Package 'freqdom.fda'

July 22, 2025

Type Package

Title Functional Time Series: Dynamic Functional Principal Components

Version 1.0.1

Date 2022-04-18

Author Hormann S., Kidzinski L.

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

Description Implementations of functional dynamic principle components analysis. Related graphic tools and frequency domain methods.
These methods directly use multivariate dynamic principal components implementation, following the guidelines from Hormann, Kidzinski and Hallin (2016), Dynamic Functional Principal Component <doi:10.1111/rssb.12076>.

License GPL-3

Depends R (>= 2.15.0), mvtnorm, stats, graphics, base, fda, freqdom

Suggests MASS, MARSS

RoxygenNote 7.1.2

LazyData true

NeedsCompilation no

Repository CRAN

Date/Publication 2022-04-19 00:04:34 UTC

Contents

`freqdom.fda-package` *Functional time series: dynamic FPCA*

Description

Implementation of dynamic functional principle component analysis (FDPCA), simulation of functional AR and functional MA processes and frequency domain tools for funcional data. The package is a wrapper for functionality of the multivariate package **freqdom** for applying frequency domain on objects from **fda**. Compared to **freqdom** some new visualization methods are added – adequate only if data has functional structure.

Details

fda.ts package allows you to analyse functional time series objects in both time and frequency domain. The main feature is dynamic functional principal component analysis. This method allows to transform a stationary functional time series into a vector process with mutually uncorrelated component processes.

There are two key differnces between classical PCA and dynamic PCA:

- Component processes returned by the dynamic procedure are mutually uncorrelated,
- The mapping maximizes the long run variance of compoments, which, in case of stationary functional time series, means that the process reconstructed from and $d > 0$ first dynamic principal components better approximates the original functional time series process than the first d classic principal components.

For functional data one can conveniently visualize properties of the filters, covariances or the spectral density operator.

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

The package **fda.ts** require the package **freqdom** provides the analogue multivariate toolset.

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.

`fts.cov.structure` *Estimate autocovariance and cross-covariances operators*

Description

This function is used to estimate a collection of cross-covariances operators of two stationary functional series.

Usage

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

Arguments

`X`	an object of class `fd` containing T functional observations.
`Y`	an object of class `fd` containing T functional observations.
`lags`	an integer-valued vector $(\ell_1, \ldots, \ell_K)$ containing the lags for which covariances are calculated.

Details

Let $X_1(u), \ldots, X_T(u)$ and $Y_1(u), \ldots, Y_T(u)$ be two samples of functional data. This function determines empirical lagged covariances between the series $(X_t(u))$ and $(Y_t(u))$. More precisely it determines

$$(\widehat{c}_h^{XY}(u,v)\colon h \in lags),$$

where $\widehat{c}_h^{XY}(u,v)$ is the empirical version of the covariance kernel $\mathrm{Cov}(X_h(u), Y_0(v))$. For a sample of size T we set $\hat{\mu}^X(u) = \frac{1}{T}\sum_{t=1}^T X_t(u)$ and $\hat{\mu}^Y(v) = \frac{1}{T}\sum_{t=1}^T Y_t(v)$. Now for $h \geq 0$

$$\frac{1}{T}\sum_{t=1}^{T-h}(X_{t+h}(u) - \hat{\mu}^X(u))(Y_t(v) - \hat{\mu}^Y(v))$$

and for $h < 0$

$$\frac{1}{T}\sum_{t=|h|+1}^{T}(X_{t+h}(u) - \hat{\mu}^X(u))(Y_t(v) - \hat{\mu}^Y(v)).$$

Since $X_t(u) = \boldsymbol{b}_1'(u)\mathbf{x}_t$ and $Y_t(u) = \mathbf{y}_t'\boldsymbol{b}_2(u)$ we can write

$$\widehat{c}_h^{XY}(u,v) = \boldsymbol{b}_1'(u)\widehat{C}^{\mathbf{xy}}\boldsymbol{b}_2(v),$$

where $\widehat{C}^{\mathbf{xy}}$ is defined as for the function "cov.structure" for series of coefficient vectors $(\mathbf{x}_t\colon 1 \leq t \leq T)$ and $(\mathbf{y}_t\colon 1 \leq t \leq T)$.

Value

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

- `operators` an array. Element `[,,k]` contains the covariance matrix of the coefficient vectors of the two time series related to lag ℓ_k.
- `lags` the lags vector from the arguments.
- `basisX` `X$basis`, an object of class `basis.fd` (see `create.basis`)
- `basisY` `Y$basis`, an object of class `basis.fd` (see `create.basis`)

See Also

The multivariate equivalent in the `freqdom` package: `cov.structure`

Examples

```
# Generate an autoregressive process
fts = fts.rar(d=3)

# Get covariance at lag 0
fts.cov.structure(fts, lags = 0)

# Get covariance at lag 10
fts.cov.structure(fts, lags = 10)

# Get entire covariance structure between -20 and 20
fts.cov.structure(fts, lags = -20:20)

# Compute covariance with another process
fts0 = fts + fts.rma(d=3)
fts.cov.structure(fts, fts0, lags = -2:2)
```

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

Description

Functional dynamic principal component analysis (FDPCA) decomposes functional time series to a vector time series with uncorrelated components. Compared to classical functional principal components, FDPCA decomposition outputs components which are uncorrelated in time, allowing simpler modeling of the processes and maximizing long run variance of the projection.

Usage

```
fts.dpca(X, q = 30, freq = (-1000:1000/1000) * pi, Ndpc = X$basis$nbasis)
```

Arguments

X	a functional time series as a fd object from fda package.
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 fts.dpca.filters

Details

This convenient function applies the FDPCA methodology and returns filters (fts.dpca.filters), scores (fts.dpca.scores), the spectral density (fts.spectral.density), variances (fts.dpca.var) and Karhunen-Leove expansion (fts.dpca.KLexpansion).

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

Value

A list containing

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

```
# Load example PM10 data from Graz, Austria
data(pm10) # loads functional time series pm10 to the environment
X = center.fd(pm10)

# Compute functional dynamic principal components with only one component
res.dpca = fts.dpca(X, Ndpc = 1, freq=(-25:25/25)*pi) # leave default freq for higher precision
plot(res.dpca$Xhat)
fts.plot.filters(res.dpca$filters)

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

```
# Compute empirical variance explained
var.dpca = (1 - sum( (res.dpca$Xhat$coefs - X$coefs)**2 ) / sum(X$coefs**2))*100
var.pca = (1 - sum( (res.pca$x %*% t(res.pca$rotation) - t(X$coefs) )**2 ) / sum(X$coefs**2))*100

cat("Variance explained by PCA (empirical):\t\t",var.pca,"%\n")
cat("Variance explained by PCA (theoretical):\t",
   (1 - (res.pca$sdev[1] / sum(res.pca$sdev)))*100,"%\n")
cat("Variance explained by DPCA (empirical):\t\t",var.dpca,"%\n")
cat("Variance explained by DPCA (theoretical):\t",(res.dpca$var[1])*100,"%\n")

# Plot filters
fts.plot.filters(res.dpca$filters)

# Plot spectral density (note that in case of these data it's concentrated around 0)
fts.plot.operators(res.dpca$spec.density,freq = c(-2,-3:3/30 * pi,2))

# Plot covariance of X
fts.plot.covariance(X)

# Compare values of the first PC scores with the first DPC scores
plot(res.pca$x[,1],t='l',xlab = "Time",ylab="Score", lwd = 2.5)
lines(res.dpca$scores[,1], col=2, lwd = 2.5)
legend(0,4,c("first PC score","first DPC score"), # puts text in the legend
       lty=c(1,1),lwd=c(2.5,2.5), col=1:2)
```

`fts.dpca.filters` *Functional dynamic PCA filters*

Description

From a given spectral density operator the dynamic principal component filter sequences are computed.

Usage

```
fts.dpca.filters(F, Ndpc = F$basisX$nbasis, q = 30)
```

Arguments

`F`	spectral density operator, provided as an object of class `fts.freqdom`.
`Ndpc`	an integer $\in \{1, \ldots, d\}$ with d =`F$basisX$nbasis`. 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. By default `q=30`.

Details

Dynamic principal components are linear filters $(\phi_{\ell k}(u)\colon k \in \mathbf{Z})$, $1 \leq \ell \leq d$. They are defined as the Fourier coefficients of the dynamic eigenvector $\varphi_\ell(\omega)(u)$ of a spectral density kernel $f_\omega(u,v)$, i.e. $\int_0^1 f_\omega(u,v)\varphi_\ell(\omega)(v)dv = \lambda_\ell(\omega)\varphi_\ell(\omega)(u)$ and

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

The index ℓ is referring to the ℓ-th largest dynamic eigenvalue $\lambda_\ell(\omega)$. For a given spectral density operator (provided as on object of class `fts.freqdom`) the function `fts.dpca.filters` computes $\phi_{\ell k}(u)$ for $|k| \leq$ q. Filters will be computed for $1 \leq \ell \leq$ `Ndpc`.

For more details we refer to Hormann et al. (2015).

Value

An object of class `fts.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 correspond to the coefficient vector of the filter functions.
- `lags` a vector with the lags of the filter coefficients.
- `basisX` `F$basis`, hence an object of class `basis.fd` (see `create.basis`).
- `correspondence` the correspondence matrix: all scalar products between basis functions.

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

The multivariate equivalent in the `freqdom` package: `dpca.filters`

Examples

```
data(pm10)
X = center.fd(pm10)

# Compute the spectral density operator with Bartlett weights
SD = fts.spectral.density(X, freq = (-50:50/50) * pi, q = 2, weight="Bartlett")
filters = fts.dpca.filters(SD, 2, q = 10)

# Plot filters 1 and 2
fts.plot.filters(filters, 2, one.plot = TRUE)
```

```
# Recompute with a different estimate of the spectral density (largerg q)
SD = fts.spectral.density(X, freq = (-50:50/50) * pi, q = 5, weight="Bartlett")
filters = fts.dpca.filters(SD, 2, q = 10)

# Plot filters 1 and 2
fts.plot.filters(filters, 2, one.plot = TRUE)
```

`fts.dpca.KLexpansion` *Dynamic KL expansion*

Description

Computes the dynamic KL expansion up to a given order.

Usage

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

Arguments

`X`	a functional time series given as an object of class `fd`.
`dpcs`	an object of class `fts.timedom`, representing the dpca filters obtained from the sample `X`. If `dpsc = NULL`, then `dpcs = fts.dpca.filter(fts.spectral.density(X))` is used.

Details

This function computes the L-order dynamic functional principal components expansion, defined by

$$\hat{X}_t^L(u) := \sum_{\ell=1}^{L} \sum_{k \in \mathbf{Z}} Y_{\ell,t+k} \phi_{\ell k}(u), \quad 1 \leq L \leq d,$$

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

For more details we refer to Hormann et al. (2015).

Value

An object of class `fd`.

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.

See Also

The multivariate equivalent in the `freqdom` package: `dpca.KLexpansion`

`fts.dpca.scores` *Functional dynamic principal component scores*

Description

Computes the dynamic principal component scores of a functional time series.

Usage

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

Arguments

`X`	a functional time series given as an object of class `fd`.
`dpcs`	an object of class `fts.timedom`, representing the dpca filters obtained from the sample `X`. If `dpsc = NULL`, then `dpcs = fts.dpca.filter(fts.spectral.density(X))` is used.

Details

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

$$Y_{\ell t} := \sum_{k \in \mathbf{Z}} \int_0^1 \phi_{\ell k}(v) X_{t-k}(v) dv, \quad 1 \leq \ell \leq d,$$

where $\phi_{\ell k}(v)$ and d are explained in `fts.dpca.filters`. (The integral is not necessarily restricted to the interval $[0, 1]$, this depends on the data.) For the sample version the sum extends over the range of lags for which the $\phi_{\ell k}$ are defined.

For more details we refer 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.

See Also

The multivariate equivalent in the `freqdom` package: `dpca.scores`

`fts.dpca.var` *Proportion of variance explained by dynamic principal components*

Description

Computes the proportion and cumulative proportion of variance explained by dynamic principal components.

Usage

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

Arguments

`F`	spectral density operator, provided as an object of class `fts.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 operator $\mathcal{F}_\omega$ 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_ℓ).

For more details we refer to Hormann et al. (2015).

Value

A 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.

See Also

The multivariate equivalent in the `freqdom` package: `dpca.var`

`fts.freqdom` *Creates an object of class* `fts.freqdom`.

Description

Creates an object of class `fts.freqdom`.

Usage

```
fts.freqdom(F, basisX, basisY = basisX)
```

Arguments

`F`	an object of class freqdom.
`basisX`	an object of class `basis.fd` (see `create.basis`)
`basisY`	an object of class `basis.fd` (see `create.basis`)

Details

This class is used to describe a frequency domain operator (for example a spectral density operator) on selected frequencies. Formally we consider an object of class `freqdom` and add some basis functions. Depending on the context, we have different interpretations for the new object.

(I) In order to define an operator which maps between two functions spaces, the we interpret `F$operators` as coefficients in the basis function expansion of the kernel of some finite rank operators

$$\mathcal{F}_k : \mathrm{span}(\texttt{basisY}) + \mathrm{i}\,\mathrm{span}(\texttt{basisY}) \to \mathrm{span}(\texttt{basisX}) + \mathrm{i}\,\mathrm{span}(\texttt{basisX}).$$

The kernels are $f_k(u,v) = \boldsymbol{b}_1'(u)\, F_k\, \boldsymbol{b}_2(v)$, where $\boldsymbol{b_1}(u) = (b_{11}(u), \ldots, b_{1d_1}(u))'$ and $\boldsymbol{b_2}(u) = (b_{21}(u), \ldots, b_{2d_1}(u))'$ are the basis functions provided by the arguments basisX and basisY, respectively. Moreover, we consider frequencies $\{\omega_1, \ldots, \omega_K\} \subset [-\pi, \pi]$. The object this function creates corresponds to the mapping $\omega_k \mapsto f_k(u,v)$.

(II) We may ignore basisX, and represent the linear mapping

$$\mathcal{F}_k : \mathrm{span}(\texttt{basisY}) + \mathrm{i}\,\mathrm{span}(\texttt{basisY}) \to C^{d_1},$$

by considering $f_k(v) := F_k\, \boldsymbol{b}_2(v)$ and $\mathcal{F}_k(x) = \int f_k(v)x(v)dv$.

(III) We may ignore basisY, and represent the linear mapping

$$\mathcal{F}_k : C^{d_1} \to \mathrm{span}(\texttt{basisX}) + \mathrm{i}\,\mathrm{span}(\texttt{basisX}),$$

by considering $f_k(u) := \boldsymbol{b}_1'(u) F_k$ and $\mathcal{F}_k(y) = f_k(u)y$.

Value

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

- `operators` returns the array `F$operators`.
- `basisX` returns `basisX` as given in the argument.
- `basisY` returns `basisY` as given in the argument.
- `freq` returns the vector `F$freq`.

See Also

The multivariate equivalent in the `freqdom` package: `freqdom`

`fts.plot.covariance` *Contour plot for the kernels of cross-covariance operators.*

Description

Contour plot for the kernels of cross-covariance operators.

Usage

```
fts.plot.covariance(X, Y = X, cor = FALSE, res = 200, lags = 0:3, nlevels = 25)
```

Arguments

`X`	an object of class `fd` representing a functional data sample.
`Y`	an object of class`fd` representing a functional data sample.
`cor`	if `FALSE` covariance kernels are plotted, if `TRUE` correlation kernel will be plotted.
`res`	number of discretization points to evaluate functional data.
`lags`	lags to plot, dafauts `0:3`
`nlevels`	number of color levels for the contour plot.

Examples

```
fts = fts.rar(100)

# Plot covariance operators of the time series curves
# We chose resolution equal 150 for better precision
fts.plot.covariance(fts, lags=0:2, res = 150)

# Plot correlation operators of the time series curves
fts.plot.covariance(fts, lags=0:2, cor = TRUE, res = 50)

# Make the grid of levels more dense
fts.plot.covariance(fts, lags=0:1, nlevels = 100)
```

fts.plot.filters *Plot kernels*

Description

Plot kernels

Usage

```
fts.plot.filters(A, Ndpc = 1, lags = -3:3, one.plot = FALSE, ...)
```

Arguments

A	a functional filter sequence given as object of class fts.timedom.
Ndpc	if Ndpc = k the first k filter sequences are plotted.
lags	number of lags to plot.
one.plot	if TRUE then functional filters corresponding belonging to the respective scores will all be plotted in the same graph.
...	arguments col, lwd, lty passed to plot

Examples

```
# Load example PM10 data from Graz, Austria
data(pm10) # loads functional time series pm10 to the environment
X = center.fd(pm10)

# Compute functional dynamic principal components with only one component
res.dpca = fts.dpca(X, Ndpc = 1, freq=(-25:25/25)*pi) # leave default freq for higher precision

# Plot Functional Dynamic Principal Component Filters
fts.plot.filters(res.dpca$filters)
```

fts.plot.operators *Contour plot of operator kernels.*

Description

Contour plot of operator kernels.

Usage

```
fts.plot.operators(A, res = 200, lags = 0, freq = 0, axis = "Re", nlevels = 25)
```

Arguments

A	an object of class fts.timedom or fts.freqdom.
res	number of discretization points to evaluate functional data.
lags	a vector of integers. For objects of class fts.timedom the lags of the operators we want to plot.
freq	a vector of frequencies in $[-\pi, \pi]$. For an object of class fts.freqdom the frequencies at which we want to plot the operator. If the chosen frequencies are not contained in A$freq, the closest frequencies will be used.
axis	if "Re" we plot the real part, if "Im" we plot the imaginary part of a complex-valued operator.
nlevels	number of color levels for the contour plot.

Examples

```
# Load example PM10 data from Graz, Austria
data(pm10) # loads functional time series pm10 to the environment
X = center.fd(pm10)

# Compute functional dynamic principal components with only one component
res.dpca = fts.dpca(X, Ndpc = 1, freq=(-25:25/25)*pi) # leave default freq for higher precision

# Plot the spectral density operator at frequencies -2, -3:3/30 * pi and 2
fts.plot.operators(res.dpca$spec.density,freq = c(-2,-3:3/30 * pi,2))
```

---

fts.rar *Simulate functional autoregressive processes*

Description

Generates functional autoregressive process.

Usage

```
fts.rar(
  n = 100,
  d = 11,
  Psi = NULL,
  op.norms = NULL,
  burnin = 20,
  noise = "mnorm",
  sigma = diag(d:1)/d,
  df = 4
)
```

Arguments

n	number of observations to generate.
d	dimension of the underlying multivariate VAR model.
Psi	an array of $p \geq 1$ coefficient matrices (need to be square matrices). `Psi[,,k]` is the k-th coefficient matrix. If Psi is provided then `d=dim(Psi)[1]`. If no value is set then we generate a functional autoregressive process of order 1. Then, `Psi[,,1]` is proportional to $\exp(-\lvert i-j\rvert : 1 \leq i,j \leq d)$ and such that Hilbert-Schmidt norm of the corresponding `lag-1` AR operator is 1/2.
op.norms	a vector with non-negative scalar entries to which the `Psi` are supposed to be scaled. The length of the vector must equal to the order of the model.
burnin	an integer ≥ 0. It specifies a number of initial observations to be trashed to achieve stationarity.
noise	`"mnorm"` for normal noise or `"t"` for student t noise. If not specified `"mvnorm"` is chosen.
sigma	covariance or scale matrix of the coefficients corresponding to functional innovations. The default value is `diag(d:1)/d`.
df	degrees of freqdom if `noise = "mt"`.

Details

The purpose is to simulate a functional autoregressive process of the form

$$X_t(u) = \sum_{k=1}^{p} \int_0^1 \Psi_k(u,v) X_{t-k}(v) dv + \varepsilon_t(u), \quad 1 \leq t \leq n.$$

Here we assume that the observations lie in a finite dimensional subspace of the function space spanned by Fourier basis functions $\boldsymbol{b}'(u) = (b_1(u), \ldots, b_d(u))$. That is $X_t(u) = \boldsymbol{b}'(u)\boldsymbol{X}_t$, $\varepsilon_t(u) = \boldsymbol{b}'(u)\boldsymbol{\varepsilon}_t$ and $\Psi_k(u,v) = \boldsymbol{b}'(u)\boldsymbol{\Psi}_k\boldsymbol{b}(v)$. Then it follows that

$$\boldsymbol{X}_t = \boldsymbol{\Psi}_1 \boldsymbol{X}_{t-1} + \cdots + \boldsymbol{\Psi}_p \boldsymbol{X}_{t-p} + \boldsymbol{\varepsilon}_t.$$

Hence the dynamic of the functional time series is described by a VAR(p) process.

In this mathematical model the law of $\boldsymbol{\varepsilon}_t$ is determined by `noise`. The matrices `Psi[,,k]` correspond to $\boldsymbol{\Psi}_k$. If `op.norms` is provided, then the coefficient matrices will be rescaled, such that the Hilbert-Schmidt norms of $\boldsymbol{\Psi}_k$ correspond to the vector.

Value

An object of class `fd`.

See Also

The multivariate equivalent in the `freqdom` package: `rar`

Examples

```
# Generate a FAR process without burnin (starting from 0)
fts = fts.rar(n = 5, d = 5, burnin = 0)
plot(fts)

# Generate a FAR process with burnin 50 (starting from observations
# already resambling the final distribution)
fts = fts.rar(n = 5, d = 5, burnin = 50)
plot(fts)

# Generate observations with very strong dependance
fts = fts.rar(n = 100, d = 5, burnin = 50, op.norms = 0.999)
plot(fts)
```

`fts.rma` *Simulate functional moving average processes*

Description

Generate a functional moving average process.

Usage

```
fts.rma(
  n = 100,
  d = 11,
  Psi = NULL,
  op.norms = NULL,
  noise = "mnorm",
  sigma = diag(d:1)/d,
  df = 4
)
```

Arguments

`n`	number of observations to generate.
`d`	dimension of the underlying multivariate VAR model.
`Psi`	an array of $p \geq 1$ coefficient matrices (need to be square matrices). `Psi[,,k]` is the k-th coefficient matrix. If `Psi` is provided then `d=dim(Psi)[1]`. If no value is set then we generate a functional autoregressive process of order 1. Then, `Psi[,,1]` is proportional to $\exp(-\|i-j\|: 1 \leq i,j \leq d)$ and such that Hilbert-Schmidt norm of the corresponding lag-1 MA operator is 1/2.
`op.norms`	a vector with non-negative scalar entries to which the `Psi` are supposed to be scaled. The length of the vector must equal to the order of the model.
`noise`	`"mnorm"` for normal noise or `"mt"` for student t noise. If not specified `"mvnorm"` is chosen.

`sigma`	covariance or scale matrix of the coefficients corresponding to functional innovations. The default value is `diag(d:1)/d`.
`df`	degrees of freqdom if `noise = "mt"`.

Details

The purpose is to simulate a functional autoregressive process of the form

$$X_t(u) = \sum_{k=1}^{p} \int_0^1 \Psi_k(u,v) X_{t-k}(v) dv + \varepsilon_t(u), \quad 1 \leq t \leq n.$$

Here we assume that the observations lie in a finite dimensional subspace of the function space spanned by Fourier basis functions $\boldsymbol{b}'(u) = (b_1(u), \ldots, b_d(u))$. That is $X_t(u) = \boldsymbol{b}'(u)\boldsymbol{X}_t$, $\varepsilon_t(u) = \boldsymbol{b}'(u)\boldsymbol{\varepsilon}_t$ and $\Psi_k(u,v) = \boldsymbol{b}'(u)\boldsymbol{\Psi}_k\boldsymbol{b}(v)$. Then it follows that

$$\boldsymbol{X}_t = \boldsymbol{\Psi}_1 \boldsymbol{X}_{t-1} + \cdots + \boldsymbol{\Psi}_p \boldsymbol{X}_{t-p} + \boldsymbol{\varepsilon}_t.$$

Hence the dynamic of the functional time series is described by a VAR(p) process.

In this mathematical model the law of $\boldsymbol{\varepsilon}_t$ is determined by `noise`. The matrices `Psi[,,k]` correspond to $\boldsymbol{\Psi}_k$. If `op.norms` is provided, then the coefficient matrices will be rescaled, such that the Hilbert-Schmidt norms of $\boldsymbol{\Psi}_k$ correspond to the vector.

Value

An object of class `fd`.

See Also

The multivariate equivalent in the `freqdom` package: `rma`

Examples

```
# Generate a FMA process without burnin (starting from 0)
fts = fts.rma(n = 5, d = 5)
plot(fts)

# Generate observations with very strong dependance
fts = fts.rma(n = 100, d = 5, op.norms = 0.999)
plot(fts)

# Generate observations with very strong dependance and noise
# from the multivariate t distribution
fts = fts.rma(n = 100, d = 5, op.norms = 0.999, noise = "mt")
plot(fts)
```

`fts.spectral.density` *Functional spectral and cross-spectral density operator*

Description

Estimates the spectral density operator and cross spectral density operator of functional time series.

Usage

```
fts.spectral.density(
  X,
  Y = X,
  freq = (-1000:1000/1000) * pi,
  q = ceiling((dim(X$coefs)[2])^{     0.33 }),
  weights = "Bartlett"
)
```

Arguments

`X`	an object of class `fd` containing T functional observations.
`Y`	an object of class `fd` containing T functional observations.
`freq`	a vector containing frequencies in $[-\pi, \pi]$ on which the spectral density should be evaluated. By default `freq=(-1000:1000/1000)*pi`.
`q`	window size for the kernel estimator, i.e. a positive integer. By default we choose `q = max(1, floor((dim(X$coefs)[2])^{0.33}))`.
`weights`	kernel used in the spectral smoothing. For possible choices see `spectral.density` in package `freqdom`. By default the Bartlett kernel is chosen.

Details

Let $X_1(u), \ldots, X_T(u)$ and $Y_1(u), \ldots, Y_T(u)$ be two samples of functional data. The cross-spectral density kernel between the two time series $(X_t(u))$ and $(Y_t(u))$ is defined as

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

The function `fts.spectral.density` determines the empirical cross-spectral density kernel between the two time series. The estimator is of the form

$$\widehat{f}_\omega^{XY}(u,v) = \sum_{|h| \le q} w(|k|/q) \widehat{c}_h^{XY}(u,v) e^{-ih\omega},$$

with $\widehat{c}_h^{XY}(u,v)$ defined in `fts.cov.structure`. The other paremeters are as in `cov.structure`.
Since $X_t(u) = \boldsymbol{b}_1'(u)\mathbf{x}_t$ and $Y_t(u) = \mathbf{y}_t'\boldsymbol{b}_2(u)$ we can write

$$\widehat{f}_\omega^{XY}(u,v) = \boldsymbol{b}_1'(u)\widehat{\mathcal{F}}^{\mathbf{xy}}(\omega)\boldsymbol{b}_2(v),$$

where $\widehat{\mathcal{F}}^{\mathbf{xy}}(\omega)$ is defined as for the function `spectral.density` for series of coefficient vectors $(\mathbf{x}_t : 1 \le t \le T)$ and $(\mathbf{y}_t : 1 \le t \le T)$.

Value

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

- `operators` an array. Element `[,,k]` in the coefficient matrix of the spectral density matrix evaluated at the k-th frequency listed in `freq`.
- `lags` returns the lags vector from the arguments.
- `basisX` returns `X$basis`, an object of class `basis.fd` (see `create.basis`).
- `basisY` returns `Y$basis`, an object of class `basis.fd` (see `create.basis`)

See Also

The multivariate equivalent in the `freqdom` package: `spectral.density`

Examples

```
data(pm10)
X = center.fd(pm10)

# Compute the spectral density operator with Bartlett weights
SD = fts.spectral.density(X, freq = (-50:50/50) * pi, q = 2, weight="Bartlett")
fts.plot.operators(SD, freq = -2:2)

# Compute the spectral density operator with Tukey weights
SD = fts.spectral.density(X, freq = (-50:50/50) * pi, q = 2, weight="Tukey")
fts.plot.operators(SD, freq = -2:2)
# Note relatively small difference between the two plots

# Now, compute the spectral density operator with Tukey weights and larger q
SD = fts.spectral.density(X, freq = (-50:50/50) * pi, q = 5, weight="Tukey")
fts.plot.operators(SD, freq = -2:2)
```

`fts.timedom` *Object of class* `fts.timedom`

Description

Creates an object of class `fts.timedom`. This object corresponds to a sequence of linear operators.

Usage

```
fts.timedom(A, basisX, basisY = basisX)
```

Arguments

`A`	an object of class timedom.
`basisX`	an object of class `basis.fd` (see `create.basis`)
`basisY`	an object of class `basis.fd` (see `create.basis`)

Details

This class is used to describe a functional linear filter, i.e. a sequence of linear operators, each of which corresponds to a certain lag. Formally we consider an object of class `timedom` and add some basis functions. Depending on the context, we have different interpretations for the new object.

(I) In order to define operators which maps between two functions spaces, the we interpret `A$operators` as coefficients in the basis function expansion of the kernel of some finite rank operators

$$\mathcal{A}_k : \text{span}(\texttt{basisY}) \to \text{span}(\texttt{basisX}).$$

The kernels are $a_k(u,v) = \boldsymbol{b}_1'(u)\, A_k\, \boldsymbol{b}_2(v)$, where $\boldsymbol{b_1}(u) = (b_{11}(u), \ldots, b_{1d_1}(u))'$ and $\boldsymbol{b_2}(u) = (b_{21}(u), \ldots, b_{2d_1}(u))'$ are the basis functions provided by the arguments basisX and basisY, respectively. Moreover, we consider lags $\ell_1 < \ell_2 < \cdots < \ell_K$. The object this function creates corresponds to the mapping $\ell_k \mapsto a_k(u,v)$.

(II) We may ignore basisX, and represent the linear mapping

$$\mathcal{A}_k : \text{span}(\texttt{basisY}) \to R^{d_1},$$

by considering $a_k(v) := A_k\, \boldsymbol{b}_2(v)$ and $\mathcal{A}_k(x) = \int a_k(v)x(v)dv$.

(III) We may ignore basisY, and represent the linear mapping

$$\mathcal{A}_k : R^{d_1} \to \text{span}(\texttt{basisX}),$$

by considering $a_k(u) := \boldsymbol{b}_1'(u)A_k$ and $\mathcal{A}_k(y) = a_k(u)y$.

Value

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

- `operators` returns the array `A$operators`.
- `basisX` returns basisX as given in the argument.
- `basisY` returns basisY as given in the argument.
- `lags` returns `A$lags`.

See Also

The multivariate equivalent in the `freqdom` package: `timedom`

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

Description

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

Usage

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

Arguments

X	some object

Value

TRUE if X is of type fts.freqdom, FALSE otherwise

See Also

fts.freqdom, fts.timedom, is.fts.timedom

---

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

Description

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

Usage

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

Arguments

X	some object

Value

TRUE if X is of type fts.timedom, FALSE otherwise

See Also

fts.freqdom, fts.timedom, is.fts.freqdom

pm10 *PM10 dataset*

Description

Concentration of partical matter of diameter 10 micrometers or smaller (PM10) is one of the most widely adopted measurements for assesment of ambient air quality. In this dataset, we provide 175 measurement of daily temporal distribution of PM10 in Graz, Austria, collected between December 2004 and June 2005.

For the purpose of this R package, raw data of 48 observations per day, was transformed to functional objects in Fourier basis using **fda** package.

Usage

```
pm10
```

Format

175 daily distribution functions in the fd format from the **fda** package

Source

Styrian Government, FA 17C Technical Environmental Protection and Safety, Air Quality Control Section,

References

Hormann, Siegfried, Brigitte Pfeiler, and Ernst Stadlober. *Analysis and prediction of particulate matter PM10 for the winter season in Graz.* Austrian Journal of Statistics 34.4 (2005): 307-326.

Examples

```
data(pm10)
summary(pm10)
plot(pm10)
```

Index