

# Package ‘HDTSA’

September 3, 2024

**Type** Package

**Title** High Dimensional Time Series Analysis Tools

**Version** 1.0.4

**Date** 2024-09-03

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

**Maintainer** Chen Lin <lincchen@mail.swufe.edu.cn>

**Description** Procedures for high-dimensional time series analysis including factor analysis proposed by Lam and Yao (2012) <[doi:10.1214/12-AOS970](https://doi.org/10.1214/12-AOS970)> and Chang, Guo and Yao (2015) <[doi:10.1016/j.jeconom.2015.03.024](https://doi.org/10.1016/j.jeconom.2015.03.024)>, martingale difference test proposed by Chang, Jiang and Shao (2022) <[doi:10.1016/j.jeconom.2022.09.001](https://doi.org/10.1016/j.jeconom.2022.09.001)> in press, principal component analysis proposed by Chang, Guo and Yao (2018) <[doi:10.1214/17-AOS1613](https://doi.org/10.1214/17-AOS1613)>, identifying cointegration proposed by Zhang, Robinson and Yao (2019) <[doi:10.1080/01621459.2018.1458620](https://doi.org/10.1080/01621459.2018.1458620)>, unit root test proposed by Chang, Cheng and Yao (2021) <[doi:10.1093/biomet/asab034](https://doi.org/10.1093/biomet/asab034)>, white noise test proposed by Chang, Yao and Zhou (2017) <[doi:10.1093/biomet/asw066](https://doi.org/10.1093/biomet/asw066)>, CP-decomposition for high-dimensional matrix time series proposed by Chang, He, Yang and Yao (2023) <[doi:10.1093/jrsssb/qkac011](https://doi.org/10.1093/jrsssb/qkac011)> and Chang, Du, Huang and Yao (2024+), and Statistical inference for high-dimensional spectral density matrix proposed by Chang, Jiang, McElroy and Shao (2023) <[doi:10.48550/arXiv.2212.13686](https://doi.org/10.48550/arXiv.2212.13686)>.

**License** GPL-3

**Depends** R (>= 3.5.0)

**Imports** stats, Rcpp, clime, sandwich, methods, MASS, geigen, jointDiag

**LinkingTo** Rcpp, RcppEigen

**Suggests** knitr

**NeedsCompilation** yes

**RoxygenNote** 7.3.1

**Encoding** UTF-8

**URL** <https://github.com/Linc2021/HDTSA>

**BugReports** <https://github.com/Linc2021/HDTSA/issues>

Repository CRAN

Date/Publication 2024-09-03 09:00:01 UTC

## Contents

Coint . . . . .	2
CP_MTS . . . . .	4
DGP.CP . . . . .	5
Factors . . . . .	7
HDSReg . . . . .	8
MartG_test . . . . .	10
PCA_TS . . . . .	11
SpecMulTest . . . . .	15
SpecTest . . . . .	16
UR_test . . . . .	18
WN_test . . . . .	19

<b>Index</b>	<b>21</b>
--------------	-----------

---

Coint	<i>Identifying cointegration rank of given time series</i>
-------	--

---

## Description

Coint seeks for a contemporaneous linear transformation for a multivariate time series such that we can identifying cointegration rank from the transformed series.

## Usage

```
Coint(
  Y,
  lag.k = 5,
  type = c("acf", "pptest", "Chang", "all"),
  c0 = 0.3,
  m = 20,
  alpha = 0.01
)
```

## Arguments

Y	$\mathbf{Y} = \{\mathbf{y}_1, \dots, \mathbf{y}_n\}'$ , a data matrix with $n$ rows and $p$ columns, where $n$ is the sample size and $p$ is the dimension of $\mathbf{y}_t$ .
lag.k	Time lag $k_0$ used to calculate the nonnegative definite matrix $\widehat{\mathbf{W}}_y$ :

$$\widehat{\mathbf{W}}_y = \sum_{k=0}^{k_0} \widehat{\Sigma}_y(k) \widehat{\Sigma}_y(k)'$$

where  $\widehat{\Sigma}_y(k)$  is the sample autocovariance of  $\widehat{\mathbf{y}}_t$  at lag  $k$ .

type	The method of identifying cointegration rank after segment procedure. Option is 'acf', 'all', 'chang' or 'pptest', the latter two methods use the unit-root test method to identify the cointegration rank, and the option type = 'all' means use all three methods to identify the cointegration rank. Default is type = 'acf'. See Sections 2.3 in Zhang, Robinson and Yao (2019) for more information.
c0	The prescribed constant for identifying cointegration rank using "acf" method. Default is 0.3.[See (2.3) in Zhang, Robinson and Yao (2019)].
m	The prescribed constant for identifying cointegration rank using "acf" method. Default is 20. [See (2.3) in Zhang, Robinson and Yao (2019)].
alpha	The prescribed significance level for identifying cointegration rank using "pptest", "chang" method. Default is 0.01. [See (2.3) in Zhang, Robinson and Yao (2019)].

### Value

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

Z	The transformed series with $n$ rows and $p$ columns.
coint_rank	A $1 \times 1$ matrix representing the cointegration rank. If type = 'all', then return a $1 \times 3$ matrix representing the cointegration rank of all three methods.
lag.k	a prescribed positive integer which means the time lags used to calculate the statistic.
method	a character string indicating which method was performed.

### References

Zhang, R., Robinson, P. & Yao, Q. (2019). *Identifying Cointegration by Eigenanalysis*. Journal of the American Statistical Association, Vol. 114, pp. 916–927

### Examples

```
p <- 10
n <- 1000
r <- 3
d <- 1
X <- mat.or.vec(p, n)
X[1,] <- arima.sim(n=d, model = list(order=c(0, d, 0)))
for(i in 2:3)X[i,] <- rnorm(n)
for(i in 4:(r+1)) X[i, ] <- arima.sim(model = list(ar = 0.5), n)
for(i in (r+2):p) X[i, ] <- arima.sim(n = (n-d), model = list(order=c(1, d, 1), ar=0.6, ma=0.8))
M1 <- matrix(c(1, 1, 0, 1/2, 0, 1, 0, 1, 0), ncol = 3, byrow = TRUE)
A <- matrix(runif(p*p, -3, 3), ncol = p)
A[1:3,1:3] <- M1
Y <- t(A%%X)
Coint(Y, type = "all")
```

**Description**

CP\_MTS() deals with CP-decomposition for high-dimensional matrix time series proposed in Chang et al. (2023):

$$\mathbf{Y}_t = \mathbf{A}\mathbf{X}_t\mathbf{B}' + \boldsymbol{\epsilon}_t,$$

where  $\mathbf{X}_t = \text{diag}(x_{t,1}, \dots, x_{t,d})$  is an  $d \times d$  latent process,  $\mathbf{A}$  and  $\mathbf{B}$  are, respectively,  $p \times d$  and  $q \times d$  unknown constant matrix, and  $\boldsymbol{\epsilon}_t$  is a  $p \times q$  matrix white noise process. This function aims to estimate the rank  $d$  and the coefficient matrices  $\mathbf{A}$  and  $\mathbf{B}$ .

**Usage**

```
CP_MTS(  
  Y,  
  xi = NULL,  
  Rank = NULL,  
  lag.k = 15,  
  lag.ktilde = 10,  
  method = c("CP.Direct", "CP.Refined", "CP.Unified")  
)
```

**Arguments**

**Y** A  $n \times p \times q$  data array, where  $n$  is the sample size and  $(p, q)$  is the dimension of  $\mathbf{Y}_t$ .

**xi** A  $n \times 1$  vector. If NULL (the default), then a PCA-based  $\xi_t$  is used [See Section 5.1 in Chang et al. (2023)] to calculate the sample auto-covariance matrix  $\widehat{\boldsymbol{\Sigma}}_{\mathbf{Y},\xi}(k)$ .

**Rank** A list of the rank  $d, d_1$  and  $d_2$ . Default to NULL.

**lag.k** Integer. Time lag  $K$  is only used in CP.Refined and CP.Unified to calculate the nonnegative definite matrices  $\widehat{\mathbf{M}}_1$  and  $\widehat{\mathbf{M}}_2$ :

$$\widehat{\mathbf{M}}_1 = \sum_{k=1}^K \widehat{\boldsymbol{\Sigma}}_{\mathbf{Y},\xi}(k) \widehat{\boldsymbol{\Sigma}}_{\mathbf{Y},\xi}(k)',$$

$$\widehat{\mathbf{M}}_2 = \sum_{k=1}^K \widehat{\boldsymbol{\Sigma}}_{\mathbf{Y},\xi}(k)' \widehat{\boldsymbol{\Sigma}}_{\mathbf{Y},\xi}(k),$$

where  $\widehat{\boldsymbol{\Sigma}}_{\mathbf{Y},\xi}(k)$  is the sample auto-covariance of  $\mathbf{Y}_t$  and  $\xi_t$  at lag  $k$ .

**lag.ktilde** Integer. Time lag  $\tilde{K}$  is only used in CP.Unified to calculate the nonnegative definite matrix  $\widehat{\mathbf{M}}$ :

$$\widehat{\mathbf{M}} = \sum_{k=1}^{\tilde{K}} \widehat{\boldsymbol{\Sigma}}_{\tilde{\mathbf{Z}}}(k) \widehat{\boldsymbol{\Sigma}}_{\tilde{\mathbf{Z}}}(k)'$$

method Method to use: CP.Direct and CP.Refined, Chang et al.(2023)'s direct and refined estimators; CP.Unified, Chang et al.(2024+)'s unified estimation procedure.

### Value

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

A	The estimated $p \times d$ left loading matrix $\hat{\mathbf{A}}$ .
B	The estimated $q \times d$ right loading matrix $\hat{\mathbf{B}}$ .
f	The estimated latent process $(\hat{x}_{1,t}, \dots, \hat{x}_{d,t})$ .
Rank	The estimated rank $(\hat{d}_1, \hat{d}_2, \hat{d})$ of the matrix CP-factor model.

### References

Chang, J., He, J., Yang, L. and Yao, Q.(2023). *Modelling matrix time series via a tensor CP-decomposition*. Journal of the Royal Statistical Society Series B: Statistical Methodology, Vol. 85(1), pp.127–148.

Chang, J., Du, Y., Huang, G. and Yao, Q.(2024+). *On the Identification and Unified Estimation Procedure for the Matrix CP-factor Model*, Working paper.

### Examples

```
p = 10
q = 10
n = 400
d = d1 = d2 = 3
data <- DGP.CP(n,p,q,d1,d2,d)
Y = data$Y
res1 <- CP_MTS(Y,method = "CP.Direct")
res2 <- CP_MTS(Y,method = "CP.Refined")
res3 <- CP_MTS(Y,method = "CP.Unified")
```

---

DGP.CP

*Data generate process of matrix CP-factor model*

---

### Description

DGP.CP() function generate the matrix time series described in Chang et al. (2023):

$$\mathbf{Y}_t = \mathbf{A}\mathbf{X}_t\mathbf{B}' + \boldsymbol{\epsilon}_t,$$

where  $\mathbf{X}_t = \text{diag}(x_{t,1}, \dots, x_{t,d})$  is an  $d \times d$  latent process,  $\mathbf{A}$  and  $\mathbf{B}$  are , respectively,  $p \times d$  and  $q \times d$  unknown constant matrix, and  $\boldsymbol{\epsilon}_t$  is a  $p \times q$  matrix white noise process.

### Usage

DGP.CP(n, p, q, d1, d2, d)

**Arguments**

n	Integer. Sample size of $\mathbf{Y}_t$ , $t = 1, \dots, n$ .
p	Integer. Number of rows of $\mathbf{Y}_t$ .
q	Integer. Number of columns of $\mathbf{Y}_t$ .
d1	Integer. Rank of $\mathbf{A}$ .
d2	Integer. Rank of $\mathbf{B}$ .
d	Integer. Number of columns of $\mathbf{A}$ and $\mathbf{B}$ .

**Value**

A list containing the following components:

Y	A $n \times p \times q$ data array of $\mathbf{Y}_t$ .
S	A $n \times p \times q$ data array of $\mathbf{S}_t = \mathbf{A}\mathbf{X}_t\mathbf{B}'$ .
A	A $p \times d$ coefficient matrix.
B	A $q \times d$ coefficient matrix.
X	A $n \times d \times d$ data array of $\mathbf{X}_t$ .
P	A $p \times d_1$ orthogonal matrix such that $\mathbf{A} = \mathbf{P}\mathbf{U}$ .
Q	A $q \times d_2$ orthogonal matrix such that $\mathbf{B} = \mathbf{Q}\mathbf{V}$ .
U	A $d_1 \times d$ matrix such that $\mathbf{A} = \mathbf{P}\mathbf{U}$ .
V	A $d_2 \times d$ matrix such that $\mathbf{B} = \mathbf{Q}\mathbf{V}$ .
W	A $d_1 d_2 \times d$ matrix such that $\mathbf{W} = (\mathbf{v}_1 \otimes \mathbf{u}_1, \dots, \mathbf{v}_d \otimes \mathbf{u}_d)$ .
Ws	A $d_1 d_2 \times d$ matrix. An orthogonal basis of $\mathbf{W}$ .
Xmat	A $n \times d$ data matrix of $\text{diag}(\mathbf{X}_t)$ .
Smat	A $n \times pq$ data matrix of $\text{vec}(\mathbf{S}_t)$ .

**References**

Chang, J., He, J., Yang, L. and Yao, Q.(2023). *Modelling matrix time series via a tensor CP-decomposition*. Journal of the Royal Statistical Society Series B: Statistical Methodology, Vol. 85(1), pp.127–148.

**See Also**

[CP\\_MTS](#).

**Examples**

```
p = 10
q = 10
n = 400
d = d1 = d2 = 3
data <- DGP.CP(n,p,q,d1,d2,d)
Y = data$Y
```

**Description**

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

$$\mathbf{y}_t = \mathbf{A}\mathbf{x}_t + \boldsymbol{\epsilon}_t,$$

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

**Usage**

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

**Arguments**

**Y**  $\mathbf{Y} = \{\mathbf{y}_1, \dots, \mathbf{y}_n\}'$ , a data matrix with  $n$  rows and  $p$  columns, where  $n$  is the sample size and  $p$  is the dimension of  $\mathbf{y}_t$ .

**lag.k** Time lag  $k_0$  used to calculate the nonnegative definite matrix  $\widehat{\mathbf{M}}$ :

$$\widehat{\mathbf{M}} = \sum_{k=1}^{k_0} \widehat{\boldsymbol{\Sigma}}_y(k) \widehat{\boldsymbol{\Sigma}}_y(k)',$$

where  $\widehat{\boldsymbol{\Sigma}}_y(k)$  is the sample autocovariance of  $\mathbf{y}_t$  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  $\mathbf{A}$  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  $\mathbf{A}$ .

**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 $\widehat{\mathbf{A}}$ .
lag.k	the time lag used in function.
method	a character string indicating what method was performed.

**References**

Lam, C. & Yao, Q. (2012). *Factor modelling for high-dimensional time series: Inference for the number of factors*, The Annals of Statistics, Vol. 40, pp. 694–726.

## Examples

```
## 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
```

---

HDSReg

*High dimensional stochastic regression with latent factors*


---

## 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:

$$\mathbf{y}_t = \mathbf{D}\mathbf{z}_t + \mathbf{A}\mathbf{x}_t + \boldsymbol{\epsilon}_t,$$

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

## Usage

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

## Arguments

- Y  $\mathbf{Y} = \{\mathbf{y}_1, \dots, \mathbf{y}_n\}'$ , a data matrix with  $n$  rows and  $p$  columns, where  $n$  is the sample size and  $p$  is the dimension of  $\mathbf{y}_t$ .
- Z  $\mathbf{Z} = \{\mathbf{z}_1, \dots, \mathbf{z}_n\}'$ , 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  $\mathbf{z}_t$ .
- D A  $p \times m$  regression coefficient matrix  $\tilde{\mathbf{D}}$ . If  $\mathbf{D} = \text{NULL}$  (the default), our procedure will estimate  $\mathbf{D}$  first and let  $\tilde{\mathbf{D}}$  be the estimate of  $\mathbf{D}$ . If  $\mathbf{D}$  is given by R users, then  $\tilde{\mathbf{D}} = \mathbf{D}$ .



lag.k Time lag  $k_0$  used to calculate the nonnegative definite matrix  $\widehat{\mathbf{M}}$ :

$$\widehat{\mathbf{M}} = \sum_{k=1}^{k_0} \widehat{\boldsymbol{\Sigma}}_{\eta}(k) \widehat{\boldsymbol{\Sigma}}_{\eta}(k)',$$

where  $\widehat{\boldsymbol{\Sigma}}_{\eta}(k)$  is the sample autocovariance of  $\boldsymbol{\eta}_t = \mathbf{y}_t - \widetilde{\mathbf{D}}\mathbf{z}_t$  at lag  $k$ .

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 "factors" is a list containing the following components:

factor_num	The estimated number of factors $\hat{r}$ .
reg.coff.mat	The estimated $p \times m$ regression coefficient matrix $\widetilde{\mathbf{D}}$ if D is not given.
loading.mat	The estimated $p \times m$ factor loading matrix $\widehat{\mathbf{A}}$ .
lag.k	the time lag used in function.
method	a character string indicating what method was performed.

### References

Chang, J., Guo, B. & Yao, Q. (2015). *High dimensional stochastic regression with latent factors, endogeneity and nonlinearity*, Journal of Econometrics, Vol. 189, pp. 297–312.

### See Also

[Factors](#).

### Examples

```
n <- 400
p <- 200
m <- 2
r <- 3
X <- mat.or.vec(n,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)
X <- cbind(x1,x2,x3)
X <- t(X)

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 c(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:

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

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

### Usage

```

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

```

### Arguments

X	$\mathbf{X} = \{\mathbf{x}_1, \dots, \mathbf{x}_n\}'$ , an $n \times p$ sample matrix, where $n$ is the sample size and $p$ is the dimension of $\mathbf{x}_t$ .
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 = 1000.
type	String, a map is chosen by the R users, such as the default option is 'Linear' means linear identity map ( $\phi(\mathbf{x}) = \mathbf{x}$ ). Also including another option 'Quad' (Both linear and quadratic terms $\phi(\mathbf{x}) = \{\mathbf{x}', (\mathbf{x}^2)'\}'$ ). Also the users can choose set the map themselves, use for example expression(X, X^2), quote(X, X^2), parse(X, X^2), substitute(X, X^2) or just map without function (such as cbind(X, X^2)) to set their own map. 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 "hdtstest" is a list containing the following components:

.	
statistic	The value of the test statistic.
p.value	Numerical value which represents the p-value of the test.
lag.k	The time lag used in function.
method	A character string indicating what method was performed.
type	A character string which map used on data matrix X.
kernel.type	A character string indicating what kernel method was performed.

### References

Chang, J., Jiang, Q. & Shao, X. (2022). *Testing the martingale difference hypothesis in high dimension*. Journal of Econometrics, in press

### Examples

```
n <- 200
p <- 10
X <- matrix(rnorm(n*p),n,p)
res <- MartG_test(X, type="Linear")
res <- MartG_test(X, type=cbind(X, X^2)) #the same as Linear type
res <- MartG_test(X, type=quote(cbind(X, X^2))) # expr using quote
res <- MartG_test(X, type=substitute(cbind(X, X^2))) # expr using substitute
res <- MartG_test(X, type=expression(cbind(X, X^2))) # expr using expression
res <- MartG_test(X, type=parse(text="cbind(X, X^2)")) # expr using parse
map_fun <- function(X) {X <- cbind(X,X^2); X}
res <- MartG_test(X, type=map_fun)
Pvalue <- res$p.value
rej <- res$reject
```

### Description

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

$$\mathbf{y}_t = \mathbf{A}\mathbf{x}_t,$$

where  $\mathbf{x}_t$  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).

### Usage

```
PCA_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**  $\mathbf{Y} = \{\mathbf{y}_1, \dots, \mathbf{y}_n\}'$ , a data matrix with  $n$  rows and  $p$  columns, where  $n$  is the sample size and  $p$  is the dimension of  $\mathbf{y}_t$ . The procedure will first normalize  $\mathbf{y}_t$  as  $\hat{\mathbf{V}}^{-1/2}\mathbf{y}_t$ , where  $\hat{\mathbf{V}}$  is an estimator for covariance of  $\mathbf{y}_t$ . See details below for the selection of  $\hat{\mathbf{V}}^{-1}$ .

**lag.k** Time lag  $k_0$  used to calculate the nonnegative definite matrix  $\widehat{\mathbf{W}}_y$ :

$$\widehat{\mathbf{W}}_y = \sum_{k=0}^{k_0} \widehat{\boldsymbol{\Sigma}}_y(k) \widehat{\boldsymbol{\Sigma}}_y(k)' = \mathbf{I}_p + \sum_{k=1}^{k_0} \widehat{\boldsymbol{\Sigma}}_y(k) \widehat{\boldsymbol{\Sigma}}_y(k)',$$

where  $\widehat{\boldsymbol{\Sigma}}_y(k)$  is the sample autocovariance of  $\hat{\mathbf{V}}^{-1/2}\mathbf{y}_t$  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  $\widehat{\mathbf{W}}_y$ . If TRUE, a thresholding method will be applied first to estimate  $\widehat{\mathbf{W}}_y$ , 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 $\hat{\mathbf{z}}_t$ [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 $\hat{\mathbf{z}}_t$ .
permutation	The method of permutation procedure to assign the components of $\hat{\mathbf{z}}_t$ 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.
m	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 permutation = 'fdr'.
just4pre	Logical. If TRUE, the procedure outputs $\hat{\mathbf{z}}_t$ , otherwise outputs $\hat{\mathbf{x}}_t$ (the permuted version of $\hat{\mathbf{z}}_t$ ).
verbose	Logical. If 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  $\hat{\mathbf{V}}^{-1}$ , otherwise uses function `cov()` to estimate  $\hat{\mathbf{V}}$  and calculate its inverse. When  $p > n^{1/2}$ , we recommend to use the thresholding method to calculate  $\hat{\mathbf{W}}_y$ , 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 $\hat{\mathbf{z}}_t = \hat{\mathbf{B}}\mathbf{y}_t$ , where $\hat{\mathbf{B}} = \hat{\mathbf{\Gamma}}_y \hat{\mathbf{V}}^{-1/2}$ .
Z	$\hat{\mathbf{Z}} = \{\hat{\mathbf{z}}_1, \dots, \hat{\mathbf{z}}_n\}'$ , the transformed series with $n$ rows and $p$ columns.

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

NoGroups	number of groups with at least two components series.
No_of_Members	The cardinalities of different groups.
Groups	The indices of the components in $\hat{\mathbf{z}}_t$ that belongs to a group.
method	a character string indicating what method was performed.

### References

- Chang, J., Guo, B. & Yao, Q. (2018). *Principal component analysis for second-order stationary vector time series*, The Annals of Statistics, Vol. 46, pp. 2094–2124.
- Cai, T. & Liu, W. (2011). *Adaptive thresholding for sparse covariance matrix estimation*, Journal of the American Statistical Association, Vol. 106, pp. 672–684.
- Cai, T., Liu, W., & Luo, X. (2011). *A constrained l1 minimization approach for sparse precision matrix estimation*, Journal of the American Statistical Association, Vol. 106, pp. 594–607.

## Examples

```

## 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)]
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 <- PCA_TS(Y, lag.k=5,permutation = "max")
res1=PCA_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 <- PCA_TS(Y, lag.k=5,permutation = "max")

```

```

res1 <- PCA_TS(Y, lag.k=5,permutation = "fdr",beta=10^(-200))
# The transformed series z_t
Z <- res$Z
# Plot the cross correlogram of x_t and y_t
Y <- data.frame(Y);Z <- data.frame(Z)
namesY=NULL;namesZ=NULL
for(i in 1:p)
{
  namesY <- c(namesY,paste0("Y",i))
  namesZ <- c(namesZ,paste0("Z",i))
}
names(Y) <- namesY;names(Z) <- namesZ
# The cross correlogram of y_t shows no block pattern
acfY <- acf(Y, plot=FALSE)
plot(acfY, max.mfrow=6, xlab='', ylab='', mar=c(1.8,1.3,1.6,0.5),
      oma=c(1,1.2,1.2,1), mgp=c(0.8,0.4,0),cex.main=1)
# The cross correlogram of z_t shows 6-5-4-3-2 block pattern
acfZ <- acf(Z, plot=FALSE)
plot(acfZ, max.mfrow=6, xlab='', ylab='', mar=c(1.8,1.3,1.6,0.5),
      oma=c(1,1.2,1.2,1), mgp=c(0.8,0.4,0),cex.main=1)
# Identify the permutation mechanism
permutation <- res
permutation$Groups

```

---

SpecMulTest

*Statistical inference for high-dimensional spectral density matrix*


---

## Description

SpecMulTest() implements a new multiple test proposed in Chang, Jiang, McElroy and Shao (2023) for the  $Q$  hypothesis testing problems:

$$H_{0,q} : f_{i,j}(\omega) = 0 \text{ for any } (i, j) \in \mathcal{I}^{(q)} \text{ and } \omega \in \mathcal{J}^{(q)} \text{ versus } H_{1,q} : H_{0,q} \text{ is not true.}$$

for  $q \in \{1, \dots, Q\}$ .

## Usage

```
SpecMulTest(Q, PVal, alpha = 0.05, seq_len = 0.01)
```

## Arguments

Q	Number of the hypothesis tests.
PVal	P-values for the $Q$ hypothesis tests, a $Q$ vector.
alpha	The prescribed significance level. Default is 0.05.
seq_len	Length used to take discrete points between 0 and $\sqrt{(2 \times \log(Q) - 2 \times \log(\log(Q)))}$ . Default is 0.01.

**Value**

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

**MultiTest** Logical vector with length Q. If the element is TRUE, it means rejecting the corresponding sub-null hypothesis, otherwise it means not rejecting the corresponding sub-null hypothesis.

**References**

Chang, J., Jiang, Q., McElroy, T. & Shao, X. (2023). *Statistical inference for high-dimensional spectral density matrix*.

**Examples**

```
n <- 200
p <- 10
flag_c <- 0.8
B <- 1000
burn <- 1000
z.sim <- matrix(rnorm((n+burn)*p), p, n+burn)
phi.mat <- 0.4*diag(p)
x.sim <- phi.mat %*% z.sim[, (burn+1):(burn+n)]
x <- x.sim - rowMeans(x.sim)
Q <- 4
ISET <- list()
ISET[[1]] <- matrix(c(1,2), ncol=2)
ISET[[2]] <- matrix(c(1,3), ncol=2)
ISET[[3]] <- matrix(c(1,4), ncol=2)
ISET[[4]] <- matrix(c(1,5), ncol=2)
JSET <- as.list(2*pi*seq(0,3)/4 - pi)
PVal <- rep(NA, Q)
for (q in 1:Q) {
  cross.indices <- ISET[[q]]
  J.set <- JSET[[q]]
  temp.q <- SpecTest(t(x), J.set, cross.indices, B, flag_c)
  PVal[q] <- temp.q$p.value
} # Q
res <- SpecMulTest(Q, PVal)
res
```

---

SpecTest

*Statistical inference for high-dimensional spectral density matrix*

---

**Description**

SpecTest() implements a new global test proposed in Chang, Jiang, McElroy and Shao (2023) for the following hypothesis testing problem:

$$H_0 : f_{i,j}(\omega) = 0 \text{ for any } (i, j) \in \mathcal{I} \text{ and } \omega \in \mathcal{J} \text{ versus } H_1 : H_0 \text{ is not true.}$$



**Usage**

```
SpecTest(X, J.set, cross.indices, B = 1000, flag_c = 0.8)
```

**Arguments**

X	$\mathbf{X} = \{\mathbf{x}_1, \dots, \mathbf{x}_n\}$ , a $n \times p$ sample matrix, where $n$ is the sample size and $p$ is the dimension of $\mathbf{x}_t$ .
J.set	Set $\mathcal{J}$ for frequencies, a vector, used to calculate the test statistic.
cross.indices	Set $\mathcal{I}$ for $(i, j)$ , a matrix with 2 columns, used to calculate the test statistic.
B	Bootstrap times for generating multivariate normal distributed random vectors in calculating the critical value. Default is $B = 2000$ .
flag_c	Bandwidth $c$ of the flat-top kernel for estimating $f_{i,j}(\omega)$ , where $c \in (0, 1]$ . Default is $\text{flag\_c} = 0.8$ .

**Value**

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

Stat	Numerical value which represents the value of test statistic.
pval	Numerical value which represents the p-value of the test.
cri95	Numerical value which represents the critical value of the test at the significance level 0.05.
method	A character string indicating what method was performed.

**References**

Chang, J., Jiang, Q., McElroy, T. & Shao, X. (2023). *Statistical inference for high-dimensional spectral density matrix*.

**Examples**

```
n <- 200
p <- 10
flag_c <- 0.8
B <- 1000
burn <- 1000
z.sim <- matrix(rnorm((n+burn)*p),p,n+burn)
phi.mat <- 0.4*diag(p)
x.sim <- phi.mat %*% z.sim[, (burn+1):(burn+n)]
x <- x.sim - rowMeans(x.sim)
cross.indices <- matrix(c(1,2), ncol=2)
J.set <- 2*pi*seq(0,3)/4 - pi
res <- SpecTest(t(x), J.set, cross.indices, B, flag_c)
Stat <- res$Stat
Pvalue <- res$p.value
CriVal <- res$cri95
```

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:

$$H_0 : Y_t \sim I(0) \text{ versus } H_1 : Y_t \sim I(d) \text{ for some integer } d \geq 2.$$

**Usage**

```
UR_test(Y, lagk.vec = NULL, con_vec = NULL, alpha = 0.05)
```

**Arguments**

Y	$Y = \{y_1, \dots, y_n\}$ , 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**

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

statistic	A vector which represents the value of the test statistic, the length of this vector is the same as lag.vec
reject	A data matrix containing result with different arguments, each column represents the results of different $c_\kappa$ calculations, and each column is a vector representing the results of different time lag K0 calculations. '1' means we reject the null hypothesis and '0' means we do not reject the null hypothesis
lag.vec	The time lag used in function.
method	A character string indicating what method was performed.
type	A character string which map used on data matrix X.

**References**

Chang, J., Cheng, G. & Yao, Q. (2021). *Testing for unit roots based on sample autocovariances*. Available at <https://arxiv.org/abs/2006.07551>

**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:

$$H_0 : \{\mathbf{x}_t\}_{t=1}^n \text{ is white noise versus } H_1 : \{\mathbf{x}_t\}_{t=1}^n \text{ is not white noise.}$$

**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	$\mathbf{X} = \{\mathbf{x}_1, \dots, \mathbf{x}_n\}'$ , an $n \times p$ sample matrix, where $n$ is the sample size and $p$ is the dimension of $\mathbf{x}_t$ .
lag.k	Time lag $K$ , a positive integer, used to calculate the test statistic [See (4) in Chang, Yao and Zhou (2017)]. Default is lag.k = 2.
B	Bootstrap times for generating multivariate normal distributed random vectors in calculating the critical value. Default is B = 2000.
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'.

pre	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 <a href="#">PCA_TS</a> .
alpha	The prescribed significance level. Default is 0.05.
k0	A positive integer specified to calculate $\widehat{W}_y$ . See parameter lag.k in <a href="#">PCA_TS</a> for more information.
thresh	Logical. It determines whether to perform the threshold method to estimate $\widehat{W}_y$ or not. See parameter thresh in <a href="#">PCA_TS</a> for more information.
tuning.vec	The value of thresholding tuning parameter $\lambda$ . See parameter tuning.vec in <a href="#">PCA_TS</a> for more information.

### Value

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

statistic	The value of the test statistic.
p.value	Numerical value which represents the p-value of the test based on the observed data $\{\mathbf{x}_t\}_{t=1}^n$ .
lag.k	The time lag used in function.
method	A character string indicating what method was performed.
kernel.type	A character string indicating what kernel method was performed.

### References

- Chang, J., Yao, Q. & Zhou, W. (2017). *Testing for high-dimensional white noise using maximum cross-correlations*, *Biometrika*, Vol. 104, pp. 111–127.
- Chang, J., Guo, B. & Yao, Q. (2018). *Principal component analysis for second-order stationary vector time series*, *The Annals of Statistics*, Vol. 46, pp. 2094–2124.
- Cai, T. and Liu, W. (2011). *Adaptive thresholding for sparse covariance matrix estimation*, *Journal of the American Statistical Association*, Vol. 106, pp. 672–684.

### See Also

[PCA\\_TS](#)

### Examples

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

# Index

Coint, [2](#)  
CP\_MTS, [4](#), [6](#)  
DGP.CP, [5](#)  
Factors, [7](#), [9](#)  
HDSReg, [8](#)  
MartG\_test, [10](#)  
PCA\_TS, [11](#), [20](#)  
SpecMulTest, [15](#)  
SpecTest, [16](#)  
UR\_test, [18](#)  
WN\_test, [19](#)