Package: HDTSA (via r-universe)

October 8, 2026

Type Package

Title High Dimensional Time Series Analysis Tools

Version 1.0.6-2

Date 2026-06-30

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

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

Description An implementation for high-dimensional time series analysis methods, including factor model for vector time series proposed by Lam and Yao (2012) <doi:10.1214/12-AOS970> and Chang, Guo and Yao (2015) <doi:10.1016/j.jeconom.2015.03.024>, martingale difference test proposed by Chang, Jiang and Shao (2023) <doi:10.1016/j.jeconom.2022.09.001>, principal component analysis for vector time series proposed by Chang, Guo and Yao (2018) <doi:10.1214/17-AOS1613>, cointegration analysis proposed by Zhang, Robinson and Yao (2019) <doi:10.1080/01621459.2018.1458620>, unit root test proposed by Chang, Cheng and Yao (2022) <doi:10.1093/biomet/asab034>, white noise tests proposed by Chang, Yao and Zhou (2017) <doi:10.1093/biomet/asw066> and Chang et al. (2026+), CP-decomposition for matrix time series proposed by Chang et al. (2023) <doi:10.1093/jrsssb/qkac011> and Chang et al. (2026+) <doi:10.48550/arXiv.2410.05634>, CP-decomposition for tensor time series proposed by Chang et al. (2026+) <doi:10.48550/arXiv.2606.08560>, and statistical inference for spectral density matrix proposed by Chang et al. (2025) <doi:10.1080/01621459.2025.2468013>.

License GPL-3

Depends R (>= 3.5.0)

Imports stats, Rcpp, clime, sandwich, methods, MASS, QZ, jointDiag, vars, forecast, rTensor

LinkingTo RcppArmadillo, RcppEigen, Rcpp

Suggests knitr

NeedsCompilation yes

RoxygenNote 7.3.2

Encoding UTF-8

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

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

Repository https://cran.r-universe.dev

Date/Publication 2026-07-07 08:46:35 UTC

RemoteUrl https://github.com/cran/HDTSA

RemoteRef HEAD

RemoteSha c2a0acf061a398a94365b93470ca1ce530f27d50

Contents

BeijingAir . . . 3
Coint . . . 4
CP_Inference . . . 6
CP_MTS . . . 8
CP_TTS . . . 12
DGP.CP . . . 15
Factors . . . 16
FamaFrench . . . 18
HDSReg . . . 18
IPindices . . . 20
MartG_test . . . 21
PCA_TS . . . 23
predict.factors . . . 26
predict.mtscp . . . 28
predict.tspca . . . 29
QWIdata . . . 31
SpecMulTest . . . 31
SpecTest . . . 33
UR_test . . . 34
WN_test . . . 36

Index **38**

`BeijingAir` *Beijing multi-site air pollution tensor data*

Description

The dataset is a tensor time series constructed from the Beijing multi-site air-quality data. The raw data contain six hourly air-pollution measurements, namely $PM_{2.5}$, PM_{10}, SO_2, NO_2, CO, and O_3, collected from 12 nationally controlled air-quality monitoring stations in Beijing from March 1, 2013 to February 28, 2017.

This dataset is not the raw data. It is a processed version constructed for studying the spatio-temporal dependence structure of air pollution in Beijing. Specifically, missing values and extreme values in the hourly observations were handled by interpolation. Then, for each monitoring station and each pollutant type, the hourly series were differenced to focus on concentration changes rather than levels. Finally, all series were standardized to remove the effects of different measurement scales across pollutants and stations.

After these preprocessing steps, the data form a tensor time series $\mathcal{Y}_t = (y_{t,\ell_1,\ell_2,\ell_3})_{12\times 6\times 24}$ for $t \in [1461]$, where $y_{t,\ell_1,\ell_2,\ell_3}$ records the standardized concentration change of pollutant ℓ_2 at station ℓ_1 during the ℓ_3-th hour of day t.

Usage

```
data(BeijingAir)
```

Format

A 4-way array with dimension $1461 \times 12 \times 6 \times 24$:

1st dimension The day index, with 1461 daily observations from March 1, 2013 to February 28, 2017.

2nd dimension The 12 air-quality monitoring stations in Beijing.

3rd dimension The 6 pollutants: $PM_{2.5}$, PM_{10}, SO_2, NO_2, CO, and O_3.

4th dimension The 24 hours within each day.

Source

The raw data can be downloaded from https://archive.ics.uci.edu/dataset/501/beijing+multi+site+air+quality+data

Coint *Identifying the cointegration rank of nonstationary vector time series*

Description

`Coint()` deals with cointegration analysis for high-dimensional vector time series proposed in Zhang, Robinson and Yao (2019). Consider the model:

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

where $\mathbf{A}$ is a $p \times p$ unknown and invertible constant matrix, $\mathbf{x}_t = (\mathbf{x}'_{t,1}, \mathbf{x}'_{t,2})'$ is a latent $p \times 1$ process, $\mathbf{x}_{t,2}$ is an $r \times 1$ $I(0)$ process, $\mathbf{x}_{t,1}$ is a process with nonstationary components, and no linear combination of $\mathbf{x}_{t,1}$ is $I(0)$. This function aims to estimate the cointegration rank r and the invertible constant matrix $\mathbf{A}$.

Usage

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

Arguments

`Y`	An $n \times p$ data matrix $\mathbf{Y} = (\mathbf{y}_1, \ldots, \mathbf{y}_n)'$, where n is the number of the observations of the $p \times 1$ time series $\{\mathbf{y}_t\}_{t=1}^n$.
`lag.k`	The time lag K used to calculate the nonnegative definte matrix $\hat{\mathbf{W}}_y$: $$\hat{\mathbf{W}}_y \;=\; \sum_{k=0}^{K} \hat{\mathbf{\Sigma}}_y(k)\hat{\mathbf{\Sigma}}_y(k)' \,,$$ where $\hat{\mathbf{\Sigma}}_y(k)$ is the sample autocovariance of $\mathbf{y}_t$ at lag k. The default is 5.
`type`	The method used to identify the cointegration rank. Available options include: `"acf"` (the default) for the method based on the sample autocorrelations, `"urtest"` for the method based on the unit root tests, and `"both"` to apply these two methods. See Section 2.3 of Zhang, Robinson and Yao (2019) and 'Details' for more information.
`c0`	The prescribed constant c_0 involved in the method based on the sample correlations, which is used when `type = "acf"` or `type = "both"`. See Section 2.3 of Zhang, Robinson and Yao (2019) and 'Details' for more information. The default is 0.3.

`m` The prescribed constant m involved in the method based on the sample correlations, which is used when `type = "acf"` or `type = "both"`. See Section 2.3 of Zhang, Robinson and Yao (2019) and 'Details' for more information. The default is 20.

`alpha` The significance level α of the unit root tests, which is used when `type = "urtest"` or `type = "both"`. See 'Details'. The default is 0.01.

Details

Write $\hat{\mathbf{x}}_t = \hat{\mathbf{A}}'\mathbf{y}_t \equiv (\hat{x}_t^1, \ldots, \hat{x}_t^p)'$. When `type = "acf"`, `Coint()` estimates r by

$$\hat{r} = \sum_{i=1}^{p} 1\left\{ \frac{S_i(m)}{m} < c_0 \right\}$$

for some constant $c_0 \in (0, 1)$ and some large constant m, where $S_i(m)$ is the sum of the sample autocorrelations of $\hat{x}_t^i$ over lags 1 to m, which is specified in Section 2.3 of Zhang, Robinson and Yao (2019).

When `type = "urtest"`, `Coint()` estimates r by unit root tests. For $i = 1, \ldots, p$, consider the null hypothesis

$$H_{0,i} : \hat{x}_t^{p-i+1} \sim I(0)\,.$$

The estimation procedure for r can be implemented as follows:

Step 1. Start with $i = 1$. Perform the unit root test proposed in Chang, Cheng and Yao (2021) for $H_{0,i}$.

Step 2. If the null hypothesis is not rejected at the significance level α, increment i by 1 and repeat Step 1. Otherwise, stop the procedure and denote the value of i at termination as i_0. The cointegration rank is then estimated as $\hat{r} = i_0 - 1$.

Value

An object of class `"coint"`, which contains the following components:

`A` The estimated $\hat{\mathbf{A}}$.

`coint_rank` The estimated cointegration rank $\hat{r}$.

`lag.k` The time lag used in function.

`method` A string indicating which method is used to identify the cointegration rank.

References

Chang, J., Cheng, G., & Yao, Q. (2022). Testing for unit roots based on sample autocovariances. *Biometrika*, **109**, 543–550. doi:10.1093/biomet/asab034.

Zhang, R., Robinson, P., & Yao, Q. (2019). Identifying cointegration by eigenanalysis. *Journal of the American Statistical Association*, **114**, 916–927. doi:10.1080/01621459.2018.1458620.

Examples

```
# Example 1 (Example 1 in Zhang, Robinson and Yao (2019))
## Generate yt
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 = "both")
```

`CP_Inference` *Inference for the Double Projection Iterations (DPI) factor loading estimator in the tensor time series CP-factor model*

Description

This function performs inference for the DPI estimator of a factor loading vector in the tensor CP-factor model (Chang et al., 2026+). Given a direction vector `h`, a factor index `i`, a mode index `j`, the observed tensor time series `Y`, and the output object returned by `CP_TTS`, the function returns the linear transformation of the debiased estimate, its estimated standard error, the original iterative loading estimator, and the estimated bias-correction term.

Usage

```
CP_Inference(
  h,
  i,
  j,
  Y,
  res.CP.DPI,
  var.est = c("plug-in", "long-run"),
  kernel = NULL,
  b_n = NULL
)
```

Arguments

`h`	A numeric vector of length d_j. It specifies the linear transformation $\mathbf{h}^\top(\hat{\mathbf{a}}_{i,j} - \hat{\boldsymbol{\vartheta}}_{i,j})$, where $\hat{\boldsymbol{\vartheta}}_{i,j}$ is the bias-correction term.

`i`	A positive integer. The factor index.
`j`	A positive integer. The tensor mode index.
`Y`	An array containing the observed tensor time series with dimension $n \times d_1 \times \cdots \times d_m$.
`res.CP.DPI`	An output object returned by `CP_TTS`.
`var.est`	A character string specifying the variance estimator. Available options are `"plug-in"` and `"long-run"`. The default is `"plug-in"`.
`kernel`	A kernel function used for the long-run variance estimator. If `NULL`, the quadratic spectral kernel is used. This argument is only used when `var.est = "long-run"`.
`b_n`	A positive number specifying the bandwidth used for the long-run variance estimator. If `NULL`, a data-driven bandwidth rule is used. This argument is only used when `var.est = "long-run"`.

Details

Let $\hat{\mathbf{a}}_{i,j}$ be the DPI estimator of the loading vector for factor i and mode j. The function computes the debiased estimator $\hat{\mathbf{a}}_{i,j} - \hat{\boldsymbol{\vartheta}}_{i,j}$ and returns the linear transformation

$$\mathbf{h}^\top \left(\hat{\mathbf{a}}_{i,j} - \hat{\boldsymbol{\vartheta}}_{i,j} \right).$$

The reported standard error is

$$\sqrt{\widehat{\mathrm{Var}} \left[\mathbf{h}^\top \left(\hat{\mathbf{a}}_{i,j} - \hat{\boldsymbol{\vartheta}}_{i,j} \right) \right] / n},$$

where n is the sample size. The variance can be estimated either by the plug-in estimator derived under the independence condition between the factors and idiosyncratic errors, or by a kernel-type long-run variance estimator. See Chang et al. (2026+) for more details on the construction of $\hat{\boldsymbol{\vartheta}}_{i,j}$ and the estimated variance.

Value

A list with the following components:

`aij.h.de` The debiased linear transformation $\mathbf{h}^\top(\hat{\mathbf{a}}_{i,j} - \hat{\boldsymbol{\vartheta}}_{i,j})$.

`se.h.ij` The estimated standard error of `aij.h.de`.

`aij.iter` The original iterative estimator $\hat{\mathbf{a}}_{i,j}$.

`vartheta.ij` The estimated bias-correction term $\hat{\boldsymbol{\vartheta}}_{i,j}$.

References

Chang, J., Huang, G., Yao, Q., & Yu, L. (2026+). CP-factorization for high dimensional tensor time series and double projection iterations. *Preprint*. doi:10.48550/arXiv.2606.08560.

Examples

```
## Not run:
fit <- CP_TTS(Y)

out <- CP_Inference(
  h = h,
  i = 1,
  j = 2,
  Y = Y,
  res.CP.DPI = fit
)

out$aij.h.de
out$se.h.ij
out$aij.iter
out$vartheta.ij

## Use the long-run variance estimator
out.lr <- CP_Inference(
  h = h,
  i = 1,
  j = 2,
  Y = Y,
  res.CP.DPI = fit,
  var.est = "long-run"
)

## End(Not run)
```

CP_MTS *Estimating the matrix time series CP-factor model*

Description

`CP_MTS()` deals with the estimation of the CP-factor model for matrix time series:

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

where $\mathbf{X}_t = \mathrm{diag}(x_{t,1}, \ldots, x_{t,d})$ is a $d \times d$ unobservable diagonal matrix, $\boldsymbol{\epsilon}_t$ is a $p \times q$ matrix white noise, $\mathbf{A}$ and $\mathbf{B}$ are, respectively, $p \times d$ and $q \times d$ unknown constant matrices with their columns being unit vectors, and $1 \leq d < \min(p, q)$ is an unknown integer. Let $\mathrm{rank}(\mathbf{A}) = d_1$ and $\mathrm{rank}(\mathbf{B}) = d_2$ with some unknown $d_1, d_2 \leq d$. This function aims to estimate d, d_1, d_2 and the loading matrices $\mathbf{A}$ and $\mathbf{B}$ using the methods proposed in Chang et al. (2023) and Chang et al. (2024).

Usage

```
CP_MTS(
  Y,
  xi = NULL,
  Rank = NULL,
  lag.k = 20,
  lag.ktilde = 10,
  method = c("CP.Direct", "CP.Refined", "CP.Unified"),
  thresh1 = FALSE,
  thresh2 = FALSE,
  thresh3 = FALSE,
  delta1 = 2 * sqrt(log(dim(Y)[2] * dim(Y)[3])/dim(Y)[1]),
  delta2 = delta1,
  delta3 = delta1
)
```

Arguments

Y	An $n \times p \times q$ array, where n is the number of observations of the $p \times q$ matrix time series $\{\mathbf{Y}_t\}_{t=1}^n$.
xi	An $n \times 1$ vector $\boldsymbol{\xi} = (\xi_1, \ldots, \xi_n)'$, where ξ_t represents a linear combination of $\mathbf{Y}_t$. If xi = NULL (the default), ξ_t is determined by the PCA method introduced in Section 5.1 of Chang et al. (2023). Otherwise, xi can be given by the users.
Rank	A list containing the following components: d representing the number of columns of $\mathbf{A}$ and $\mathbf{B}$, d1 representing the rank of $\mathbf{A}$, and d2 representing the rank of $\mathbf{B}$. If set to NULL (default), d, d_1, and d_2 will be estimated. Otherwise, they can be given by the users.
lag.k	The time lag K used to calculate the nonnegative definite matrices $\hat{\mathbf{M}}_1$ and $\hat{\mathbf{M}}_2$ when method = "CP.Refined" or method = "CP.Unified": $$\hat{\mathbf{M}}_1 = \sum_{k=1}^{K} \hat{\mathbf{\Sigma}}_k \hat{\mathbf{\Sigma}}_k' \text{ and } \hat{\mathbf{M}}_2 = \sum_{k=1}^{K} \hat{\mathbf{\Sigma}}_k' \hat{\mathbf{\Sigma}}_k \,,$$ where $\hat{\mathbf{\Sigma}}_k$ is an estimate of the cross-covariance between $\mathbf{Y}_t$ and ξ_t at lag k. See 'Details'. The default is 20.
lag.ktilde	The time lag $\tilde{K}$ involved in the unified estimation method [See (16) in Chang et al. (2024)], which is used when method = "CP.Unified". The default is 10.
method	A string indicating which CP-decomposition method is used. Available options include: "CP.Direct" (the default) for the direct estimation method [See Section 3.1 of Chang et al. (2023)], "CP.Refined" for the refined estimation method [See Section 3.2 of Chang et al. (2023)], and "CP.Unified" for the unified estimation method [See Section 4 of Chang et al. (2024)]. The validity of methods "CP.Direct" and "CP.Refined" depends on the assumption $d_1 = d_2 = d$. When $d_1, d_2 \leq d$, the method "CP.Unified" can be applied. See Chang et al. (2024) for details.

`thresh1` Logical. If `FALSE` (the default), no thresholding will be applied in $\hat{\mathbf{\Sigma}}_k$, which indicates that the threshold level $\delta_1 = 0$. If `TRUE`, δ_1 will be set through `delta1`. `thresh1` is used for all three methods. See 'Details'.

`thresh2` Logical. If `FALSE` (the default), no thresholding will be applied in $\check{\mathbf{\Sigma}}_k$, which indicates that the threshold level $\delta_2 = 0$. If `TRUE`, δ_2 will be set through `delta2`. `thresh2` is used only when `method = "CP.Refined"`. See 'Details'.

`thresh3` Logical. If `FALSE` (the default), no thresholding will be applied in $\vec{\mathbf{\Sigma}}_k$, which indicates that the threshold level $\delta_3 = 0$. If `TRUE`, δ_3 will be set through `delta3`. `thresh3` is used only when `method = "CP.Unified"`. See 'Details'.

`delta1` The value of the threshold level δ_1. The default is $\delta_1 = 2\sqrt{n^{-1}\log(pq)}$.

`delta2` The value of the threshold level δ_2. The default is $\delta_2 = 2\sqrt{n^{-1}\log(pq)}$.

`delta3` The value of the threshold level δ_3. The default is $\delta_3 = 2\sqrt{n^{-1}\log(pq)}$.

Details

All three CP-decomposition methods involve the estimation of the autocovariance of $\mathbf{Y}_t$ and ξ_t at lag k, which is defined as follows:

$$\hat{\mathbf{\Sigma}}_k = T_{\delta_1}\{\hat{\mathbf{\Sigma}}_{\mathbf{Y},\xi}(k)\} \text{ with } \hat{\mathbf{\Sigma}}_{\mathbf{Y},\xi}(k) = \frac{1}{n-k}\sum_{t=k+1}^{n}(\mathbf{Y}_t - \bar{\mathbf{Y}})(\xi_{t-k} - \bar{\xi})\,,$$

where $\bar{\mathbf{Y}} = n^{-1}\sum_{t=1}^{n}\mathbf{Y}_t$, $\bar{\xi} = n^{-1}\sum_{t=1}^{n}\xi_t$ and $T_{\delta_1}(\cdot)$ is a threshold operator defined as $T_{\delta_1}(\mathbf{W}) = \{w_{i,j}1(|w_{i,j}| \geq \delta_1)\}$ for any matrix $\mathbf{W} = (w_{i,j})$, with the threshold level $\delta_1 \geq 0$ and $1(\cdot)$ representing the indicator function. Chang et al. (2023) and Chang et al. (2024) suggest to choose $\delta_1 = 0$ when p, q are fixed and $\delta_1 > 0$ when $pq \gg n$.

The refined estimation method involves

$$\check{\mathbf{\Sigma}}_k = T_{\delta_2}\{\hat{\mathbf{\Sigma}}_{\check{\mathbf{Y}}}(k)\} \text{ with } \hat{\mathbf{\Sigma}}_{\check{\mathbf{Y}}}(k) = \frac{1}{n-k}\sum_{t=k+1}^{n}(\mathbf{Y}_t - \bar{\mathbf{Y}}) \otimes \mathrm{vec}(\mathbf{Y}_{t-k} - \bar{\mathbf{Y}})\,,$$

where $T_{\delta_2}(\cdot)$ is a threshold operator with the threshold level $\delta_2 \geq 0$, and $\mathrm{vec}(\cdot)$ is a vecterization operator with $\mathrm{vec}(\mathbf{H})$ being the $(m_1 m_2)\times 1$ vector obtained by stacking the columns of the $m_1 \times m_2$ matrix $\mathbf{H}$. See Section 3.2.2 of Chang et al. (2023) for details.

The unified estimation method involves

$$\vec{\mathbf{\Sigma}}_k = T_{\delta_3}\{\hat{\mathbf{\Sigma}}_{\vec{\mathbf{Y}}}(k)\} \text{ with } \hat{\mathbf{\Sigma}}_{\vec{\mathbf{Y}}}(k) = \frac{1}{n-k}\sum_{t=k+1}^{n}\mathrm{vec}(\mathbf{Y}_t - \bar{\mathbf{Y}})\{\mathrm{vec}(\mathbf{Y}_{t-k} - \bar{\mathbf{Y}})\}'\,,$$

where $T_{\delta_3}(\cdot)$ is a threshold operator with the threshold level $\delta_3 \geq 0$. See Section 4.2 of Chang et al. (2024) for details.

Value

An object of class `"mtscp"`, which contains the following components:

`A` The estimated $p \times \hat{d}$ left loading matrix $\hat{\mathbf{A}}$.

`B` The estimated $q \times \hat{d}$ right loading matrix $\hat{\mathbf{B}}$.

f	The estimated latent process $\hat{x}_{t,1}, \ldots, \hat{x}_{t,\hat{d}}$.
Rank	The estimated $\hat{d}_1, \hat{d}_2$, and $\hat{d}$.
method	A string indicating which CP-decomposition method is used.

References

Chang, J., He, J., Yang, L., & Yao, Q. (2023). Modelling matrix time series via a tensor CP-decomposition. *Journal of the Royal Statistical Society Series B: Statistical Methodology*, **85**, 127–148. doi:10.1093/jrsssb/qkac011.

Chang, J., Du, Y., Huang, G., & Yao, Q. (2026+). Identification and estimation for matrix time series CP-factor models. *The Annals of Statistics*, in press. doi:10.48550/arXiv.2410.05634.

Examples

```
# Example 1.
p <- 10
q <- 10
n <- 400
d = d1 = d2 <- 3
## DGP.CP() generates simulated data for the example in Chang et al. (2024).
data <- DGP.CP(n, p, q, d, d1, d2)
Y <- data$Y

## d is unknown
res1 <- CP_MTS(Y, method = "CP.Direct")
res2 <- CP_MTS(Y, method = "CP.Refined")
res3 <- CP_MTS(Y, method = "CP.Unified")

## d is known
res4 <- CP_MTS(Y, Rank = list(d = 3), method = "CP.Direct")
res5 <- CP_MTS(Y, Rank = list(d = 3), method = "CP.Refined")


# Example 2.
p <- 10
q <- 10
n <- 400
d1 = d2 <- 2
d <-3
data <- DGP.CP(n, p, q, d, d1, d2)
Y1 <- data$Y

## d, d1 and d2 are unknown
res6 <- CP_MTS(Y1, method = "CP.Unified")
## d, d1 and d2 are known
res7 <- CP_MTS(Y1, Rank = list(d = 3, d1 = 2, d2 = 2), method = "CP.Unified")
```

CP_TTS *Estimating the tensor time series CP-factor model*

Description

`CP_TTS()` deals with the estimation of the CP-factor model for tensor time series. Let $\mathcal{Y}_t$ be a tensor in $\mathbb{R}^{d_1 \times \cdots \times d_m}$. The tensor CP-factor model is given by

$$\mathcal{Y}_t = \sum_{i=1}^{r} w_i f_{t,i}\, \mathbf{a}_{i,1} \circ \mathbf{a}_{i,2} \circ \cdots \circ \mathbf{a}_{i,m} + \mathcal{E}_t, \quad t \geq 1,$$

where $1 \leq r \leq \min_{j\in[m]} d_j$ is a fixed but unknown constant, $\mathcal{E}_t \in \mathbb{R}^{d_1 \times \cdots \times d_m}$ is the idiosyncratic error tensor, $\mathbf{f}_t = (f_{t,1}, \ldots, f_{t,r})'$ is the r-dimensional factor vector, and $\mathbf{a}_{i,j}$ is a d_j-dimensional loading vector corresponding to the i-th factor and the j-th mode. Without loss of generality, we assume $|\mathbf{a}_{i,j}|_2 = 1$ for $i \in [r]$ and $j \in [m]$. This function aims to estimate r and the loading vectors $\{\mathbf{a}_{i,j}\}_{i\in[r],j\in[m]}$ using the method proposed in Chang et al. (2026+).

Usage

```
CP_TTS(Y, xi = NULL, r = NULL, A.init = NULL, control.DPI = list())
```

Arguments

`Y`	An array representing a tensor-valued time series with dimension $n \times d_1 \times \cdots \times d_m$, where n is the sample size and $m \geq 2$.
`xi`	An auxiliary scalar series $(\xi_1, \ldots, \xi_n)'$, which is a linear combination of $\mathrm{vec}(\mathcal{Y}_t)$. If `xi = NULL` (the default), ξ_t is estimated by the PCA method described in Chang et al. (2023), or by a randomized projection method by setting `random.projection = TRUE` in `control.DPI`.
`r`	The prescribed number of factors. If set to `NULL` (the default), r is estimated from the data by the ER method or the Log-ER method by setting `ratio.type` in `control.DPI` to `"classical"` or `"log"`, respectively.
`A.init`	Optional initial loading matrices. It should be a list of length m, where the j-th sublist is a $d_j \times r$ matrix. If `NULL`, an initial estimator is obtained from an initialization step.
`control.DPI`	A named list of control parameters used in the double projection iteration (DPI) algorithm. The supported components are:
	`lag.k.dpi` Positive integer. Number of lags K used in $\tilde{\mathbf{M}}_j = \sum_{k=1}^{K} \tilde{\mathbf{\Sigma}}_{k,j} \tilde{\mathbf{\Sigma}}_{k,j}'$, where $\tilde{\mathbf{\Sigma}}_{k,j}$ is an estimate of the cross-covariance between $\mathbf{Y}_{t,j}$, the mode-j matricization of $\mathcal{Y}_t$ with dimension $d_j \times \prod_{j'\neq j} d_{j'}$, and ξ_t at lag k. Default is 10.
	`threshold` Logical. Whether thresholding is applied in the initialization and iteration steps. Default is `TRUE`.
	`delta` Optional thresholding level used in the initialization step. Default is `NULL`. If `NULL`, it is selected via a grid search method.

`delta2` Numeric vector of length `m`, controlling the thresholding level in each tensor mode during the iterative update. The default `j`-th element is $\hat{\sigma}_0(n^{-1}\log d_j)^{1/2}$, where $\hat{\sigma}_0^2 = (n\prod_{j=1}^m d_j)^{-1}\sum_{t=1}^n \|\mathcal{Y}_t\|_{\mathrm{F}}^2$.

`ratio.type` Character string specifying the ratio criterion used in estimating r. Typical choices are `"log"` for the Log-ER method and `"classical"` for the ER method. Default is `"log"`.

`random.projection` Logical. If `TRUE`, a randomized projection step is used to select `xi`. See Section 3.4 of Chang et al. (2026+) for details. Default is `FALSE`.

`iter.max` Maximum number of iterative updates. Default is 20.

`eps` Stopping tolerance for the iterative algorithm. Default is 10^{-4}.

`grid.num` Integer. Number of grid points used when selecting the thresholding level in the initialization step. Default is 50.

`delta.max` Maximum value of the thresholding grid in the initialization step. Default is $0.1\hat{\sigma}_0(n^{-1}\sum_{j=1}^m \log d_j)^{1/2}$.

`print.eps` Logical. Whether to print the iterative convergence measure. Default is `FALSE`.

`iter.lag` Positive integer. Number of candidate lags used in each iterative update. Default is 1.

`all.put` Logical. If `TRUE`, the iterative routine returns full intermediate outputs; otherwise, only a compact result is returned. Default is `FALSE`.

`A` Optional true loading matrices, used only for diagnostic purposes in simulations. Default is `NULL`.

`component` Optional true common component tensor, used only for diagnostic purposes in simulations. Default is `NULL`.

Details

The initial method involves the estimation of the autocovariance between $\mathbf{Y}_{t,j}$ and ξ_t at lag k, which is defined as follows:

$$\hat{\mathbf{\Sigma}}_{k,j} = T_{\delta_1}\{\hat{\mathbf{\Sigma}}_{\mathbf{Y}_j,\xi}(k)\} \quad \text{with} \quad \hat{\mathbf{\Sigma}}_{\mathbf{Y}_j,\xi}(k) = \frac{1}{n-k}\sum_{t=k+1}^{n}(\mathbf{Y}_{t,j} - \bar{\mathbf{Y}}_j)(\xi_{t-k} - \bar{\xi}),$$

where $\bar{\mathbf{Y}}_j = n^{-1}\sum_{t=1}^n \mathbf{Y}_{t,j}$, $\bar{\xi} = n^{-1}\sum_{t=1}^n \xi_t$, and $T_{\delta_1}(\cdot)$ is a threshold operator defined as $T_{\delta_1}(\mathbf{W}) = \{w_{i,j}1(|w_{i,j}| \geq \delta_1)\}$ for any matrix $\mathbf{W} = (w_{i,j})$, with threshold level $\delta_1 \geq 0$ and $1(\cdot)$ denoting the indicator function. Chang et al. (2026+) suggest choosing δ_1 by a grid search method. See Section 3.4 of Chang et al. (2026+) for details.

Value

The function returns a list containing the following components:

`A.hat` The final iterative loading matrices.

`A.init` The initial loading matrices used to start the iteration.

`Sigma.yij.xii.1` The thresholded moment vectors used in the iterative updates and inference.

`r` The number of factors used in the iterative procedure.

`iter.step` The number of iterations performed.

`fnorm.resid` The relative Frobenius norm of the residuals recorded during the iterations.

`f.hat` The estimated factor series based on the final iterative loading matrices.

`f.hat.inl` The estimated factor series based on the initial loading matrices.

`delta.sel` The selected threshold level from the initial one-pass estimation. If `A.init` is supplied by the user, this value is `NULL`.

`control.DPI` The control list actually used in the function after merging user-supplied values with the defaults.

References

Chang, J., He, J., Yang, L., & Yao, Q. (2023). Modelling matrix time series via a tensor CP-decomposition. *Journal of the Royal Statistical Society Series B: Statistical Methodology*, **85**, 127–148. doi:10.1093/jrsssb/qkac011.

Chang, J., Huang, G., Yao, Q., & Yu, L. (2026+). CP-factorization for high dimensional tensor time series and double projection iterations. *Preprint*. doi:10.48550/arXiv.2606.08560.

Examples

```
n <- 200
D <- c(10, 10)
r <- 2

data <- HDTSA:::DGP.TCP(
  n = n,
  m = 2,
  D = D,
  r = r,
  w = c(10, 10),
  ar.coef = list(0.5, 0.3),
  factor.loading = "sparse-random",
  alpha = 0.3
)

Y <- data$Y

fit <- CP_TTS(Y)

fit$r
fit$A.hat
fit$f.hat

fit.known <- CP_TTS(Y, r = 2)
```

DGP.CP *Generating simulated data for the example in Chang et al. (2024)*

Description

DGP.CP() function generates simulated data following the data generating process described in Section 7.1 of Chang et al. (2024).

Usage

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

Arguments

n	Integer. The number of observations of the $p \times q$ matrix time series $\mathbf{Y}_t$.
p	Integer. The number of rows of $\mathbf{Y}_t$.
q	Integer. The number of columns of $\mathbf{Y}_t$.
d	Integer. The number of columns of the factor loading matrices $\mathbf{A}$ and $\mathbf{B}$.
d1	Integer. The rank of the $p \times d$ matrix $\mathbf{A}$.
d2	Integer. The rank of the $q \times d$ matrix $\mathbf{B}$.

Details

We generate

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

for any $t = 1, \ldots, n$, where $\mathbf{X}_t = \mathrm{diag}(\mathbf{x}_t)$ with $\mathbf{x}_t = (x_{t,1}, \ldots, x_{t,d})'$ being a $d \times 1$ time series, $\boldsymbol{\epsilon}_t$ is a $p \times q$ matrix white noise, and $\mathbf{A}$ and $\mathbf{B}$ are, respectively, $p \times d$ and $q \times d$ factor loading matrices. $\mathbf{A}$, $\mathbf{X}_t$, and $\mathbf{B}$ are generated based on the data generating process described in Section 7.1 of Chang et al. (2024) and satisfy $\mathrm{rank}(\mathbf{A}) = d_1$ and $\mathrm{rank}(\mathbf{B}) = d_2$, $1 \le d_1, d_2 \le d$.

Value

A list containing the following components:

Y	An $n \times p \times q$ array.
A	The $p \times d$ left loading matrix $\mathbf{A}$.
B	The $q \times d$ right loading matrix $\mathbf{B}$.
X	An $n \times d \times d$ array.

References

Chang, J., Du, Y., Huang, G., & Yao, Q. (2026+). Identification and estimation for matrix time series CP-factor models. *The Annals of Statistics*, in press. doi:10.48550/arXiv.2410.05634.

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

## The first observation: Y_1
Y[1, , ]
```

Factors *Factor analysis for vector time series*

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 $\boldsymbol{\epsilon}_t$ 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,
  thresh = FALSE,
  delta = 2 * sqrt(log(ncol(Y))/nrow(Y)),
  twostep = FALSE
)
```

Arguments

`Y`	An $n \times p$ data matrix $\mathbf{Y} = (\mathbf{y}_1, \ldots, \mathbf{y}_n)'$, where n is the number of the observations of the $p \times 1$ time series $\{\mathbf{y}_t\}_{t=1}^n$.
`lag.k`	The time lag K used to calculate the nonnegative definite matrix $\hat{\mathbf{M}}$:

$$\hat{\mathbf{M}} = \sum_{k=1}^{K} T_\delta\{\hat{\boldsymbol{\Sigma}}_y(k)\} T_\delta\{\hat{\boldsymbol{\Sigma}}_y(k)\}',$$

where $\hat{\mathbf{\Sigma}}_y(k)$ is the sample autocovariance of $\mathbf{y}_t$ at lag k and $T_\delta(\cdot)$ is a threshold operator with the threshold level $\delta \geq 0$. See 'Details'. The default is 5.

`thresh` Logical. If `thresh = FALSE` (the default), no thresholding will be applied to estimate $\hat{\mathbf{M}}$. If `thresh = TRUE`, δ will be set through `delta`.

`delta` The value of the threshold level δ. The default is $\delta = 2\sqrt{n^{-1}\log p}$.

`twostep` Logical. If `twostep = FALSE` (the default), the standard procedure [See Section 2.2 in Lam and Yao (2012)] for estimating r and $\mathbf{A}$ will be implemented. If `twostep = TRUE`, the two-step estimation procedure [See Section 4 in Lam and Yao (2012)] for estimating r and $\mathbf{A}$ will be implemented.

Details

The threshold operator $T_\delta(\cdot)$ is defined as $T_\delta(\mathbf{W}) = \{w_{i,j}1(|w_{i,j}| \geq \delta)\}$ for any matrix $\mathbf{W} = (w_{i,j})$, with the threshold level $\delta \geq 0$ and $1(\cdot)$ representing the indicator function. We recommend to choose $\delta = 0$ when p is fixed and $\delta > 0$ when $p \gg n$.

Value

An object of class `"factors"`, which contains the following components:

`factor_num` The estimated number of factors $\hat{r}$.

`loading.mat` The estimated $p \times \hat{r}$ factor loading matrix $\hat{\mathbf{A}}$.

`X` The $n \times \hat{r}$ matrix $\hat{\mathbf{X}} = (\hat{\mathbf{x}}_1, \ldots, \hat{\mathbf{x}}_n)'$ with $\hat{\mathbf{x}}_t = \hat{\mathbf{A}}'\hat{\mathbf{y}}_t$.

`lag.k` The time lag used in function.

References

Lam, C., & Yao, Q. (2012). Factor modelling for high-dimensional time series: Inference for the number of factors. *The Annals of Statistics*, **40**, 694–726. doi:10.1214/12AOS970.

Examples

```
# Example 1 (Example in Section 3.3 of lam and Yao 2012)
## Generate y_t
p <- 200
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
```

FamaFrench *Fama-French 10*10 return series*

Description

The portfolios are constructed by the intersections of 10 levels of size, denoted by $\mathrm{S}_1, \ldots, \mathrm{S}_{10}$, and 10 levels of the book equity to market equity ratio (BE), denoted by $\mathrm{BE}_1, \ldots, \mathrm{BE}_{10}$. The dataset consists of monthly returns from January 1964 to December 2021, which contains 69600 observations for 696 total months.

Usage

```
data(FamaFrench)
```

Format

A data frame with 696 rows and 102 columns. The first column represents the month, and the second column named `MKT.RF` represents the monthly market returns. The rest of the columns represent the return series for different sizes and BE-ratios.

Source

http://mba.tuck.dartmouth.edu/pages/faculty/ken.french/data_library.html

HDSReg *Factor analysis with observed regressors for vector time series*

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, $\boldsymbol{\epsilon}_t$ is a vector white noise process, $\mathbf{D}$ 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 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 = 5,
  thresh = FALSE,
  delta = 2 * sqrt(log(ncol(Y))/nrow(Y)),
  twostep = FALSE
)
```

Arguments

Y	An $n \times p$ data matrix $\mathbf{Y} = (\mathbf{y}_1, \ldots, \mathbf{y}_n)'$, where n is the number of the observations of the $p \times 1$ time series $\{\mathbf{y}_t\}_{t=1}^n$.
Z	An $n \times m$ data matrix $\mathbf{Z} = (\mathbf{z}_1, \ldots, \mathbf{z}_n)'$ consisting of the observed regressors.
D	A $p \times m$ regression coefficient matrix $\tilde{\mathbf{D}}$. If D = NULL (the default), our procedure will estimate $\mathbf{D}$ first and let $\tilde{\mathbf{D}}$ be the estimate of $\mathbf{D}$. If D is given by the users, then $\tilde{\mathbf{D}} = \mathbf{D}$.
lag.k	The time lag K used to calculate the nonnegative definte matrix $\hat{\mathbf{M}}_\eta$: $$\hat{\mathbf{M}}_\eta = \sum_{k=1}^{K} T_\delta\{\hat{\mathbf{\Sigma}}_\eta(k)\} T_\delta\{\hat{\mathbf{\Sigma}}_\eta(k)\}',$$ where $\hat{\mathbf{\Sigma}}_\eta(k)$ is the sample autocovariance of $\boldsymbol{\eta}_t = \mathbf{y}_t - \tilde{\mathbf{D}}\mathbf{z}_t$ at lag k and $T_\delta(\cdot)$ is a threshold operator with the threshold level $\delta \geq 0$. See 'Details'. The default is 5.
thresh	Logical. If thresh = FALSE (the default), no thresholding will be applied to estimate $\hat{\mathbf{M}}_\eta$. If thresh = TRUE, δ will be set through delta. See 'Details'.
delta	The value of the threshold level δ. The default is $\delta = 2\sqrt{n^{-1}\log p}$.
twostep	Logical. The same as the argument twostep in Factors.

Details

The threshold operator $T_\delta(\cdot)$ is defined as $T_\delta(\mathbf{W}) = \{w_{i,j} 1(|w_{i,j}| \geq \delta)\}$ for any matrix $\mathbf{W} = (w_{i,j})$, with the threshold level $\delta \geq 0$ and $1(\cdot)$ representing the indicator function. We recommend to choose $\delta = 0$ when p is fixed and $\delta > 0$ when $p \gg n$.

Value

An object of class "factors", which contains the following components:

factor_num	The estimated number of factors $\hat{r}$.
reg.coff.mat	The estimated $p \times m$ regression coefficient matrix $\tilde{\mathbf{D}}$.
loading.mat	The estimated $p \times \hat{r}$ factor loading matrix $\hat{\mathbf{A}}$.
X	The $n \times \hat{r}$ matrix $\hat{\mathbf{X}} = (\hat{\mathbf{x}}_1, \ldots, \hat{\mathbf{x}}_n)'$ with $\hat{\mathbf{x}}_t = \hat{\mathbf{A}}'(\mathbf{y}_t - \tilde{\mathbf{D}}\mathbf{z}_t)$.
lag.k	The time lag used in function.

References

Chang, J., Guo, B., & Yao, Q. (2015). High dimensional stochastic regression with latent factors, endogeneity and nonlinearity. *Journal of Econometrics*, **189**, 297–312. doi:10.1016/j.jeconom.2015.03.024.

See Also

Factors.

Examples

```
# Example 1 (Example 1 in Chang, Guo and Yao (2015)).
## Generate xt
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)

## Generate yt
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)

## D is known
res1 <- HDSReg(Y, Z, D, lag.k = 2)
## D is unknown
res2 <- HDSReg(Y, Z, lag.k = 2)
```

IPindices *U.S. Industrial Production indices*

Description

The dataset consists of 7 monthly U.S. Industrial Production indices, namely *the total index*, *nonindustrial supplies*, *final products*, *manufacturing*, *materials*, *mining*, and *utilities*, from January 1947 to December 2023 published by the U.S. Federal Reserve.

Usage

```
data(IPindices)
```

Format

A data frame with 924 rows and 8 variables:

DATE The observation date

INDPRO The total index

IPB54000S Nonindustrial supplies

IPFINAL Final products

IPMANSICS Manufacturing

IPMAT Materials

IPMINE Mining

IPUTIL Utilities

Source

https://fred.stlouisfed.org/release/tables?rid=13&eid=49670

MartG_test *Testing for martingale difference hypothesis in high dimension*

Description

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

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

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

Usage

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

Arguments

`Y`	An $n \times p$ data matrix $\mathbf{Y} = (\mathbf{y}_1, \ldots, \mathbf{y}_n)'$, where n is the number of the observations of the $p \times 1$ time series $\{\mathbf{y}_t\}_{t=1}^n$.
`lag.k`	The time lag K used to calculate the test statistic [See (3) in Chang, Jiang and Shao (2023)]. The default is 2.
`B`	The number of bootstrap replications for generating multivariate normally distributed random vectors when calculating the critical value. The default is 1000.
`type`	The map used for constructing the test statistic. Available options include: `"Linear"` (the default) for the linear identity map and `"Quad"` for the map including both linear and quadratic terms. `type` can also be set by the users. See 'Details' and Section 2.1 of Chang, Jiang and Shao (2023) for more information.
`alpha`	The significance level of the test. The default is 0.05.
`kernel.type`	The option for choosing the symmetric kernel used in the estimation of long-run covariance matrix. Available options include: `"QS"` (the default) for the Quadratic spectral kernel, `"Par"` for the Parzen kernel, and `"Bart"` for the Bartlett kernel. See Chang, Jiang and Shao (2023) for more information.

Details

Write $\mathbf{x} = (x_1, \ldots, x_p)'$. When `type = "Linear"`, the linear identity map is defined as $\phi(\mathbf{x}) = \mathbf{x}$.

When `type = "Quad"`, $\phi(\mathbf{x}) = \{\mathbf{x}', (\mathbf{x}^2)'\}'$ includes both linear and quadratic terms, where $\mathbf{x}^2 = (x_1^2, \ldots, x_p^2)'$.

We can also choose $\phi(\mathbf{x}) = \cos(\mathbf{x})$ to capture certain type of nonlinear dependence, where $\cos(\mathbf{x}) = (\cos x_1, \ldots, \cos x_p)'$.

See 'Examples'.

Value

An object of class `"hdtstest"`, which contains the following components:

`statistic`	The test statistic of the test.
`p.value`	The p-value of the test.
`lag.k`	The time lag used in function.
`type`	The map used in function.
`kernel.type`	The kernel used in function.

References

Chang, J., Jiang, Q., & Shao, X. (2023). Testing the martingale difference hypothesis in high dimension. *Journal of Econometrics*, **235**, 972–1000. doi:10.1016/j.jeconom.2022.09.001.

Examples

```
# Example 1
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 type = "Quad"

## map can also be defined as an expression in R.
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 can also be defined as a function in R.
map_fun <- function(X) {X <- cbind(X, X^2); X}

res <- MartG_test(X, type=map_fun)
Pvalue <- res$p.value
rej <- res$reject
```

PCA_TS *Principal component analysis for vector time series*

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 (≥ 1) both contemporaneously and serially uncorrelated subseries. See Chang, Guo and Yao (2018).

Usage

```
PCA_TS(
  Y,
  lag.k = 5,
  opt = 1,
  permutation = c("max", "fdr"),
  thresh = FALSE,
  delta = 2 * sqrt(log(ncol(Y))/nrow(Y)),
  prewhiten = TRUE,
  m = NULL,
  beta,
  control = list()
)
```

Arguments

Y	An $n \times p$ data matrix $\mathbf{Y} = (\mathbf{y}_1, \ldots, \mathbf{y}_n)'$, where n is the number of the observations of the $p \times 1$ time series $\{\mathbf{y}_t\}_{t=1}^n$. 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	The time lag K used to calculate the nonnegative definte matrix $\hat{\mathbf{W}}_y$: $$\hat{\mathbf{W}}_y \;=\; \mathbf{I}_p + \sum_{k=1}^{K} T_\delta\{\hat{\mathbf{\Sigma}}_y(k)\} T_\delta\{\hat{\mathbf{\Sigma}}_y(k)\}',$$ where $\hat{\mathbf{\Sigma}}_y(k)$ is the sample autocovariance of $\hat{\mathbf{V}}^{-1/2}\mathbf{y}_t$ at lag k and $T_\delta(\cdot)$ is a threshold operator with the threshold level $\delta \geq 0$. See 'Details'. The default is 5.
opt	An option used to choose which method will be implemented to get a consistent estimate $\hat{\mathbf{V}}$ (or $\hat{\mathbf{V}}^{-1}$) for the covariance (precision) matrix of $\mathbf{y}_t$. If opt = 1, $\hat{\mathbf{V}}$ will be defined as the sample covariance matrix. If opt = 2, the precision matrix $\hat{\mathbf{V}}^{-1}$ will be calculated by using the function clime() of **clime** (Cai, Liu and Luo, 2011) with the arguments passed by control.
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)]. Available options include: "max" (the default) for the maximum cross correlation method and "fdr" for the false discovery rate procedure based on multiple tests. See Sections 2.2.2 and 2.2.3 in Chang, Guo and Yao (2018) for more information.
thresh	Logical. If thresh = FALSE (the default), no thresholding will be applied to estimate $\hat{\mathbf{W}}_y$. If thresh = TRUE, the argument delta is used to specify the threshold level δ.
delta	The value of the threshold level δ. The default is $\delta = 2\sqrt{n^{-1}\log p}$.
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 the prewhiten procedure will not be performed.
m	A positive integer used in the permutation procedure [See (2.10) in Chang, Guo and Yao (2018)]. The default is 10.
beta	The error rate used in the permutation procedure[See (2.16) in Chang, Guo and Yao (2018)] when permutation = "fdr".
control	A list of control arguments. See 'Details'.

Details

The threshold operator $T_\delta(\cdot)$ is defined as $T_\delta(\mathbf{W}) = \{w_{i,j}1(|w_{i,j}| \geq \delta)\}$ for any matrix $\mathbf{W} = (w_{i,j})$, with the threshold level $\delta \geq 0$ and $1(\cdot)$ representing the indicator function. We recommend to choose $\delta = 0$ when p is fixed and $\delta > 0$ when $p \gg n$.

For large p, since the sample covariance matrix may not be consistent, we recommend to use the method proposed in Cai, Liu and Luo (2011) to estimate the precision matrix $\hat{\mathbf{V}}^{-1}$ (opt = 2).

control is a list of arguments passed to the function clime(), which contains the following components:

- `nlambda`: Number of values for program generated lambda. The default is 100.
- `lambda.max`: Maximum value of program generated lambda. The default is 0.8.
- `lambda.min`: Minimum value of program generated lambda. The default is 10^{-4} ($n > p$) or 10^{-2} ($n < p$).
- `standardize`: Logical. If `standardize = TRUE`, the variables will be standardized to have mean zero and unit standard deviation. The default is `FALSE`.
- `linsolver`: An option used to choose which method should be employed. Available options include `"primaldual"` (the default) and `"simplex"`. Rule of thumb: `"primaldual"` for large p, `"simplex"` for small p.

Value

An object of class `"tspca"`, which contains the following components:

`B`	The $p \times p$ transformation matrix $\hat{\mathbf{B}} = \hat{\mathbf{\Gamma}}_y' \hat{\mathbf{V}}^{-1/2}$, where $\hat{\mathbf{\Gamma}}_y$ is a $p \times p$ orthogonal matrix with the columns being the eigenvectors of $\hat{\mathbf{W}}_y$.
`X`	The $n \times p$ matrix $\hat{\mathbf{X}} = (\hat{\mathbf{x}}_1, \ldots, \hat{\mathbf{x}}_n)'$ with $\hat{\mathbf{x}}_t = \hat{\mathbf{B}}\mathbf{y}_t$.
`NoGroups`	The number of groups.
`No_of_Members`	The number of members in each group.
`Groups`	The indices of the components of $\hat{\mathbf{x}}_t$ that belong to each group.
`method`	A string indicating which permutation procedure is performed.

References

Cai, T., Liu, W., & Luo, X. (2011). A constrained L1 minimization approach for sparse precision matrix estimation. *Journal of the American Statistical Association*, **106**, 594–607. doi:10.1198/jasa.2011.tm10155.

Chang, J., Guo, B., & Yao, Q. (2018). Principal component analysis for second-order stationary vector time series. *The Annals of Statistics*, **46**, 2094–2124. doi:10.1214/17AOS1613.

Examples

```
# Example 1 (Example 1 in the supplementary material of Chang, Guo and Yao (2018)).
# p=6, x_t consists of 3 independent subseries with 3, 2 and 1 components.

## Generate x_t
p <- 6;n <- 1500
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)

## permutation = "max" or permutation = "fdr"
res <- PCA_TS(Y, lag.k = 5,permutation = "max")
res1 <- PCA_TS(Y, lag.k = 5,permutation = "fdr", beta = 10^(-10))
Z <- res$X


# Example 2 (Example 2 in the supplementary material of Chang, Guo and Yao (2018)).
# p=20, x_t consists of 5 independent subseries with 6, 5, 4, 3 and 2 components.

## Generate x_t
p <- 20;n <- 3000
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)

## permutation = "max" or permutation = "fdr"
res <- PCA_TS(Y, lag.k = 5,permutation = "max")
res1 <- PCA_TS(Y, lag.k = 5,permutation = "fdr",beta = 10^(-200))
Z <- res$X
```

`predict.factors` *Make predictions from a* `"factors"` *object*

Description

This function makes predictions from a `"factors"` object.

Usage

```
## S3 method for class 'factors'
predict(
  object,
  newdata = NULL,
  n.ahead = 10,
  control_ARIMA = list(),
  control_VAR = list(),
  ...
)
```

Arguments

object	An object of class "factors" constructed by Factors.
newdata	Optional. A new data matrix to predict from.
n.ahead	An integer specifying the number of steps ahead for prediction.
control_ARIMA	A list of arguments passed to the function auto.arima() of **forecast**. See 'Details' and the manual of auto.arima(). The default is list(ic = "aic").
control_VAR	A list of arguments passed to the function VAR() of **vars**. See 'Details' and the manual of VAR(). The default is list(type = "const", lag.max = 6, ic = "AIC").
...	Currently not used.

Details

Forecasting for $\mathbf{y}_t$ can be implemented in two steps:

Step 1. Get the h-step ahead forecast of the $\hat{r} \times 1$ time series $\hat{\mathbf{x}}_t$ [See Factors], denoted by $\hat{\mathbf{x}}_{n+h}$, using a VAR model (if $\hat{r} > 1$) or an ARIMA model (if $\hat{r} = 1$). The orders of VAR and ARIMA models are determined by AIC by default. Otherwise, they can also be specified by users through the arguments control_VAR and control_ARIMA, respectively.

Step 2. The forecasted value for $\mathbf{y}_t$ is obtained by $\hat{\mathbf{y}}_{n+h} = \hat{\mathbf{A}}\hat{\mathbf{x}}_{n+h}$.

Value

ts_pred	A matrix of predicted values.

See Also

Factors

Examples

```
library(HDTSA)
data(FamaFrench, package = "HDTSA")

## Remove the market effects
reg <- lm(as.matrix(FamaFrench[, -c(1:2)]) ~ as.matrix(FamaFrench$MKT.RF))
```

```
Y_2d = reg$residuals

res_factors <- Factors(Y_2d, lag.k = 5)
pred_fac_Y <- predict(res_factors, n.ahead = 1)
```

`predict.mtscp` *Make predictions from a* `"mtscp"` *object*

Description

This function makes predictions from a `"mtscp"` object.

Usage

```
## S3 method for class 'mtscp'
predict(
  object,
  newdata = NULL,
  n.ahead = 10,
  control_ARIMA = list(),
  control_VAR = list(),
  ...
)
```

Arguments

`object`	An object of class `"mtscp"` constructed by `CP_MTS`.
`newdata`	Optional. A new data matrix to predict from.
`n.ahead`	An integer specifying the number of steps ahead for prediction.
`control_ARIMA`	A list of arguments passed to the function `auto.arima()` of **forecast**. See 'Details' and the manual of `auto.arima()`. The default is `list(ic = "aic")`.
`control_VAR`	A list of arguments passed to the function `VAR()` of **vars**. See 'Details' and the manual of `VAR()`. The default is `list(type = "const", lag.max = 6, ic = "AIC")`.
`...`	Currently not used.

Details

Forecasting for $\mathbf{y}_t$ can be implemented in two steps:

Step 1. Get the h-step ahead forecast of the $\hat{d} \times 1$ time series $\hat{\mathbf{x}}_t = (\hat{x}_{t,1}, \ldots, \hat{x}_{t,\hat{d}})'$ [See `CP_MTS`], denoted by $\hat{\mathbf{x}}_{n+h}$, using a VAR model (if $\hat{d} > 1$) or an ARIMA model (if $\hat{d} = 1$). The orders of VAR and ARIMA models are determined by AIC by default. Otherwise, they can also be specified by users through the arguments `control_VAR` and `control_ARIMA`, respectively.

Step 2. The forecasted value for $\mathbf{Y}_t$ is obtained by $\hat{\mathbf{Y}}_{n+h} = \hat{\mathbf{A}}\hat{\mathbf{X}}_{n+h}\hat{\mathbf{B}}'$ with $\hat{\mathbf{X}}_{n+h} = \mathrm{diag}(\hat{\mathbf{x}}_{n+h})$.

Value

Y_pred — A list of length n.ahead, where each element is a $p \times q$ matrix representing the predicted values at each time step.

See Also

CP_MTS

Examples

```
library(HDTSA)
data(FamaFrench, package = "HDTSA")

## Remove the market effects
reg <- lm(as.matrix(FamaFrench[, -c(1:2)]) ~ as.matrix(FamaFrench$MKT.RF))
Y_2d = reg$residuals

## Rearrange Y_2d into a 3-dimensional array Y
Y = array(NA, dim = c(NROW(Y_2d), 10, 10))
for (tt in 1:NROW(Y_2d)) {
  for (ii in 1:10) {
    Y[tt, ii, ] <- Y_2d[tt, (1 + 10*(ii - 1)):(10 * ii)]
  }
}

res_cp <- CP_MTS(Y, lag.k = 20, method = "CP.Refined")
pred_cp_Y <- predict(res_cp, n.ahead = 1)[[1]]
```

---

predict.tspca *Make predictions from a "tspca" object*

Description

This function makes predictions from a "tspca" object.

Usage

```
## S3 method for class 'tspca'
predict(
  object,
  newdata = NULL,
  n.ahead = 10,
  control_ARIMA = list(),
  control_VAR = list(),
  ...
)
```

Arguments

`object`	An object of class `"tspca"` constructed by `PCA_TS`.
`newdata`	Optional. A new data matrix to predict from.
`n.ahead`	An integer specifying the number of steps ahead for prediction.
`control_ARIMA`	A list of arguments passed to the function `auto.arima()` of **forecast**. See 'Details' and the manual of `auto.arima()`. The default is `list(max.d = 0, max.q = 0, ic = "aic")`.
`control_VAR`	A list of arguments passed to the function `VAR()` of **vars**. See 'Details' and the manual of `VAR()`. The default is `list(type = "const", lag.max = 6, ic = "AIC")`.
`...`	Currently not used.

Details

Having obtained $\hat{\mathbf{x}}_t$ using the `PCA_TS` function, which is segmented into q uncorrelated subseries $\hat{\mathbf{x}}_t^{(1)}, \ldots, \hat{\mathbf{x}}_t^{(q)}$, the forecasting for $\mathbf{y}_t$ can be performed in two steps:

Step 1. Get the h-step ahead forecast $\hat{\mathbf{x}}_{n+h}^{(j)}$ $(j = 1, \ldots, q)$ by using a VAR model (if the dimension of $\hat{\mathbf{x}}_t^{(j)}$ is larger than 1) or an ARIMA model (if the dimension of $\hat{\mathbf{x}}_t^{(j)}$ is 1). The orders of VAR and ARIMA models are determined by AIC by default. Otherwise, they can also be specified by users through the arguments `control_VAR` and `control_ARIMA`, respectively.

Step 2. Let $\hat{\mathbf{x}}_{n+h} = (\{\hat{\mathbf{x}}_{n+h}^{(1)}\}', \ldots, \{\hat{\mathbf{x}}_{n+h}^{(q)}\}')'$. The forecasted value for $\mathbf{y}_t$ is obtained by $\hat{\mathbf{y}}_{n+h} = \hat{\mathbf{B}}^{-1}\hat{\mathbf{x}}_{n+h}$.

Value

`Y_pred`	A matrix of predicted values.

See Also

`PCA_TS`

Examples

```
library(HDTSA)
data(FamaFrench, package = "HDTSA")

## Remove the market effects
reg <- lm(as.matrix(FamaFrench[, -c(1:2)]) ~ as.matrix(FamaFrench$MKT.RF))
Y_2d = reg$residuals

res_pca <- PCA_TS(Y_2d, lag.k = 5, thresh = TRUE)
pred_pca_Y <- predict(res_pca, n.ahead = 1)
```

QWIdata — *The national QWI hires data*

Description

The data on new hires at a national level are obtained from the Quarterly Workforce Indicators (QWI) of the Longitudinal Employer-Household Dynamics program at the U.S. Census Bureau (Abowd et al., 2009). The national QWI hires data covers a variable number of years, with some states providing time series going back to 1990 (e.g., Washington), and others (e.g., Massachusetts) only commencing at 2010. For each of 51 states (excluding D.C. but including Puerto Rico) there is a new hires time series for each county. Additional description of the data, along with its relevancy to labor economics, can be found in Hyatt and McElroy (2019).

Usage

```
data(QWIdata)
```

Format

A list with 51 elements. Every element contains a multivariate time series.

Source

https://qwiexplorer.ces.census.gov/

https://ledextract.ces.census.gov/qwi/all

References

Abowd, J. M., Stephens, B. E., Vilhuber, L., Andersson, F., McKinney, K. L., Roemer, M., and Woodcock, S. (2009). The LEHD infrastructure files and the creation of the quarterly workforce indicators. In *Producer dynamics: New evidence from micro data*, pages 149–230. University of Chicago Press. doi:10.7208/chicago/9780226172576.003.0006.

Hyatt, H. R. and McElroy, T. S. (2019). Labor reallocation, employment, and earnings: Vector autoregression evidence. *Labour*, **33**, 463–487. doi:10.1111/labr.12153

SpecMulTest — *Multiple testing with FDR control for spectral density matrix*

Description

`SpecMulTest()` implements a new multiple testing procedure proposed in Chang et al. (2022) for the following 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 $q = 1, \ldots, Q$. Here, $f_{i,j}(\omega)$ represents the cross-spectral density between $x_{t,i}$ and $x_{t,j}$ at frequency ω with $x_{t,i}$ being the i-th component of the $p \times 1$ times series $\mathbf{x}_t$, and $\mathcal{I}^{(q)}$ and $\mathcal{J}^{(q)}$ refer to the set of index pairs and the set of frequencies associated with the q-th test, respectively.

Usage

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

Arguments

`Q`	The number of the hypothesis tests.
`PVal`	A vector of length Q representing p-values of the Q hypothesis tests.
`alpha`	The prescribed level for the FDR control. The default is 0.05.
`seq_len`	The step size for generating a sequence from 0 to $\sqrt{2 \times \log Q - 2 \times \log(\log Q)}$. The default is 0.01.

Value

An object of class `"hdtstest"`, which contains the following component:

`MultiTest`	A logical vector of length Q. If its q-th element is `TRUE`, it indicates that $H_{0,q}$ should be rejected. Otherwise, $H_{0,q}$ should not be rejected.

References

Chang, J., Jiang, Q., McElroy, T. S., & Shao, X. (2025). Statistical inference for high-dimensional spectral density matrix. *Journal of the American Statistical Association*, **120**, 1960–1974. doi:10.1080/01621459.2025.2468013.

See Also

`SpecTest`

Examples

```
# Example 1
## Generate xt
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

## Generate the sets Iq and Jq
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)
```

```
## Calculate Q p-values
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
}
res <- SpecMulTest(Q, PVal)
res
```

`SpecTest` *Global testing for spectral density matrix*

Description

`SpecTest()` implements a new global test proposed in Chang et al. (2022) for the following hypothesis testing problem:

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

where $f_{i,j}(\omega)$ represents the cross-spectral density between $x_{t,i}$ and $x_{t,j}$ at frequency ω with $x_{t,i}$ being the i-th component of the $p \times 1$ times series $\mathbf{x}_t$. Here, $\mathcal{I}$ is the set of index pairs of interest, and $\mathcal{J}$ is the set of frequencies.

Usage

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

Arguments

`X`	An $n \times p$ data matrix $\mathbf{X} = (\mathbf{x}_1, \ldots, \mathbf{x}_n)'$, where n is the number of observations of the $p \times 1$ time series $\{\mathbf{x}_t\}_{t=1}^n$.
`J.set`	A vector representing the set $\mathcal{J}$ of frequencies.
`cross.indices`	An $r \times 2$ matrix representing the set $\mathcal{I}$ of r index pairs, where each row represents an index pair.
`B`	The number of bootstrap replications for generating multivariate normally distributed random vectors when calculating the critical value. The default is 2000.
`flag_c`	The bandwidth $c \in (0, 1]$ of the flat-top kernel for estimating $f_{i,j}(\omega)$ [See (2) in Chang et al. (2022)]. The default is 0.8.

Value

An object of class `"hdtstest"`, which contains the following components:

`Stat`	The test statistic of the test.
`pval`	The p-value of the test.
`cri95`	The critical value of the test at the significance level 0.05.

References

Chang, J., Jiang, Q., McElroy, T. S., & Shao, X. (2025). Statistical inference for high-dimensional spectral density matrix. *Journal of the American Statistical Association*, **120**, 1960–1974. doi:10.1080/01621459.2025.2468013.

See Also

SpecMulTest

Examples

```
# Example 1
## Generate xt
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)

## Generate the sets I and J
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$statistic
Pvalue <- res$p.value
CriVal <- res$cri95
```

UR_test *Testing for unit roots based on sample autocovariances*

Description

This function implements the test proposed in Chang, Cheng and Yao (2022) for the following hypothesis testing problem:

$$H_0 : Y_t \sim I(0) \text{ versus } H_1 : Y_t \sim I(d) \text{ for some integer } d \geq 1\,,$$

where Y_t is a univariate time series.

Usage

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

Arguments

`Y`	A vector $\mathbf{Y} = (Y_1, \ldots, Y_n)'$, where n is the number of the observations.
`lagk.vec`	The time lag K_0 used to calculate the test statistic [See Section 2.1 of Chang, Cheng and Yao (2022)]. It can be a vector specifying multiple time lags. If provided as a $s \times 1$ vector, the function will output the test results corresponding to each of the s values in `lagk.vec`. The default is `c(0, 1, 2, 3, 4)`.
`con_vec`	The constant c_κ specified in (5) of Chang, Cheng and Yao (2022). The default is 0.55. Alternatively, it can be an $m \times 1$ vector specified by users, representing m candidate values of c_κ.
`alpha`	The significance level of the test. The default is 0.05.

Value

An object of class `"urtest"`, which contains the following components:

`statistic`	A $s \times 1$ vector with each element representing the test statistic value associated with each of the s time lags specified in `lagk.vec`.
`reject`	An $m \times s$ data matrix $\mathbf{R} = (R_{i,j})$ where $R_{i,j}$ represents whether the null hypothesis H_0 should be rejected for c_κ specified by the i-th component of `con_vec`, and K_0 specified by the j-th component of `lagk.vec`. $R_{i,j} = 1$ indicates rejection of the null hypothesis, while $R_{i,j} = 0$ indicates non-rejection.
`lag.vec`	The time lags used in function.

References

Chang, J., Cheng, G., & Yao, Q. (2022). Testing for unit roots based on sample autocovariances. *Biometrika*, **109**, 543–550. doi:10.1093/biomet/asab034.

Examples

```
# Example 1
## Generate yt
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()` implements the white noise tests proposed in Chang, Yao and Zhou (2017) and Chang et al. (2026+) for the following hypothesis testing problem:

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

Usage

```
WN_test(
  Y,
  lag.k = 2,
  B = 1000,
  method = c("L_inf", "L_2"),
  kernel.type = c("QS", "Par", "Bart"),
  pre = FALSE,
  alpha = 0.05,
  control.PCA = list()
)
```

Arguments

`Y`	An $n \times p$ data matrix $\mathbf{Y} = (\mathbf{y}_1, \ldots, \mathbf{y}_n)'$, where n is the number of the observations of the $p \times 1$ time series $\{\mathbf{y}_t\}_{t=1}^n$.
`lag.k`	The time lag K used to calculate the test statistic [See (4) of Chang, Yao and Zhou (2017) and (6) of Chang et al. (2026+).]. The default is 2.
`B`	The number of bootstrap replications for calculating the critical value. The default is 1000.
`method`	The option for method used in the white noise test. Available options include: `"L_inf"` (the default) for the method proposed by Chang, Yao and Zhou (2017), and `"L_2"` for the method proposed by Chang et al. (2026+).
`kernel.type`	The option for choosing the symmetric kernel used in the estimation of long-run covariance matrix, which is used when `method = "L_inf"`. Available options include: `"QS"` (the default) for the Quadratic spectral kernel, `"Par"` for the Parzen kernel, and `"Bart"` for the Bartlett kernel. See Chang, Yao and Zhou (2017) for more information.
`pre`	Logical. This parameter is used when `method = "L_inf"`. If `TRUE` (the default), the time series PCA proposed in Chang, Guo and Yao (2018) should be performed on $\{\mathbf{y}_t\}_{t=1}^n$ before implementing the white noise test [See Remark 1 of Chang, Yao and Zhou (2017)]. The time series PCA is implemented by using the function `PCA_TS` with the arguments passed by `control.PCA`.
`alpha`	The significance level of the test. The default is 0.05.

`control.PCA` A list of control arguments passed to the function `PCA_TS` when `method = "L_inf"` and `pre = TRUE`, including `lag.k`, `opt`, `thresh`, `delta`, and the associated arguments passed to the `clime` function (when `opt = 2`). See 'Details' in `PCA_TS`.

Value

An object of class `"hdtstest"`, which contains the following components:

`statistic` The test statistic of the test.

`p.value` The p-value of the test.

`lag.k` The time lag used in function.

`kernel.type` The kernel used in function.

References

Chang, J., He, J., Li, W., & Lin, C. (2026+). An adaptive L_2-type test for high-dimensional white noise. *Preprint*.

Chang, J., Guo, B., & Yao, Q. (2018). Principal component analysis for second-order stationary vector time series. *The Annals of Statistics*, **46**, 2094–2124. doi:10.1214/17AOS1613.

Chang, J., Yao, Q., & Zhou, W. (2017). Testing for high-dimensional white noise using maximum cross-correlations. *Biometrika*, **104**, 111–127. doi:10.1093/biomet/asw066.

See Also

`PCA_TS`

Examples

```
#Example 1
## Generate data
n <- 200
p <- 10
Y <- matrix(rnorm(n * p), n, p)

## L_inf
res1 <- WN_test(Y, method ="L_inf")
Pvalue1 <- res1$p.value
statistic1 <- res1$statistic

## L_2
res2 <- WN_test(Y, method = "L_2")
Pvalue2 <- res2$p.value
statistic2 <- res2$statistic
```

Index

∗ **data**
BeijingAir, 3
FamaFrench, 18
IPindices, 20
QWIdata, 31

BeijingAir, 3

Coint, 4
CP_Inference, 6
CP_MTS, 8, *16*, *28, 29*
CP_TTS, 12

DGP.CP, 15

Factors, 16, *19, 20*, *27*
FamaFrench, 18

HDSReg, 18

IPindices, 20

MartG_test, 21

PCA_TS, 23, *30*, *36, 37*
predict.factors, 26
predict.mtscp, 28
predict.tspca, 29

QWIdata, 31

SpecMulTest, 31, *34*
SpecTest, *32*, 33

UR_test, 34

WN_test, 36