Package ‘freqdom’

October 13, 2022

Type Package

Title Frequency Domain Based Analysis: Dynamic PCA

Version 2.0.3

Date 2022-10-04

Author Hormann S., Kidzinski L.

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

Description Implementation of dynamic principal component analysis (DPCA), simulation of VAR and VMA processes and frequency domain tools. These frequency domain methods for dimensionality reduction of multivariate time series were introduced by David Brillinger in his book Time Series (1974). We follow implementation guidelines as described in Hormann, Kidzinski and Hallin (2016), Dynamic Functional Principal Component <doi:10.1111/rssb.12076>.

Depends R (>= 2.15.0), mvtnorm, stats, graphics, base, matrixcalc, utils

Suggests fda, MASS, MARSS, testthat (>= 3.0.0)

RoxygenNote 7.2.1

Config/testthat/edition 3

NeedsCompilation no

Repository CRAN

Date/Publication 2022-10-04 09:50:07 UTC

R topics documented:

`freqdom-package` *Frequency domain basde analysis: dynamic PCA*

Description

Implementation of dynamic principle component analysis (DPCA), simulation of VAR and VMA processes and frequency domain tools. The package also provides a toolset for developers simplifying construction of new frequency domain based methods for multivariate signals.

Details

freqdom package allows you to manipulate time series objects in both time and frequency domains. We implement dynamic principal component analysis methods, enabling spectral decomposition of a stationary vector time series into uncorrelated components.

Dynamic principal component analysis enables estimation of temporal filters which transform a vector time series into another vector time series with uncorrelated components, maximizing the long run variance explained. There are two key differnces between classical PCA and dynamic PCA:

- Components returned by the dynamic procedure are uncorrelated in time, i.e. for any $i \neq j$ and $l \in Z$, $Y_i(t)$ and $Y_j(t_l)$ are uncorrelated,
- The mapping maximizes the long run variance, which, in case of stationary vector time series, means that the process reconstructed from and $d > 0$ first dynamic principal components better approximates your vector time series process than the first d classic principal components.

For details, please refer to literature below and to help pages of functions `dpca` for estimating the components, `dpca.scores` for estimating scores and `dpca.KLexpansion` for retrieving the signal from components.

Apart from frequency domain techniques for stationary vector time series, **freqdom** provides a toolset of operators such as the vector Fourier Transform (`fourier.transform`) or a vector spectral density operator (`spectral.density`) as well as simulation of vector time series models `rar`, `rma` generating vector autoregressive and moving average respectively. These functions enable developing new techniques based on the Frequency domain analysis.

References

Hormann Siegfried, Kidzinski Lukasz and Hallin Marc. *Dynamic functional principal components.* Journal of the Royal Statistical Society: Series B (Statistical Methodology) 77.2 (2015): 319-348.

Hormann Siegfried, Kidzinski Lukasz and Kokoszka Piotr. *Estimation in functional lagged regression.* Journal of Time Series Analysis 36.4 (2015): 541-561.

Hormann Siegfried and Kidzinski Lukasz. *A note on estimation in Hilbertian linear models.* Scandinavian journal of statistics 42.1 (2015): 43-62.

`cov.structure` *Estimate cross-covariances of two stationary multivariate time series*

Description

This function computes the empirical cross-covariance of two stationary multivariate time series. If only one time series is provided it determines the empirical autocovariance function.

Usage

```
cov.structure(X, Y = X, lags = 0)
```

Arguments

`X`	vector or matrix. If matrix, then each row corresponds to a timepoint of a vector time series.
`Y`	vector or matrix. If matrix, then each row corresponds to a timepoint of a vector time series.
`lags`	an integer-valued vector $(\ell_1, \ldots, \ell_K)$ containing the lags for which covariances are calculated.

Details

Let $[X_1, \ldots, X_T]'$ be a $T \times d_1$ matrix and $[Y_1, \ldots, Y_T]'$ be a $T \times d_2$ matrix. We stack the vectors and assume that $(X_t', Y_t')'$ is a stationary multivariate time series of dimension $d_1 + d_2$. This function determines empirical lagged covariances between the series (X_t) and (Y_t). More precisely it determines $\widehat{C}^{XY}(h)$ for $h \in$ lags, where $\widehat{C}^{XY}(h)$ is the empirical version of $\mathrm{Cov}(X_h, Y_0)$. For a sample of size T we set $\hat{\mu}^X = \frac{1}{T}\sum_{t=1}^T X_t$ and $\hat{\mu}^Y = \frac{1}{T}\sum_{t=1}^T Y_t$ and

$$\hat{C}^{XY}(h) = \frac{1}{T}\sum_{t=1}^{T-h}(X_{t+h} - \hat{\mu}^X)(Y_t - \hat{\mu}^Y)'$$

and for $h < 0$

$$\hat{C}^{XY}(h) = \frac{1}{T}\sum_{t=|h|+1}^{T}(X_{t+h} - \hat{\mu}^X)(Y_t - \hat{\mu}^Y)'.$$

Value

An object of class `timedom`. The list contains

- `operators` an array. Element `[,,k]` contains the covariance matrix related to lag ℓ_k.
- `lags` returns the lags vector from the arguments.

`dpca` *Compute Dynamic Principal Components and dynamic Karhunen Loeve extepansion*

Description

Dynamic principal component analysis (DPCA) decomposes multivariate time series into uncorrelated components. Compared to classical principal components, DPCA decomposition outputs components which are uncorrelated in time, allowing simpler modeling of the processes and maximizing long run variance of the projection.

Usage

```
dpca(X, q = 30, freq = (-1000:1000/1000) * pi, Ndpc = dim(X)[2])
```

Arguments

`X`	a vector time series given as a $(T \times d)$-matix. Each row corresponds to a timepoint.
`q`	window size for the kernel estimator, i.e. a positive integer.
`freq`	a vector containing frequencies in $[-\pi, \pi]$ on which the spectral density should be evaluated.
`Ndpc`	is the number of principal component filters to compute as in `dpca.filters`

Details

This convenience function applies the DPCA methodology and returns filters (`dpca.filters`), scores (`dpca.scores`), the spectral density (`spectral.density`), variances (`dpca.var`) and Karhunen-Leove expansion (`dpca.KLexpansion`).

See the example for understanding usage, and help pages for details on individual functions.

Value

A list containing

- `scores` DPCA scores (`dpca.scores`)
- `filters` DPCA filters (`dpca.filters`)
- `spec.density` spectral density of `X` (`spectral.density`)
- `var` amount of variance explained by dynamic principal components (`dpca.var`)
- `Xhat` Karhunen-Loeve expansion using `Ndpc` dynamic principal components (`dpca.KLexpansion`)

References

Hormann, S., Kidzinski, L., and Hallin, M. *Dynamic functional principal components.* Journal of the Royal Statistical Society: Series B (Statistical Methodology) 77.2 (2015): 319-348.

Brillinger, D. *Time Series* (2001), SIAM, San Francisco.

Shumway, R., and Stoffer, D. *Time series analysis and its applications: with R examples* (2010), Springer Science & Business Media

Examples

```
X = rar(100,3)

# Compute DPCA with only one component
res.dpca = dpca(X, q = 5, Ndpc = 1)

# Compute PCA with only one component
res.pca = prcomp(X, center = TRUE)
res.pca$x[,-1] = 0

# Reconstruct the data
var.dpca = (1 - sum( (res.dpca$Xhat - X)**2 ) / sum(X**2))*100
var.pca = (1 - sum( (res.pca$x %*% t(res.pca$rotation) - X)**2 ) / sum(X**2))*100

cat("Variance explained by DPCA:\t",var.dpca,"%\n")
cat("Variance explained by PCA:\t",var.pca,"%\n")
```

`dpca.filters` *Compute DPCA filter coefficients*

Description

For a given spectral density matrix dynamic principal component filter sequences are computed.

Usage

```
dpca.filters(F, Ndpc = dim(F$operators)[1], q = 30)
```

Arguments

`F`	$(d \times d)$ spectral density matrix, provided as an object of class `freqdom`.
`Ndpc`	an integer $\in \{1, \dots, d\}$. It is the number of dynamic principal components to be computed. By default it is set equal to d.
`q`	a non-negative integer. DPCA filter coefficients at lags $\|h\| \leq$ `q` will be computed.

Details

Dynamic principal components are linear filters $(\phi_{\ell k}\colon k \in \mathbf{Z})$, $1 \leq \ell \leq d$. They are defined as the Fourier coefficients of the dynamic eigenvector $\varphi_\ell(\omega)$ of a spectral density matrix $\mathcal{F}_\omega$:

$$\phi_{\ell k} := \frac{1}{2\pi}\int_{-\pi}^{\pi} \varphi_\ell(\omega)\exp(-ik\omega)d\omega.$$

The index ℓ is referring to the ℓ-th #'largest dynamic eigenvalue. Since the $\phi_{\ell k}$ are real, we have

$$\phi_{\ell k}' = \phi_{\ell k}^* = \frac{1}{2\pi}\int_{-\pi}^{\pi} \varphi_\ell^* \exp(ik\omega)d\omega.$$

For a given spectral density (provided as on object of class `freqdom`) the function `dpca.filters()` computes $(\phi_{\ell k})$ for $|k| \leq$ `q` and $1 \leq \ell \leq$ `Ndpc`.

For more details we refer to Chapter 9 in Brillinger (2001), Chapter 7.8 in Shumway and Stoffer (2006) and to Hormann et al. (2015).

Value

An object of class `timedom`. The list has the following components:

- `operators` an array. Each matrix in this array has dimension `Ndpc` $\times d$ and is assigned to a certain lag. For a given lag k, the rows of the matrix correpsond to $\phi_{\ell k}$.
- `lags` a vector with the lags of the filter coefficients.

References

Hormann, S., Kidzinski, L., and Hallin, M. *Dynamic functional principal components.* Journal of the Royal Statistical Society: Series B (Statistical Methodology) 77.2 (2015): 319-348.

Brillinger, D. *Time Series* (2001), SIAM, San Francisco.

Shumway, R.H., and Stoffer, D.S. *Time Series Analysis and Its Applications* (2006), Springer, New York.

See Also

`dpca.var`, `dpca.scores`, `dpca.KLexpansion`

`dpca.KLexpansion` *Dynamic KL expansion*

Description

Computes the dynamic Karhunen-Loeve expansion of a vector time series up to a given order.

Usage

```
dpca.KLexpansion(X, dpcs)
```

Arguments

`X`	a vector time series given as a $(T \times d)$-matix. Each row corresponds to a timepoint.
`dpcs`	an object of class `timedom`, representing the dpca filters obtained from the sample X. If `dpsc = NULL`, then `dpcs = dpca.filter(spectral.density(X))` is used.

Details

We obtain the dynamic Karhnunen-Loeve expansion of order L, $1 \le L \le d$. It is defined as

$$\sum_{\ell=1}^{L} \sum_{k \in \mathbf{Z}} Y_{\ell,t+k} \phi_{\ell k},$$

where $\phi_{\ell k}$ are the dynamic PC filters as explained in `dpca.filters` and $Y_{\ell k}$ are dynamic scores as explained in `dpca.scores`. For the sample version the sum in k extends over the range of lags for which the $\phi_{\ell k}$ are defined.

For more details we refer to Chapter 9 in Brillinger (2001), Chapter 7.8 in Shumway and Stoffer (2006) and to Hormann et al. (2015).

Value

A $(T \times d)$-matix. The ℓ-th column contains the ℓ-th data point.

References

Hormann, S., Kidzinski, L., and Hallin, M. *Dynamic functional principal components.* Journal of the Royal Statistical Society: Series B (Statistical Methodology) 77.2 (2015): 319-348.

Brillinger, D. *Time Series* (2001), SIAM, San Francisco.

Shumway, R.H., and Stoffer, D.S. *Time Series Analysis and Its Applications* (2006), Springer, New York.

See Also

`dpca.filters`, `filter.process`, `dpca.scores`

`dpca.scores` *Obtain dynamic principal components scores*

Description

Computes dynamic principal component score vectors of a vector time series.

Usage

```
dpca.scores(X, dpcs = dpca.filters(spectral.density(X)))
```

Arguments

`X` a vector time series given as a $(T \times d)$-matix. Each row corresponds to a timepoint.

`dpcs` an object of class `timedom`, representing the dpca filters obtained from the sample X. If `dpsc = NULL`, then `dpcs = dpca.filter(spectral.density(X))` is used.

Details

The ℓ-th dynamic principal components score sequence is defined by

$$Y_{\ell t} := \sum_{k \in \mathbf{Z}} \phi'_{\ell k} X_{t-k}, \quad 1 \leq \ell \leq d,$$

where $\phi_{\ell k}$ are the dynamic PC filters as explained in `dpca.filters`. For the sample version the sum extends over the range of lags for which the $\phi_{\ell k}$ are defined. The actual operation carried out is `filter.process(X, A = dpcs)`.

We for more details we refer to Chapter 9 in Brillinger (2001), Chapter 7.8 in Shumway and Stoffer (2006) and to Hormann et al. (2015).

Value

A $T\times$ `Ndpc`-matix with `Ndpc = dim(dpcs$operators)[1]`. The ℓ-th column contains the ℓ-th dynamic principal component score sequence.

References

Hormann, S., Kidzinski, L., and Hallin, M. *Dynamic functional principal components.* Journal of the Royal Statistical Society: Series B (Statistical Methodology) 77.2 (2015): 319-348.

Brillinger, D. *Time Series* (2001), SIAM, San Francisco.

Shumway, R.H., and Stoffer, D.S. *Time Series Analysis and Its Applications* (2006), Springer, New York.

See Also

`dpca.filters`, `dpca.KLexpansion`, `dpca.var`

`dpca.var` *Proportion of variance explained*

Description

Computes the proportion of variance explained by a given dynamic principal component.

Usage

```
dpca.var(F)
```

Arguments

F — $(d \times d)$ spectral density matrix, provided as an object of class `freqdom`. To guarantee accuracy of numerical integration it is important that `F$freq` is a dense grid of frequencies in $[-\pi, \pi]$.

Details

Consider a spectral density matrix $\mathcal{F}_\omega$ and let $\lambda_\ell(\omega)$ by the ℓ-th dynamic eigenvalue. The proportion of variance described by the ℓ-th dynamic principal component is given as

$$v_\ell := \int_{-\pi}^{\pi} \lambda_\ell(\omega) d\omega / \int_{-\pi}^{\pi} \mathrm{tr}(\mathcal{F}_\omega) d\omega.$$

This function numerically computes the vectors $(v_\ell \colon 1 \leq \ell \leq d)$.

For more details we refer to Chapter 9 in Brillinger (2001), Chapter 7.8 in Shumway and Stoffer (2006) and to Hormann et al. (2015).

Value

A d-dimensional vector containing the v_ℓ.

References

Hormann, S., Kidzinski, L., and Hallin, M. *Dynamic functional principal components.* Journal of the Royal Statistical Society: Series B (Statistical Methodology) 77.2 (2015): 319-348.

Brillinger, D. *Time Series* (2001), SIAM, San Francisco.

Shumway, R.H., and Stoffer, D.S. *Time Series Analysis and Its Applications* (2006), Springer, New York.

See Also

`dpca.filters`, `dpca.KLexpansion`, `dpca.scores`

`filter.process` *Convolute (filter) a multivariate time series using a time-domain filter*

Description

This function applies a linear filter to some vector time series.

Usage

```
filter.process(X, A)

X %c% A
```

Arguments

`X`	vector time series given in matrix form. Each row corresponds to a timepoint.
`A`	an object of class `timedom`.

Details

Let $[X_1, \ldots, X_T]'$ be a $T \times d$ matrix corresponding to a vector series $X_1, \ldots, X_T$. This time series is transformed to the series $Y_1, \ldots, Y_T$, where

$$Y_t = \sum_{k=-q}^{p} A_k X_{t-k}, \quad t \in \{p+1, \ldots, T-q\}.$$

The index k of A_k is determined by the lags defined for the time domain object. When index $t-k$ falls outside the domain $\{1, \ldots, T\}$ we set $X_t = \frac{1}{T}\sum_{k=1}^{T} X_k$.

Value

A matrix. Row t corresponds to Y_t.

Functions

- `filter.process()`: Multivariate convolution (filter) in the time domain
- `X %c% A`: Convenience operator for `filter.process` function

See Also

`timedom`

`fourier.inverse` *Coefficients of a discrete Fourier transform*

Description

Computes Fourier coefficients of some functional represented by an object of class `freqdom`.

Usage

```
fourier.inverse(F, lags = 0)
```

Arguments

`F`	an object of class `freqdom` which is corresponding to a function with values in $\mathbf{C}^{d_1 \times d_2}$. To guarantee accuracy of inversion it is important that `F$freq` is a dense grid of frequencies in $[-\pi, \pi]$.
`lags`	lags of the Fourier coefficients to be computed.

Details

Consider a function $F\colon [-\pi,\pi] \to \mathbf{C}^{d_1\times d_2}$. Its k-th Fourier coefficient is given as

$$\frac{1}{2\pi}\int_{-\pi}^{\pi} F(\omega)\exp(ik\omega)d\omega.$$

We represent the function F by an object of class `freqdom` and approximate the integral via

$$\frac{1}{|F\$freq|}\sum_{\omega\in F\$freq} F(\omega)\exp(ik\omega),$$

for $k \in$ lags.

Value

An object of class timedom. The list has the following components:

- `operators` an array. The k-th matrix in this array corresponds to the k-th Fourier coefficient.
- `lags` the lags of the corresponding Fourier coefficients.

See Also

fourier.transform, freqdom

Examples

```
Y = rar(100)
grid = c(pi*(1:2000) / 1000 - pi) #a dense grid on -pi, pi
fourier.inverse(spectral.density(Y, q=2, freq=grid))

# compare this to
cov.structure(Y)
```

`fourier.transform` *Computes the Fourier transformation of a filter given as* `timedom` *object*

Description

Computes the frequency response function of a linear filter and returns it as a freqdom object.

Usage

```
fourier.transform(A, freq = pi * -100:100/100)
```

Arguments

`A`	an object of class `timedom`.
`freq`	a vector of frequencies $\in [-\pi,\pi]$.

Details

Consider a filter (a sequence of vectors or matrices) $(A_k)_{k \in A\$lags}$. Then this function computes

$$\sum_{k \in A\$lags} A_k e^{-ik\omega}$$

for all frequencies ω listed in the vector `freq`.

Value

An object of class `freqdom`.

See Also

`fourier.inverse`

Examples

```
# We compute the discrete Fourier transform (DFT) of a time series X_1,..., X_T.

X = rar(100)
T=dim(X)[1]
tdX = timedom(X/sqrt(T),lags=1:T)
DFT = fourier.transform(tdX, freq= pi*-1000:1000/1000)
```

---

freqdom — *Create an object corresponding to a frequency domain functional*

Description

Creates an object of class `freqdom`. This object corresponds to a functional with domain $[-\pi, \pi]$ and some complex vector space as codomain.

Usage

```
freqdom(F, freq)
```

Arguments

`F`	a vector, a matrix or an array. For vectors $F[k], 1 \leq k \leq K$ are complex numbers. For matrices $F[k,]$ are complex vectors. For arrays the elements $F[,,k]$, are complex valued $(d_1 \times d_2)$ matrices (all of same dimension).
`freq`	a vector of dimension K containing frequencies in $[-\pi, \pi]$.

Details

This class is used to describe a frequency domain functional (like a spectral density matrix, a discrete Fourier transform, an impulse response function, etc.) on selected frequencies. Formally we consider a collection $[F_1, \ldots, F_K]$ of complex-valued matrices F_k, all of which have the same dimension $d_1 \times d_2$. Moreover, we consider frequencies $\{\omega_1, \ldots, \omega_K\} \subset [-\pi, \pi]$. The object this function creates corresponds to the mapping $f : \mathrm{freq} \to \mathbf{C}^{d_1 \times d_2}$, where $\omega_k \mapsto F_k$.

Consider, for example, the discrete Fourier transform of a vector time series $X_1, \ldots, X_T$:. It is defined as

$$D_T(\omega) = \frac{1}{\sqrt{T}} \sum_{t=1}^{T} X_t e^{-it\omega}, \quad \omega \in [-\pi, \pi].$$

We may choose $\omega_k = 2\pi k/K - \pi$ and $F_k = D_T(\omega_k)$. Then, the object `freqdom` creates, is corresponding to the function which associates ω_k and $D_T(\omega_k)$.

Value

Returns an object of class `freqdom`. An object of class `freqdom` is a list containing the following components:

- `operators` the array `F` as given in the argument.
- `freq` the vector `freq` as given in the argument.

See Also

`fourier.transform`

Examples

```
i = complex(imaginary=1)
OP = array(0, c(2, 2, 3))
OP[,,1] = diag(2) * exp(i)/2
OP[,,2] = diag(2)
OP[,,3] = diag(2) * exp(-i)/2
freq = c(-pi/3, 0, pi/3)
A = freqdom(OP, freq)
```

`freqdom.eigen` *Eigendecompose a frequency domain operator at each frequency*

Description

Gives the eigendecomposition of objects of class `freqdom`.

Usage

```
freqdom.eigen(F)
```

Arguments

`F`	an object of class freqdom. The matrices `F\$operator[,,k]` are required to be square matrices, say $d \times d$.

Details

This function makes an eigendecomposition for each of the matrices `F\$operator[,,k]`.

Value

Returns a list. The list is containing the following components:

- `vectors` an array containing d matrices. The i-th matrix contains in its k-th row the conjugate transpose eigenvector belonging to the k-th largest eigenvalue of `F\$operator[,,i]`.
- `values` matrix containing in k-th column the eigenvalues of `F\$operator[,,k]`.
- `freq` vector of frequencies defining the object `F`.

See Also

`freqdom`

`is.freqdom` *Checks if an object belongs to the class freqdom*

Description

Checks if an object belongs to the class `freqdom`.

Usage

```
is.freqdom(X)
```

Arguments

`X`	some object

Value

`TRUE` if `X` is of type `freqdom`, `FALSE` otherwise

See Also

`freqdom`, `timedom`, `is.timedom`

is.timedom — *Checks if an object belongs to the class timedom*

Description

Checks if an object belongs to the class `timedom`.

Usage

```
is.timedom(X)
```

Arguments

`X`	some object

Value

`TRUE` if `X` is of type `timedom`, `FALSE` otherwise

See Also

`freqdom`, `timedom`, `is.freqdom`

rar — *Simulate a multivariate autoregressive time series*

Description

Generates a zero mean vector autoregressive process of a given order.

Usage

```
rar(
  n,
  d = 2,
  Psi = NULL,
  burnin = 10,
  noise = c("mnormal", "mt"),
  sigma = NULL,
  df = 4
)
```

Arguments

`n`	number of observations to generate.
`d`	dimension of the time series.
`Psi`	array of $p \geq 1$ coefficient matrices. `Psi[,,k]` is the k-th coefficient. If no value is set then we generate a vector autoregressive process of order 1. Then, `Psi[,,1]` is proportional to $\exp(-(i+j)\colon 1 \leq i,j \leq d)$ and such that the spectral radius of `Psi[,,1]` is 1/2.
`burnin`	an integer ≥ 0. It specifies a number of initial observations to be trashed to achieve stationarity.
`noise`	`mnormal` for multivariate normal noise or `mt` for multivariate student t noise. If not specified `mnormal` is chosen.
`sigma`	covariance or scale matrix of the innovations. By default the identity matrix.
`df`	degrees of freedom if `noise = "mt"`.

Details

We simulate a vector autoregressive process

$$X_t = \sum_{k=1}^{p} \Psi_k X_{t-k} + \varepsilon_t, \quad 1 \leq t \leq n.$$

The innovation process ε_t is either multivariate normal or multivariate t with a predefined covariance/scale matrix sigma and zero mean. The noise is generated with the package `mvtnorm`. For Gaussian noise we use `rmvnorm`. For Student-t noise we use `rmvt`. The parameters sigma and df are imported as arguments, otherwise we use default settings. To initialise the process we set $[X_{1-p},\ldots,X_0] = [\varepsilon_{1-p},\ldots,\varepsilon_0]$. When `burnin` is set equal to K then, n+K observations are generated and the first K will be trashed.

Value

A matrix with `d` columns and `n` rows. Each row corresponds to one time point.

See Also

`rma`

`rma` *Moving average process*

Description

Generates a zero mean vector moving average process.

Usage

```
rma(n, d = 2, Psi = NULL, noise = c("mnormal", "mt"), sigma = NULL, df = 4)
```

Arguments

n	number of observations to generate.
d	dimension of the time series.
Psi	a timedom object with operators Psi$operators, where Psi$operators[,,k] is the operator on thelag lags[k]. If no value is set then we generate a vector moving average process of order 1. Then, Psi$lags = c(1) and Psi$operators[,,1] is proportional to $\exp(-(i+j)\colon 1 \leq i,j \leq d)$ and such that the spectral radius of Psi[,,1] is $1/2$.
noise	mnormal for multivariate normal noise or mt for multivariate t noise. If not specified mnormal is chosen.
sigma	covariance or scale matrix of the innovations. If NULL then the identity matrix is used.
df	degrees of freedom if noise = "mt".

Details

This simulates a vector moving average process

$$X_t = \varepsilon_t + \sum_{k \in lags} \Psi_k \varepsilon_{t-k}, \quad 1 \leq t \leq n.$$

The innovation process ε_t is either multivariate normal or multivarite t with a predefined covariance/scale matrix sigma and zero mean. The noise is generated with the package mvtnorm. For Gaussian noise we use rmvnorm. For Student-t noise we use rmvt. The parameters sigma and df are imported as arguments, otherwise we use default settings.

Value

A matrix with d columns and n rows. Each row corresponds to one time point.

See Also

rar

---

spectral.density *Compute empirical spectral density*

Description

Estimates the spectral density and cross spectral density of vector time series.

Usage

```
spectral.density(
  X,
  Y = X,
  freq = (-1000:1000/1000) * pi,
  q = max(1, floor(dim(X)[1]^(1/3))),
  weights = c("Bartlett", "trunc", "Tukey", "Parzen", "Bohman", "Daniell",
    "ParzenCogburnDavis")
)
```

Arguments

`X`	a vector or a vector time series given in matrix form. Each row corresponds to a timepoint.
`Y`	a vector or vector time series given in matrix form. Each row corresponds to a timepoint.
`freq`	a vector containing frequencies in $[-\pi, \pi]$ on which the spectral density should be evaluated.
`q`	window size for the kernel estimator, i.e. a positive integer.
`weights`	kernel used in the spectral smoothing. By default the Bartlett kernel is chosen.

Details

Let $[X_1, \ldots, X_T]'$ be a $T \times d_1$ matrix and $[Y_1, \ldots, Y_T]'$ be a $T \times d_2$ matrix. We stack the vectors and assume that $(X_t', Y_t')'$ is a stationary multivariate time series of dimension $d_1 + d_2$. The cross-spectral density between the two time series (X_t) and (Y_t) is defined as

$$\sum_{h \in \mathbf{Z}} \mathrm{Cov}(X_h, Y_0) e^{-ih\omega}.$$

The function `spectral.density` determines the empirical cross-spectral density between the two time series (X_t) and (Y_t). The estimator is of form

$$\widehat{\mathcal{F}}^{XY}(\omega) = \sum_{|h| \leq q} w(|k|/q) \widehat{C}^{XY}(h) e^{-ih\omega},$$

with $\widehat{C}^{XY}(h)$ defined in `cov.structure` Here w is a kernel of the specified type and q is the window size. By default the Bartlett kernel $w(x) = 1 - |x|$ is used.

See, e.g., Chapter 10 and 11 in Brockwell and Davis (1991) for details.

Value

Returns an object of class `freqdom`. The list is containing the following components:

- `operators` an array. The k-th matrix in this array corresponds to the spectral density matrix evaluated at the k-th frequency listed in `freq`.
- `freq` returns argument vector `freq`.

References

Peter J. Brockwell and Richard A. Davis *Time Series: Theory and Methods* Springer Series in Statistics, 2009

`timedom` *Defines a linear filter*

Description

Creates an object of class `timedom`. This object corresponds to a multivariate linear filter.

Usage

```
timedom(A, lags)
```

Arguments

`A`	a vector, matrix or array. If array, the elements $A[,,k], 1 \leq k \leq K$, are real valued $(d_1 \times d_2)$ matrices (all of same dimension). If A is a matrix, the k-th row is treated as $A[,,k]$. Same for the k-th element of a vector. These matrices, vectors or scalars define a linear filter.
`lags`	a vector of increasing integers. It corresponds to the time lags of the filter.

Details

This class is used to describe a linear filter, i.e. a sequence of matrices, each of which correspond to a certain lag. Filters can, for example, be used to transform a sequence (X_t) into a new sequence (Y_t) by defining

$$Y_t = \sum_k A_k X_{t-k}.$$

See `filter.process()`. Formally we consider a collection $[A_1, \ldots, A_K]$ of complex-valued matrices A_k, all of which have the same dimension $d_1 \times d_2$. Moreover, we consider lags $\ell_1 < \ell_2 < \cdots < \ell_K$. The object this function creates corresponds to the mapping $f : \text{lags} \to \mathbf{R}^{d_1 \times d_2}$, where $\ell_k \mapsto A_k$.

Value

Returns an object of class `timedom`. An object of class `timedom` is a list containing the following components:

- `operators` returns the array `A` as given in the argument.
- `lags` returns the vector `lags` as given in the argument.

See Also

`freqdom`, `is.timedom`

Examples

```
# In this example we apply the difference operator: Delta X_t= X_t-X_{t-1} to a time series
X = rar(20)
OP = array(0,c(2,2,2))
OP[,,1] = diag(2)
OP[,,2] = -diag(2)
A = timedom(OP, lags = c(0,1))
filter.process(X, A)
```

`timedom.norms` *Compute operator norms of elements of a filter*

Description

This function determines the norms of the matrices defining some linear filter.

Usage

```
timedom.norms(A, type = "2")
```

Arguments

`A`	an object of class `timedom`
`type`	matrix norm to be used as in `norm`

Details

Computes $\|A_h\|$ for h in the set of lags belonging to the object A. When type is `2` then $\|A\|$ is the spectral radius of A. When type is `F` then $\|A\|$ is the Frobenius norm (or the Hilbert-Schmidt norm, or Schatten 2-norm) of A. Same options as for the function `norm` as in base package.

Value

A list which contains the following components:

- `lags` a vector containing the lags of `A`.
- `norms` a vector containing the norms of the matrices defining `A`.

Examples

```
d = 2

A = array(0,c(d,d,2))
A[1,,] = 2 * diag(d:1)/d
A[2,,] = 1.5 * diag(d:1)/d
OP = timedom(A,c(-2,1))
timedom.norms(OP)
```

timedom.trunc *Choose lags of an object of class* timedom

Description

This function removes lags from a linear filter.

Usage

```
timedom.trunc(A, lags)
```

Arguments

A	an object of class timedom.
lags	a vector which contains a set of lags. These lags must be a subset of the lags defined for timedom object A. Only those lags will be kept, the other lags are removed.

Value

An object of class timedom.

Index