 = out$loading1
Ft = out$Ft.all
```

`mfmda`	*mfmda*

Description

This is a wrapper for all approaches

Usage

```
mfmda(Yt, approach = "3", hzero = 1, iscentering = 1)
```

Arguments

`approach`	Select estimation approaches, 1 for noniterative approach with no NaNs, 2 for iterative approach with NaNs, 3 for iterative approach allowing NaNs.
`hzero`	Pre-determined parameter
`iscentering`	The data is subtracted by its mean value
`Yc`	Time Series data for a matrix

Value

The sample version of M matrix

Examples

```
A <- 1:180
dim(A) <- c(3,3,20)
M <- mfmda(A,"3",1,0)
```

`mfmda.estqk`	*mfmda.estqk*

Description

Compute the estimated number of factors and the corresponding eigen-space

Usage

```
mfmda.estqk(Mhat, inputk = 1)
```

Arguments

`Mhat`	The estimated value for matrix M
`inputk`	The pre-determined number of dimension of factor matrix

Value

The estimated number of factors to use, the corresponding estimated Q matrix, the eigenvalue, the estimated Q matrix with requested number of factors

Examples

```
A <- 1:180
dim(A) <- c(3,3,20)
M <- mfmda(A,"3",1,0)
inputk <- 3
eig.ans <- mfmda.estqk(M,inputk)
khat <- eig.ans$estk
Qhat <- eig.ans$Qhatestk
eigval <- eig.ans$eigval
Qhatinputk <- eig.ans$Qhatinputk
```

`mfmda.na.iter` *mfmda.na.iter*

Description

The input data could have NaNs. The estimation approach is iterative.

Usage

```
mfmda.na.iter(Yc, hzero)
```

Arguments

`Yc`	Time Series data for a matrix allowing NaNs
`hzero`	Pre-determined parameter

Value

The sample version of M matrix

`mfmda.na.vec` *mfmda.na.vec*

Description

This approach is for the vector-valued estimation with NaNs.

Usage

```
mfmda.na.vec(Yc, hzero)
```

Arguments

`Yc`	Time Series data for a matrix(dimensions n*p*q), allowing NA input
`hzero`	Pre-scribed parameter h

Value

The sample version of M matrix

mfmda.nona.iter *mfmda.nona.iter*

Description

The input data do not have zeros. The estimation approach is iterative.

Usage

```
mfmda.nona.iter(Yc, hzero)
```

Arguments

Yc	Time Series data for a matrix(dimensions n*p*q), no NA input allowed
hzero	Pre-scribed parameter

Value

The sample version of M matrix

mfmda.nona.noniter *mfmda.nona.noiter*

Description

The input data do not have zeros. The estimation approach is noniterative.

Usage

```
mfmda.nona.noniter(Yc, hzero)
```

Arguments

Yc	Time Series data for a matrix(dimensions n*p*q), no NA input allowed
hzero	Pre-scribed parameter

Value

The sample version of M matrix

`mfmda.nona.vec` *mfmda.nona.vec*

Description

This approach is for the vector-valued estimation WITHOUT NaNs.

Usage

```
mfmda.nona.vec(Yc, hzero)
```

Arguments

`Yc`	Time Series data for a matrix(dimensions n*p*q), no NA input allowed
`hzero`	Pre-scribed parameter

Value

The sample version of M matrix

Examples

```
A <- 1:180
dim(A) <- c(3,3,20)
M <- mfmda.nona.vec(A,2)
```

`PlotNetwork_AB` *PlotNetwork_AB*

Description

Plot the network graph

Usage

```
PlotNetwork_AB(Ft, iterated_A, iterated_B = iterated_A, labels = use2)
```

Arguments

`Ft`	The estimated factor matrix
`iterated_A`	The left loading matrix
`iterated_B`	The right loading matrix
`labels`	The row labels

Value

Plot the network graph

`PM` *Permutation matrix PM*

Description

Permutation matrix PM.

Usage

```
PM(m, n)
```

Arguments

`m`	an array of dimentions of matrices $A_1, A_2, \cdots, A_k$
`n`	length of time

Value

Permutation matrix PM #' @seealso `trearrange`

Examples

```
PM(m,n)
```

`projection` *Kronecker Product Approximation*

Description

Kronecker product approximation used in Projection Method of matrix-value time series.

Usage

```
projection(A, m1, m2, n1, n2)
```

Arguments

`A`	m by n matrix such that $m = m1 * n1$ and $n = m2 * n2$
`m1`	`ncol` of A
`m2`	`ncol` of B
`n1`	`nrow` of A
`n2`	`nrow` of B

Value

a list contaning two estimator (matrix)

See Also

`MAR1.projection`

Examples

```
A <- matrix(runif(6),ncol=2),
projection(A,3,3,2,2)
```

`rearrange` *Rearrangement Operator*

Description

Rearrangement Operator used for projection method.

Usage

```
rearrange(A, m1, m2, n1, n2)
```

Arguments

A	m by n matrix such that $m = m1 * n1$ and $n = m2 * n2$
m1	`ncol` of A
m2	`ncol` of B
n1	`nrow` of A
n2	`nrow` of B

Value

rearengement matrix

See Also

MAR1.projection

Examples

```
A <- matrix(runif(6),ncol=2),
B <- matrix(runif(6),ncol=2),
M <- kronecker(B,A)
rearrange(M,3,3,2,2) == t(as.vector(A)) %*% as.vector(B)
'TRUE'
```

`run.test` *Test Function*

Description

For test only

Usage

```
run.test(m1, m2, m3, n = 100, T)
```

TAR *Tensor Method*

Description

Estimation function for tensor-valued time series, including the projection method, the Iterated least squares method, MLE under a structured covariance tensor and stacked vector AR(1) model, as determined by the value of `type`.

Usage

```
TAR(xx, type = c("projection", "LS", "MLE", "ar"))
```

Arguments

`xx`	$T * m_1 * \cdots * m_K$ tensor-valued time series
`type`	character string, specifying the type of the estimation method to be used. `"projection",` Projection method. `"LS",` Iterated least squares. `"MLE",` MLE under a structured covariance tensor. `"ar",` Stacked vector AR(1) Model.

Value

a list containing the following:

`A` estimator of coeficient matrices $A_1, A_2, \cdots, A_K$

`res` residual of the MAR(1)

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

`niter` number of iterations

TAR.SE *Asymptotic Covariance Matrix of* `TAR1.LS`

Description

Asymptotic covariance Matrix of `TAR1.LS` for given A tensor-valued time series xx, see Theory 2 in paper.

Usage

```
TAR.SE(xx, C, B, A, Sigma)
```

Arguments

`xx`	$T * m_1 * \cdots * m_K$ tensor-valued time series
`C`	m3 by m3 matrix in MAR(1) model
`B`	m2 by m2 matrix in MAR(1) model
`A`	m1 by m1 matrix in MAR(1) model
`Sig`	covariance matrix cov(vec(E_t)) in TAR(1) model

Value

asmptotic covariance matrix

`TAR1.LS` *Least Squares Iterative Estimation for Tensor-Valued Time Series*

Description

Iterated least squares estimation in the model $X_t = X_{t-1} \times A_1 \times \cdots \times A_K + E_t$.

Usage

```
TAR1.LS(xx, niter = 1000, tol = 1e-06, print.true = FALSE)
```

Arguments

`xx`	$T * m_1 * \cdots * m_K$ tensor-valued time series
`niter`	maximum number of iterations if error stays above `tol`
`tol`	relative Frobenius norm error tolerance
`print.true`	printe A_i

Value

a list containing the following:

`A` estimator of coeficient matrices $A_1, A_2, \cdots, A_K$
`res` residual of the MAR(1)
`Sig` covariance matrix cov(vec(E_t))
`niter` number of iterations

`TAR1.projection` *Projection Method for Tensor-Valued Time Series*

Description

TAR(1) one step projection estimation in the model $X_t = X_{t-1} \times A_1 \times \cdots \times A_K + E_t$.

Usage

```
TAR1.projection(xx)
```

Arguments

`xx`	$T * m_1 * \cdots * m_K$ tensor-valued time series
`m1`	dim(A1)
`m2`	dim(A2)
`m3`	dim(A3)

Value

a list containing the estimation of matrices $A_1, A_2, \cdots, A_K$

TAR1.VAR	*Stacked vector AR(1) Model for Tensor-Valued Time Series*

Description

vector AR(1) Model.

Usage

```
TAR1.VAR(xx)
```

Arguments

`xx` $T * m_1 * \cdots * m_K$ tensor-valued time series

Value

a list containing the following:

`coef` coeficient of the fitted VAR(1) model

`res` residual of the VAR(1) model

trearrange	*trearrange*

Description

(alpha version) rearrangement operator for tensor.

Usage

```
trearrange(A, m1, m2, m3, n1, n2, n3)
```

var1	*Stacked vector AR(1) Model*

Description

vector AR(1) Model.

Usage

```
var1(xx)
```

Arguments

`xx` T * p * q matrix-valued time series

Value

a list containing the following:

`coef` coeficient of the fitted VAR(1) model

`res` residual of the VAR(1) model

Examples

```
out.var1=var1(xx)
sum(out.var1$res**2)
```

---

vector_factor *vector_factor*

Description

The main estimation function, vector version

Usage

```
vector_factor(Yt, inputk.vec, iscentering = 1, hzero = 1)
```

Arguments

Yt	Time Series data for a matrix
inputk.vec	The pre-determined dimensions of the factor matrix in vector
iscentering	The data is subtracted by its mean value
hzero	Pre-determined parameter

Value

a list containing the following:

`eigval1` estimated dimensions of the factor matrix

`loading` estimated loading matrices

`Ft` Estimated factor matrix with pre-determined number of dimensions

`Ft.all` Sum of Ft

`Et` The estimated random term, by subtracting estimated signal term from the data

Examples

```
A <- 1:180
dim(A) <- c(3,3,20)
M <- mfmda(A,"3",1,0)
eig.ans <- vector_factor(M,3,0,1)
khat <- eig.ans$estk
Qhat <- eig.ans$Qhatestk
eigval <- eig.ans$eigval
Q1hatinputk <- eig.ans$Qhatinputk
```

Index