Package ‘pcd pca’

October 14, 2022

Title Dynamic Principal Components for Periodically Correlated Functional Time Series

Version 0.4

Description Method extends multivariate and functional dynamic principal components to periodically correlated multivariate time series. This package allows you to compute true dynamic principal components in the presence of periodicity. We follow implementation guidelines as described in Kidzinski, Kokoszka and Jouzdani (2017), in Principal component analysis of periodically correlated functional time series <arXiv:1612.00040>.

Depends R (>= 3.3.1)

Imports freqdom, fda

License GPL-3

Encoding UTF-8

LazyData true

RoxygenNote 6.0.1

NeedsCompilation no

Author Lukasz Kidzinski [aut, cre], Neda Jouzdani [aut], Piotr Kokoszka [aut]

Maintainer Lukasz Kidzinski <lukasz.kidzinski@stanford.edu>

Repository CRAN

Date/Publication 2017-09-03 05:17:18 UTC

R topics documented:

  pcdpca .................................................. 2
  pcdpca.inverse ....................................... 3
  pcdpca.scores ......................................... 4

Index 5
pcd pca

Compute periodically correlated DPCA filter coefficients

Description
For a given periodically correlated multivariate process X eigendecompose it’s spectral density and use an inverse fourier transform to get coefficients of the optimal filters.

Usage
pcd pca(X, period = NULL, q = 30, freq = (-1000:1000/1000) * pi)

Arguments
- **X**: multivariate stationary time series
- **period**: period of the periodic time series
- **q**: window for spectral density estimation as in spectral.density
- **freq**: frequency grid to estimate on as in spectral.density

Value
principal components series

References

See Also
pcd pca.inverse, pcd pca.scores

Examples
```r
## Prepare some process
library(fda)
library(freqdom)

MSE = function(X,Y=0){ sum((X-Y)**2) / nrow(X) }

d = 7
n = 100
A = t(t(matrix(rnorm(d*n),ncol=d,nrow=n))*7:1)
B = t(t(matrix(rnorm(d*n),ncol=d,nrow=n))*7:1)
C = t(t(matrix(rnorm(d*n),ncol=d,nrow=n))*7:1)

X = matrix(0,ncol=d,nrow=3*n)
X[3*(1:n) - 1,] = A
```
\[
X[3*(1:n) - 2,] = A + B \\
X[3*(1:n) ,] = 2*A - B + C
\]

```r
basis = create.fourier.basis(nbasis=7) 
X.fd = fd(t(Re(X)),basis=basis) 
plot(X.fd)

## Hold out some datapoints 
train = 1:(50*3) 
test = (50*3) : (3*n)

## Static PCA ## 
PR = prcomp(as.matrix(X[train,])) 
Y1 = as.matrix(X) %*% PR$rotation 
Y1[,,-1] = 0 
Xpca.est = Y1 %*% t(PR$rotation)

## Dynamic PCA ## 
XI.est = dpca(as.matrix(X[train,]), 
q=3,  
freq=pi*(-150:150/150),  
Ndpc=1)  # finds the optimal filter 
Y.est = freqdom::filter.process(X, XI.est$filters ) 
Xdpca.est = freqdom::filter.process(Y.est, t(rev(XI.est$filters)) )  # deconvolution

## Periodically correlated PCA ## 
XI.est.pc = pcdpca(as.matrix(X[train,]), 
q=3,  
freq=pi*(-150:150/150),period=3)  # finds the optimal filter 
Y.est.pc = pcdpca.scores(X, XI.est.pc)  # applies the filter 
Y.est.pc[,,-1] = 0  # forces the use of only one component 
Xpcdpca.est = pcdpca.inverse(Y.est.pc, XI.est.pc)  # deconvolution

## Results ##
cat("NMSE PCA = ") 
r0 = MSE(X[test,],Xpca.est[test,]) / MSE(X[test,],0) 
cat(r0) 
cat("NMSE DPCA = ") 
r1 = MSE(X[test,],Xdpca.est[test,]) / MSE(X[test,],0) 
cat(r1) 
cat("NMSE PCDPCA = ") 
r2 = MSE(X[test,],Xpcdpca.est[test,]) / MSE(X[test,],0) 
cat(r2) 
cat("\n")
```

**pcdpca.inverse**

Retrieves a process from given scores.
pcdpcascores

Description
For given scores Y and dynamic principal components XI retrieve a series from which scores Y were calculated. This procedure should be seen as the inverse of pcdpca.scores.

Usage
pcdpcainverse(Y, XI)

Arguments
Y scores process
XI principal components series

Value
Retrieved process X

References

See Also
pcdpcascores, pcdpca

pcdpcascores Compute periodically correlated DPCA scores, given the filters XI

Description
Compute periodically correlated DPCA scores, given the filters XI

Usage
pcdpcascores(X, XI)

Arguments
X multivariate time series
XI series of filters returned from pcdpca
Index

* pcdpca
  pcdpca.scores, 4

pcdpca, 2, 4
pcdpca.inverse, 2, 3
pcdpca.scores, 2, 4, 4
spectral.density, 2