MDPCA {MDPCA} R Documentation

Moving Dynamic Principal Component Analysis for Non- Stationary Multivariate Time Series

Description

This function reduce the dimension of non-stationary (and stationary) multivariate time series by performing eigenanalysis on the moving cross-covriance matrix of the extended data matrix up to some specified lag. Notice that the following libraries are needed to be installed before using the MDPCA function: library(roll); library(expm).

Usage

```
MDPCA(x,w,l)
```

Arguments

`x`
a T-by-m data matrix, where the rows are "T" time points, and the columns are "m" variables

`w`
window width (i.e. window length)

`l`
number of lagged series to be included in the calculation of MDPCA

Value

`F`
returns the moving cross-covariance matrix of the extended data matrix xdata

`Lambda`
returns the eigenvalues of the matix F

`U`
returns the eigenvectors of the matix F

`xdata`
returns the extended data matrix of x

Note

step1: Build the extended data matrix (i.e. xdata) and obtain its eigenvalues and eigenvectors. step2: Transfer extended data matrix (i.e. xdata) using eigenvectors (i.e. U) that correspond to the largest eigenvalues (i.e. Lambda). For example, if we find the first two eigenvalues to be large enough, then we can choose the corresponding two eigenvectors to obtain the final results (i.e. two MDPCs).See the examples below.

Author(s)

Fayed Alshammri

References

Alshammri, F. and Pan, J. (2019). Moving dynamic principal component analysis for non-stationary multivariate time series. Manuscript submitted for publication.

Examples

```
##This is Example 2 of Alshammri and Pan (2019).
##The data matrix X is a non-stationary time series with m=10 and T=1500.
m=10;T=1500
# Generate x_t
X=mat.or.vec(m,T)
a1=arima.sim(list(order=c(1,1,1),ar=0.75,ma=0.9),n=T+1,sd=1)
for(i in 1:2) X[i,]=a1[i+1:T]
a2=arima.sim(list(order=c(1,1,1),ar=0.6,ma=-1.4),n=T+1,sd=1)
for(i in 3:4) X[i,]=a2[(i-1):(T+i-2)]
a3=arima.sim(list(order=c(1,1,1),ar=-0.7,ma=-2.3),n=T+1,sd=1)
for(i in 5:6) X[i,]=a3[(i-3):(T+i-4)]
a4=arima.sim(list(order=c(1,1,1),ar=-0.5,ma=0.55),n=T+1,sd=1)
for(i in 7:8) X[i,]=a4[(i-5):(T+i-6)]
a5=arima.sim(list(order=c(1,1,1),ar=0.6,ma=1.65),n=T+1,sd=1)
for(i in 9:10) X[i,]=a5[(i-7):(T+i-8)]
X=t(X)
X=ts(X)
##apply MDPCA with 100 window length and 5 lagged series.
Analysis=MDPCA(X,100,5)
U=Analysis$U
Lambda=Analysis$Lambda
F=Analysis$F
xdata=Analysis$xdata
##For example, if we find the first two eigenvalues to be large enough, then we can choose the corresponding two eigenvectors to obtain the final results (i.e. two MDPCs)
sum(Lambda[1:2])/sum(Lambda)
Transform=xdata%*%U[,1:2]
##Final results (i.e. two MDPCs)
Transform=ts(Transform)
```
