Package ‘HDTSA’

November 8, 2021

Type Package

Title High Dimensional Time Series Analysis Tools

Version 1.0.1

Date 2021-11-08

Author Chen Lin [aut, cre],
Guanghui Cheng [aut],
Jinyuan Chang [aut],
Qiwei Yao [aut]

Maintainer Chen Lin <linchen@smail.swufe.edu.cn>


License GPL-3

Depends R (>= 3.5.0)

Imports stats, Rcpp, clime, sandwich

LinkingTo Rcpp, RcppEigen

Suggests knitr

NeedsCompilation yes

RoxygenNote 7.1.1

Encoding UTF-8

URL https://github.com/Linc2021/HDTSA

BugReports https://github.com/Linc2021/HDTSA/issues

Repository CRAN

Date/Publication 2021-11-08 09:50:02 UTC
Factors modeling: Inference for the number of factors

Description

`factors()` deals with factor modeling for high-dimensional time series proposed in Lam and Yao (2012):

\[ y_t = Ax_t + \epsilon_t, \]

where \( x_t \) is an \( r \times 1 \) latent process with (unknown) \( r \leq p \), \( A \) is a \( p \times r \) unknown constant matrix, and \( \epsilon_t \sim \text{WN}(\mu_\epsilon, \Sigma_\epsilon) \) is a vector white noise process. The number of factors \( r \) and the factor loadings \( A \) can be estimated in terms of an eigenanalysis for a nonnegative definite matrix, and is therefore applicable when the dimension of \( y_t \) is on the order of a few thousands. This function aims to estimate the number of factors \( r \) and the factor loading matrix \( A \).

Usage

`factors(Y, lag.k = 5, twostep = FALSE)`

Arguments

Y

\( Y = \{y_1, \ldots, y_n\}' \), a data matrix with \( n \) rows and \( p \) columns, where \( n \) is the sample size and \( p \) is the dimension of \( y_t \).

lag.k

Time lag \( k_0 \) used to calculate the nonnegative definite matrix \( \tilde{M} \):

\[
\tilde{M} = \sum_{k=1}^{k_0} \tilde{\Sigma}_y(k) \tilde{\Sigma}_y(k)',
\]

where \( \tilde{\Sigma}_y(k) \) is the sample autocovariance of \( y_t \) at lag \( k \).

twostep

Logical. If FALSE (the default), then standard procedures [See Section 2.2 in Lam and Yao (2012)] for estimating \( r \) and \( A \) will be implemented. If TRUE, then a two step estimation procedure [See Section 4 in Lam and Yao (2012)] will be implemented for estimating \( r \) and \( A \).
### Value

An object of class "factors" is a list containing the following components:

- `factor_num`: The estimated number of factors $\hat{r}$.
- `loading.mat`: The estimated $p \times r$ factor loading matrix $\hat{A}$.

### References


### Examples

```r
## Generate x_t
p <- 400
n <- 400
r <- 3
X <- mat.or.vec(n, r)
A <- matrix(runif(p*r, -1, 1), ncol=r)
x1 <- arima.sim(model=list(ar=c(0.6)), n=n)
x2 <- arima.sim(model=list(ar=c(-0.5)), n=n)
x3 <- arima.sim(model=list(ar=c(0.3)), n=n)
eps <- matrix(rnorm(n*p), p, n)
X <- t(cbind(x1, x2, x3))
Y <- A %*% X + eps
Y <- t(Y)
fac <- factors(Y, lag.k=2)
r_hat <- fac$factor_num
loading_Mat <- fac$loading.mat
```

### Description

`HDSReg()` considers a multivariate time series model which represents a high dimensional vector process as a sum of three terms: a linear regression of some observed regressors, a linear combination of some latent and serially correlated factors, and a vector white noise:

$$y_t = Dz_t + Ax_t + \epsilon_t,$$

where $y_t$ and $z_t$ are, respectively, observable $p \times 1$ and $m \times 1$ time series, $x_t$ is an $r \times 1$ latent factor process, $\epsilon_t \sim \text{WN}(0, \Sigma_\epsilon)$ is a white noise with zero mean and covariance matrix $\Sigma_\epsilon$, and $\epsilon_t$ is uncorrelated with $(z_t, \epsilon_t)$. $D$ is an unknown regression coefficient matrix, and $A$ is an unknown factor loading matrix. This procedure proposed in Chang, Guo and Yao (2015) aims to estimate the unknown regression coefficient matrix $D$, the number of factors $r$ and the factor loading matrix $A$.

### Usage

```r
HDSReg(Y, Z, D = NULL, lag.k = 1, twostep = FALSE)
```
Arguments

\( \mathbf{Y} = \{y_1, \ldots, y_n\}' \), a data matrix with \( n \) rows and \( p \) columns, where \( n \) is the sample size and \( p \) is the dimension of \( y_t \).

\( \mathbf{Z} = \{z_1, \ldots, z_n\}' \), a data matrix representing some observed regressors with \( n \) rows and \( m \) columns, where \( n \) is the sample size and \( m \) is the dimension of \( z_t \).

\( \mathbf{D} \) A \( p \times m \) regression coefficient matrix \( \mathbf{D} \). If \( \mathbf{D} = \text{NULL} \) (the default), our procedure will estimate \( \mathbf{D} \) first and let \( \hat{\mathbf{D}} \) be the estimate of \( \mathbf{D} \). If \( \mathbf{D} \) is given by \( \text{R} \) users, then \( \hat{\mathbf{D}} = \mathbf{D} \).

\( \text{lag.k} \) Time lag \( k_0 \) used to calculate the nonnegative definite matrix \( \hat{\mathbf{M}} \):

\[
\hat{\mathbf{M}} = \sum_{k=1}^{k_0} \hat{\Sigma}_\eta(k)\hat{\Sigma}_\eta(k)',
\]

where \( \hat{\Sigma}_\eta(k) \) is the sample autocovariance of \( \eta_t = y_t - \hat{\mathbf{D}}z_t \) at lag \( k \).

\( \text{twostep} \) Logical. If FALSE (the default), then standard procedures (see factors) will be implemented to estimate \( r \) and \( \mathbf{A} \). If TRUE, then a two step estimation procedure (see factors) will be implemented to estimate \( r \) and \( \mathbf{A} \).

Value

An object of class "HDSReg" is a list containing the following components:

- \( \text{factor.num} \) The estimated number of factors \( \hat{r} \).
- \( \text{reg.coff.mat} \) The estimated \( p \times m \) regression coefficient matrix \( \hat{\mathbf{D}} \) if \( \mathbf{D} \) is not given.
- \( \text{loading.mat} \) The estimated \( p \times m \) factor loading matrix \( \hat{\mathbf{A}} \).

References


See Also

factors.

Examples

\[
\begin{align*}
n & \leftarrow 400 \\
p & \leftarrow 200 \\
m & \leftarrow 2 \\
r & \leftarrow 3 \\
X & \leftarrow \text{mat.or.vec}(n,r) \\
x1 & \leftarrow \text{arima.sim(model=list(ar=c(0.6))),n=n)} \\
x2 & \leftarrow \text{arima.sim(model=list(ar=c(-0.5)),n=n)} \\
x3 & \leftarrow \text{arima.sim(model=list(ar=c(0.3)),n=n)} \\
X & \leftarrow \text{cbind(x1,x2,x3)} \\
X & \leftarrow \text{t(X)}
\end{align*}
\]
```r
Z <- mat.or.vec(m,n)
S1 <- matrix(c(5/8,1/8,1/8,5/8),2,2)
Z[,1] <- c(rnorm(m))
for(i in 2:n){
  Z[,i] <- S1%*%Z[, i-1] + c(rnorm(m))
}
D <- matrix(runif(p*m, -2, 2), ncol=m)
A <- matrix(runif(p*r, -2, 2), ncol=r)
eps <- mat.or.vec(n, p)
eps <- matrix(rnorm(n*p), p, n)
Y <- D %*% Z + A %*% X + eps
Y <- t(Y)
Z <- t(Z)
res1 <- HDSReg(Y,Z,D,lag.k=2)
res2 <- HDSReg(Y,Z,lag.k=2)
```

---

**MartG_test**  
*Testing for martingale difference hypothesis in high dimension*

**Description**

MartG_test() implements a new test proposed in Chang, Jiang and Shao (2021) for the following hypothesis testing problem:

$$H_0 : \{x_t\}_{t=1}^n \text{ is a MDS} \quad \text{versus} \quad H_1 : \{x_t\}_{t=1}^n \text{ is not a MDS},$$

where MDS is the abbreviation of "martingale difference sequence".

**Usage**

```r
MartG_test(
  X,
  lag.k = 2,
  B = 1000,
  type = c("Linear", "Quad"),
  alpha = 0.05,
  kernel.type = c("QS", "Par", "Bart")
)
```

**Arguments**

- `X`  
  \(X = \{x_1, \ldots, x_n\}'\), an \(n \times p\) sample matrix, where \(n\) is the sample size and \(p\) is the dimension of \(x_t\).

- `lag.k`  
  Time lag \(K\), a positive integer, used to calculate the test statistic. Default is `lag.k = 2`.

- `B`  
  Bootstrap times for generating multivariate normal distributed random vectors in calculating the critical value. Default is \(B = 2000\).
type String, a map is chosen by the R users, such as the default option 'Linear' means linear identity map ($\phi(x) = x$). Also including another option 'Quad' (Both linear and quadratic terms $\phi(x) = \{x', (x^2)\}'$). See Section 2.1 in Chang, Jiang and Shao (2021) for more information.

alpha The prescribed significance level. Default is 0.05.

kernel.type String, an option for choosing the symmetric kernel used in the estimation of long-run covariance matrix, for example, 'QS' (Quadratic spectral kernel), 'Par' (Parzen kernel) and 'Bart' (Bartlett kernel), see Andrews (1991) for more information. Default option is kernel.type = 'QS'.

Value
An object of class "MartG_test" is a list containing the following components:

reject Logical value. If TRUE, it means rejecting the null hypothesis, otherwise it means not rejecting the null hypothesis.
p.value Numerical value which represents the p-value of the test.

References

Examples
```
n <- 200
p <- 10
X <- matrix(rnorm(n*p),n,p)
res <- MartG_test(X)
Pvalue <- res$p.value
rej <- res$reject
```

Description
PCA4_Ts() seeks for a contemporaneous linear transformation for a multivariate time series such that the transformed series is segmented into several lower-dimensional subseries:

$$y_t = Ax_t,$$

where $x_t$ is an unobservable $p \times 1$ weakly stationary time series consisting of $q \geq 1$ both contemporaneously and serially uncorrelated subseries. See Chang, Guo and Yao (2018).
PCA4_TS

Usage

PCA4_TS(
  Y,
  lag.k = 5,
  thresh = FALSE,
  tuning.vec = NULL,
  K = 5,
  prewhiten = TRUE,
  permutation = c("max", "fdr"),
  m = NULL,
  beta,
  just4pre = FALSE,
  verbose = FALSE
)

Arguments

Y

Y = \{y_1, \ldots, y_n\}', a data matrix with n rows and p columns, where n is the sample size and p is the dimension of y_t. The procedure will first normalize y_t as \( \hat{\mathbf{V}}^{-1/2}y_t \), where \( \hat{\mathbf{V}} \) is an estimator for covariance of y_t. See details below for the selection of \( \hat{\mathbf{V}}^{-1} \).

lag.k

Time lag \( k_0 \) used to calculate the nonnegative definite matrix \( \hat{\mathbf{W}}_y \):

\[
\hat{\mathbf{W}}_y = \sum_{k=0}^{k_0} \hat{\Sigma}_y(k)\hat{\Sigma}_y(k)' = \mathbf{I}_p + \sum_{k=1}^{k_0} \hat{\Sigma}_y(k)\hat{\Sigma}_y(k)',
\]

where \( \hat{\Sigma}_y(k) \) is the sample autocovariance of \( \hat{\mathbf{V}}^{-1/2}y_t \) at lag \( k \). See (2.5) in Chang, Guo and Yao (2018).

thresh

Logical. If FALSE (the default), no thresholding will be applied to estimate \( \hat{\mathbf{W}}_y \). If TRUE, a thresholding method will be applied first to estimate \( \hat{\mathbf{W}}_y \), see (3.5) in Chang, Guo and Yao (2018).

tuning.vec

The value of the tuning parameter \( \lambda \) in the thresholding level \( u = \lambda \sqrt{n^{-1}\log p} \), where default value is 2. If tuning.vec is a vector, then a cross validation method proposed in Cai and Liu (2011) will be used to choose the best tuning parameter \( \lambda \).

K

The number of folders used in the cross validation for the selection of \( \lambda \), the default is 5. It is required when thresh = TRUE.

prewhiten

Logical. If TRUE (the default), we prewhiten each transformed component series of \( \hat{z}_t \) [See Section 2.2.1 in Chang, Guo and Yao (2018)] by fitting a univariate AR model with the order between 0 and 5 determined by AIC. If FALSE, then prewhiten procedure will not be performed to \( \hat{z}_t \).

permutation

The method of permutation procedure to assign the components of \( \hat{z}_t \) to different groups [See Section 2.2.1 in Chang, Guo and Yao (2018)]. Option is 'max' (Maximum cross correlation method) or 'fdr' (False discovery rate procedure based on multiple tests), default is permutation = 'max'. See Sections 2.2.2 and 2.2.3 in Chang, Guo and Yao (2018) for more information.
A positive constant used in the permutation procedure [See (2.10) in Chang, Guo and Yao (2018)]. If \( m \) is not specified, then default option is \( m = 10 \).

**beta**
The error rate used in the permutation procedure when \( \text{permutation} = 'fdr' \).

**just4pre** Logical. If \( \text{TRUE} \), the procedure outputs \( \hat{z}_t \), otherwise outputs \( \tilde{x}_t \) (the permutated version of \( \hat{z}_t \)).

**verbose** Logical. If \( \text{TRUE} \), the main results of the permutation procedure will be output on the console. Otherwise, the result will not be output.

**Details**
When \( p > n^{1/2} \), the procedure use package **clime** to estimate the precision matrix \( \hat{\Sigma}^{-1/2} \), otherwise uses function **cov()** to estimate \( \hat{\Sigma} \) and calculate its inverse. When \( p > n^{1/2} \), we recommend to use the thresholding method to calculate \( \hat{\Sigma}_y \), see more information in Chang, Guo and Yao (2018).

**Value**
The output of the segment procedure is a list containing the following components:

| B       | The \( p \times p \) transformation matrix such that \( \hat{z}_t = \hat{B}\tilde{y}_t \), where \( \hat{B} = \hat{\Gamma}_y\hat{\Sigma}^{-1/2} \).
| Z       | \( \hat{Z} = \{\hat{z}_1, \ldots, \hat{z}_n\}' \), the transformed series with \( n \) rows and \( p \) columns.

The output of the permutation procedure is a list containing the following components:

| NoGroups | The number of groups.
| No_of_Members | The cardinalities of different groups.
| Groups | The indices of the components in \( \hat{z}_t \) that belongs to a group.

**References**


**Examples**
```r
## Example 1 (Example 5 of Chang Guo and Yao (2018)).
## p=6, x_t consists of 3 independent subseries with 3, 2 and 1 components.

p <- 6; n <- 1500
# Generate x.t
X <- mat.or.vec(p,n)
x <- arima.sim(model=list(ar=c(0.5, 0.3), ma=c(-0.9, 0.3, 1.2,1.3)),
n=n+2,sd=1)
for(i in 1:3) X[,i] <- x[i:(n+i-1)]
x <- arima.sim(model=list(ar=c(0.8,-0.5),ma=c(1,0.8,1.8) ),n=n+1,sd=1)
for(i in 4:5) X[,i] <- x[(i-3):(n+i-4)]
```

\begin{verbatim}
x <- arima.sim(model=list(ar=c(-0.7, -0.5), ma=c(-1, -0.8)), n=n, sd=1)
X[6,] <- x
# Generate y_t
A <- matrix(runif(p*p, -3, 3), ncol=p)
Y <- A%*%X
Y <- t(Y)
res <- PCA4_TS(Y, lag.k=5, permutation = "max")
res1 <- PCA4_TS(Y, lag.k=5, permutation = "fdr", beta=10^(-10))
# The transformed series z_t
Z <- res$Z
# Plot the cross correlogram of z_t and y_t
Y <- data.frame(Y); Z <- data.frame(Z)
names(Y) <- c("Y1", "Y2", "Y3", "Y4", "Y5", "Y6")
names(Z) <- c("Z1", "Z2", "Z3", "Z4", "Z5", "Z6")
# The cross correlogram of y_t shows no block pattern
acfY <- acf(Y)
# The cross correlogram of z_t shows 3-2-1 block pattern
acfZ <- acf(Z)
## Example 2 (Example 6 of Chang Guo and Yao (2018)).
## p=20, x_t consists of 5 independent subseries with 6, 5, 4, 3 and 2 components.
p <- 20; n <- 3000
# Generate x_t
X <- mat.or.vec(p, n)
x <- arima.sim(model=list(ar=c(0.5, 0.3), ma=c(-0.9, 0.3, 1.2, 1.3)), n.start=500, n=n+5, sd=1)
for(i in 1:6) X[,i] <- x[i:(n+i-1)]
x <- arima.sim(model=list(ar=c(-0.4, 0.5), ma=c(1, 0.8, 1.5, 1.8)), n.start=500, n=n+4, sd=1)
for(i in 7:11) X[,i] <- x[i-6:(n+i-7)]
x <- arima.sim(model=list(ar=c(0.85, -0.3), ma=c(1, 0.5, 1.2)), n.start=500, n=n+3, sd=1)
for(i in 12:15) X[,i] <- x[i-11:(n+i-12)]
x <- arima.sim(model=list(ar=c(0.8, -0.5), ma=c(1, 0.8, 1.8)), n.start=500, n=n+2, sd=1)
for(i in 16:18) X[,i] <- x[i-15:(n+i-16)]
x <- arima.sim(model=list(ar=c(-0.7, -0.5), ma=c(-1, -0.8)), n.start=500, n=n+1, sd=1)
for(i in 19:20) X[,i] <- x[i-18:(n+i-19)]
# Generate y_t
A <- matrix(runif(p*p, -3, 3), ncol=p)
Y <- A%*%X
Y <- t(Y)
res <- PCA4_TS(Y, lag.k=5, permutation = "max")
res1 <- PCA4_TS(Y, lag.k=5, permutation = "fdr", beta=10^(-200))
# The transformed series z_t
Z <- res$Z
# Plot the cross correlogram of x_t and y_t
Y <- data.frame(Y); Z <- data.frame(Z)
namesY <- NULL; namesZ <- NULL
for(i in 1:p)
{
 namesY <- c(namesY, paste0("Y", i))
 namesZ <- c(namesZ, paste0("Z", i))
}
names(Y) <- namesY; names(Z) <- namesZ
# The cross correlogram of y_t shows no block pattern
\end{verbatim}
acfY <- acf(Y, plot=FALSE)
plot(acfY, max.mfrow=6, xlab='\text{Var}', ylab='\text{Var}',
     mar=c(1.8,1.3,1.6,0.5), oma=c(1.1,2,1.2,1), mgp=c(0.8,0.4,0),cex.main=1)
# The cross correlogram of \text{Z}_t shows 6-5-4-3-2 block pattern
acfZ <- acf(Z, plot=FALSE)
plot(acfZ, max.mfrow=6, xlab='\text{Var}', ylab='\text{Var}',
     mar=c(1.8,1.3,1.6,0.5), oma=c(1.1,2,1.2,1), mgp=c(0.8,0.4,0),cex.main=1)
# Identify the permutation mechanism
permutation <- res
permutation$Groups

ur.test

Testing for unit roots based on sample autocovariances

Description

The test proposed in Chang, Cheng and Yao (2021) for the following hypothesis testing problems:

\[ H_0 : Y_t \sim I(0) \] versus \[ H_1 : Y_t \sim I(d) \] for some integer \( d \geq 2 \).

Usage

ur.test(Y, lagk.vec = lagk.vec, con_vec = con_vec, alpha = alpha)

Arguments

- **Y** \( Y = \{y_1, \ldots, y_n\} \), the observations of a univariate time series used for the test.
- **lagk.vec** Time lag \( K_0 \) used to calculate the test statistic, see Section 2.1 in Chang, Cheng and Yao (2021). It can be a vector containing more than one time lag. If it is a vector, the procedure will output all the test results based on the different \( K_0 \) in the vector lagk.vec. If lagk.vec is missing, the default value we choose lagk.vec=c(0,1,2,3,4).
- **con_vec** Constant \( c_\kappa \), see (5) in Chang, Cheng and Yao (2021). It also can be a vector. If missing, the default value we use 0.55.
- **alpha** The prescribed significance level. Default is 0.05.

Value

A dataframe containing the following components:

- **result** '1' means we reject the null hypothesis and '0' means we do not reject the null hypothesis.

References

Examples

N=100
Y=arima.sim(list(ar=c(0.9)), n = 2*N, sd=sqrt(1))
con_vec=c(0.45,0.55,0.65)
lagk.vec=c(0,1,2)
ur.test(Y,lagk.vec=lagk.vec, con_vec=con_vec,alpha=0.05)
ur.test(Y,alpha=0.05)

WN_test

Testing for white noise hypothesis in high dimension

Description

WN_test() is the test proposed in Chang, Yao and Zhou (2017) for the following hypothesis testing problems:

\[ H_0 : \{ x_t \}_{t=1}^n \text{ is white noise versus } H_1 : \{ x_t \}_{t=1}^n \text{ is not white noise.} \]

Usage

WN_test(
X, 
lag.k = 2, 
B = 2000, 
kernel.type = c("QS", "Par", "Bart"), 
pre = FALSE, 
alpha = 0.05, 
k0 = 5, 
thresh = FALSE, 
tuning.vec = NULL
)

Arguments

X \quad \text{X} = \{ x_1, \ldots, x_n \}^\prime, \text{ an } n \times p \text{ sample matrix, where } n \text{ is the sample size and } p \text{ is the dimension of } x_t.

lag.k \quad \text{Time lag } K, \text{ a positive integer, used to calculate the test statistic [See (4) in Chang, Yao and Zhou (2017)]. Default is } lag.k = 2.

B \quad \text{Bootstrap times for generating multivariate normal distributed random vectors in calculating the critical value. Default is } B = 2000.

kernel.type \quad \text{String, an option for choosing the symmetric kernel used in the estimation of long-run covariance matrix, for example, } 'QS' \text{ (Quadratic spectral kernel), 'Par' (Parzen kernel) and 'Bart' (Bartlett kernel), see Andrews (1991) for more information. Default option is } kernel.type = 'QS'.

Logical value which determines whether to performs preprocessing procedure on data matrix $X$ or not, see Remark 1 in Chang, Yao and Zhou (2017) for more information. If TRUE, then the segment procedure will be performed to data $X$ first. The three additional options including thresh, tuning.vec and cv.num are the same as those in PCA4_TS.

The prescribed significance level. Default is 0.05.

A positive integer specified to calculate $\hat{W}_y$. See parameter lag.k in PCA4_TS for more information.

Logical. It determines whether to perform the threshold method to estimate $\hat{W}_y$ or not. See parameter thresh in PCA4_TS for more information.

The value of thresholding tuning parameter $\lambda$. See parameter tuning.vec in PCA4_TS for more information.

An object of class "WN_test" is a list containing the following components:

- **reject**: Logical value. If TRUE, it means rejecting the null hypothesis, otherwise it means not rejecting the null hypothesis.
- **p.value**: Numerical value which represents the p-value of the test based on the observed data $\{x_t\}_{t=1}^n$.

References


See Also

PCA4_TS

Examples

```r
n <- 200
p <- 10
X <- matrix(rnorm(n*p),n,p)
res <- WN_test(X)
Pvalue <- res$p.value
rej <- res.reject
```
Index

factors, 2, 4
HDSReg, 3
MartG_test, 5
PCA4_TS, 6, 12
ur.test, 10
WN_test, 11