

Package ‘fdaACF’

August 11, 2020

Type Package

Title Autocorrelation Function for Functional Time Series

Version 0.2.0

Date 2020-08-09

Author Guillermo Mestre Marcos [aut, cre],
José Portela González [aut],
Gregory Rice [aut],
Antonio Muñoz San Roque [ctb],
Estrella Alonso Pérez [ctb]

Maintainer Guillermo Mestre Marcos <guillermo.mestre@comillas.edu>

Description Quantify the serial correlation across lags of a given functional time series using an autocorrelation function and a partial autocorrelation function for functional time series.

The autocorrelation functions are based on the L2 norm of the lagged covariance operators of the series. Functions are available for estimating the distribution of the autocorrelation functions under the assumption of strong functional white noise.

Imports CompQuadForm, pracma, fda, Matrix, vars

NeedsCompilation no

URL <https://github.com/GMestreM/fdaACF>

BugReports <https://github.com/GMestreM/fdaACF/issues>

License GPL (>= 2)

Encoding UTF-8

LazyData true

Depends R (>= 3.5.0)

RoxygenNote 7.0.1

Suggests testthat, fields

Repository CRAN

Date/Publication 2020-08-11 15:00:07 UTC

R topics documented:

elec_prices	2
estimate_iid_distr_Imhof	3
estimate_iid_distr_MC	5
fdaACF	6
fit_ARHp_FPCA	7
integral_operator	8
mat2fd	9
obtain_autocorrelation	10
obtain_autocovariance	12
obtain_autocov_eigenvalues	13
obtain_FACF	14
obtain_FPACF	16
obtain_suface_L2_norm	17
plot_autocovariance	18
plot_FACF	19
reconstruct_fd_from_PCA	20
simulate_iid_brownian_bridge	22
simulate_iid_brownian_motion	23
Index	24

elec_prices	<i>Daily electricity price profiles from the Day-Ahead Spanish Electricity Market</i>
-------------	---

Description

A dataset containing the hourly electricity prices for Spain in the Day-Ahead Market (MIBEL)

Usage

```
elec_prices
```

Format

A data frame with 365 rows and 24 variables:

- H1** Electricity price for hour 1
- H2** Electricity price for hour 2
- H3** Electricity price for hour 3
- H4** Electricity price for hour 4
- H5** Electricity price for hour 5
- H6** Electricity price for hour 6
- H7** Electricity price for hour 7
- H8** Electricity price for hour 8

- H9** Electricity price for hour 9
- H10** Electricity price for hour 10
- H11** Electricity price for hour 11
- H12** Electricity price for hour 12
- H13** Electricity price for hour 13
- H14** Electricity price for hour 14
- H15** Electricity price for hour 15
- H16** Electricity price for hour 16
- H17** Electricity price for hour 17
- H18** Electricity price for hour 18
- H19** Electricity price for hour 19
- H20** Electricity price for hour 20
- H21** Electricity price for hour 21
- H22** Electricity price for hour 22
- H23** Electricity price for hour 23
- H24** Electricity price for hour 24

Source

<https://www.esios.ree.es/es/analisis/600>

estimate_iid_distr_Imhof

Estimate distribution of the fACF under the iid. hypothesis using Imhof's method

Description

Estimate the distribution of the autocorrelation function under the hypothesis of strong functional white noise. This function uses Imhof's method to estimate the distribution.

Usage

```
estimate_iid_distr_Imhof(Y, v, autocovSurface, matindex, figure = FALSE,  
...)
```

Arguments

<code>Y</code>	Matrix containing the discretized values of the functional time series. The dimension of the matrix is $(n \times m)$, where n is the number of curves and m is the number of points observed in each curve.
<code>v</code>	Discretization points of the curves, by default <code>seq(from = 0, to = 1, length.out = 100)</code> .
<code>autocovSurface</code>	An $(m \times m)$ matrix with the discretized values of the autocovariance operator \hat{C}_0 , obtained by calling the function <code>obtain_autocovariance</code> . The value m indicates the number of points observed in each curve.
<code>matindex</code>	A vector containing the L2 norm of the autocovariance function. It can be obtained by calling function <code>obtain_surface_L2_norm</code> .
<code>figure</code>	Logical. If TRUE, plots the estimated distribution.
<code>...</code>	Further arguments passed to the plot function.

Value

Return a list with:

- `ex`: Knots where the distribution has been estimated
- `ef`: Discretized values of the estimated distribution.

Examples

```
# Example 1

N <- 100
v <- seq(from = 0, to = 1, length.out = 10)
sig <- 2
Y <- simulate_iid_brownian_bridge(N, v, sig)
nlags <- 1
autocovSurface <- obtain_autocovariance(Y, nlags)
matindex <- obtain_surface_L2_norm(v, autocovSurface)
# Remove lag 0
matindex <- matindex[-1]
Imhof_dist <- estimate_iid_distr_Imhof(Y, v, autocovSurface, matindex)
plot(Imhof_dist$ex, Imhof_dist$ef, type = "l", main = "ecdf obtained by Imhof's method")
grid()

# Example 2

N <- 400
v <- seq(from = 0, to = 1, length.out = 50)
sig <- 2
Y <- simulate_iid_brownian_bridge(N, v, sig)
autocovSurface <- obtain_autocovariance(Y, nlags)
matindex <- obtain_surface_L2_norm(v, autocovSurface)
# Remove lag 0
matindex <- matindex[-1]
```

```
Imhof_dist <- estimate_iid_distr_Imhof(Y,v,autocovSurface,matindex)
plot(Imhof_dist$ex,Imhof_dist$ef,type = "l",main = "ecdf obtained by Imhof's method")
grid()
```

estimate_iid_distr_MC *Estimate distribution of the fACF under the iid. hypothesis using MC method*

Description

Estimate the distribution of the autocorrelation function under the hypothesis of strong functional white noise. This function uses Montecarlo's method to estimate the distribution.

Usage

```
estimate_iid_distr_MC(Y, v, autocovSurface, matindex, nsimul = 10000,
  figure = FALSE, ...)
```

Arguments

Y	Matrix containing the discretized values of the functional time series. The dimension of the matrix is $(n \times m)$, where n is the number of curves and m is the number of points observed in each curve.
v	Discretization points of the curves, by default <code>seq(from = 0, to = 1, length.out = 100)</code> .
autocovSurface	An $(m \times m)$ matrix with the discretized values of the autocovariance operator \hat{C}_0 , obtained by calling the function <code>obtain_autocovariance</code> . The value m indicates the number of points observed in each curve.
matindex	A vector containing the L2 norm of the autocovariance function. It can be obtained by calling function <code>obtain_surface_L2_norm</code> .
nsimul	Positive integer indicating the number of MC simulations that will be used to estimate the distribution of the statistic. Increasing the number of simulations will improve the estimation, but it will increase the computational time. By default, <code>nsimul = 10000</code> .
figure	Logical. If TRUE, plots the estimated distribution.
...	Further arguments passed to the <code>plot</code> function.

Value

Return a list with:

- `ex`: Knots where the distribution has been estimated
- `ef`: Discretized values of the estimated distribution.
- `Reig`: Raw values of the i.i.d. statistic for each MC simulation.

Examples

```
# Example 1

N <- 100
v <- seq(from = 0, to = 1, length.out = 10)
sig <- 2
Y <- simulate_iid_brownian_bridge(N, v, sig)
nlags <- 1
autocovSurface <- obtain_autocovariance(Y,nlags)
matindex <- obtain_sufact_L2_norm (v,autocovSurface)
# Remove lag 0
matindex <- matindex[-1]
MC_dist <- estimate_iid_distr_MC(Y,v,autocovSurface,matindex)
plot(MC_dist$ex,MC_dist$ef,type = "l",main = "ecdf obtained by MC simulation")
grid()

# Example 2

N <- 400
v <- seq(from = 0, to = 1, length.out = 50)
sig <- 2
Y <- simulate_iid_brownian_bridge(N, v, sig)
nlags <- 20
autocovSurface <- obtain_autocovariance(Y,nlags)
matindex <- obtain_sufact_L2_norm (v,autocovSurface)
# Remove lag 0
matindex <- matindex[-1]
MC_dist <- estimate_iid_distr_MC(Y,v,autocovSurface,matindex)
plot(MC_dist$ex,MC_dist$ef,type = "l",main = "ecdf obtained by MC simulation")
grid()
```

Description

The fdaACF package provides diagnostic and analysis tools to quantify the serial autocorrelation across lags of a given functional time series in order to improve the identification and diagnosis of functional ARIMA models. The autocorrelation function is based on the L2 norm of the lagged covariance operators of the series. Several real-world datasets are included to illustrate the application of these techniques.

fit_ARHp_FPCA

*Fit an ARH(p) to a given functional time series***Description**

Fit an $ARH(p)$ model to a given functional time series. The fitted model is based on the model proposed in (Aue et al, 2015), first decomposing the original functional observations into a vector time series of n_{harm} FPCA scores, and then fitting a vector autoregressive model of order p ($VAR(p)$) to the time series of the scores. Once fitted, the Karhunen-Loève expansion is used to re-transform the fitted values into functional observations.

Usage

```
fit_ARHp_FPCA(y, v, p, n_harm, show_varprop = T)
```

Arguments

y	Matrix containing the discretized values of the functional time series. The dimension of the matrix is $(n \times m)$, where n is the number of curves and m is the number of points observed in each curve.
v	Numeric vector that contains the discretization points of the curves.
p	Numeric value specifying the order of the functional autoregressive model to be fitted.
n_harm	Numeric value specifying the number of functional principal components to be used when fitting the $ARH(p)$ model.
show_varprop	Logical. If <code>show_varprop = TRUE</code> , a plot of the proportion of variance explained by the first n_{harm} functional principal components will be shown. By default <code>show_varprop = TRUE</code> .

References

Aue, A., Norinho, D. D., Hormann, S. (2015). *On the Prediction of Stationary Functional Time Series* Journal of the American Statistical Association, 110, 378–392. <https://www.tandfonline.com/doi/full/10.1080/01621459.2014.909317>

Examples

```
# Example 1

# Simulate an ARH(1) process
N <- 250
dv <- 20
v <- seq(from = 0, to = 1, length.out = 20)

phi <- 1.3 * ((v) %**% t(v))

persp(v, v, phi,
```

```

    ticktype = "detailed",
    main = "Integral operator")

set.seed(3)
white_noise <- simulate_iid_brownian_bridge(N, v = v)

y <- matrix(nrow = N, ncol = dv)
y[1,] <- white_noise[1,]
for(jj in 2:N){
  y[jj,] <- white_noise[jj,];

  y[jj,] = y[jj,] + integral_operator(operator_kernel = phi,
                                     v = v,
                                     curve = y[jj-1,])
}

# Fit an ARH(1) model
mod <- fit_ARHp_FPCA(y = y,
                    v = v,
                    p = 1,
                    n_harm = 5)

# Plot results
plot(v, y[50,], type = "l", lty = 1, ylab = "")
lines(v, mod$y_est[50,], col = "red")
legend("bottomleft", legend = c("real", "est"),
      lty = 1, col = c(1,2))

```

integral_operator

Integral transformation of a curve using an integral operator

Description

Compute the integral transform of the curve Y_i with respect to a given integral operator Ψ . The transformation is given by

$$\Psi(Y_i)(v) = \int \psi(u, v) Y_i(u) du$$

Usage

```
integral_operator(operator_kernel, curve, v)
```

Arguments

operator_kernel

Matrix with the values of the kernel surface of the integral operator. The dimension of the matrix is $(g \times m)$, where g is the number of discretization points of the input curve and m is the number of discretization points of the output curve.

curve Vector containing the discretized values of a functional observation. The dimension of the matrix is $(1 \times m)$, where m is the number of points observed in the curve.

v Numerical vector specifying the discretization points of the curves.

Value

Returns a matrix the same size as `curve` with the transformed values.

Examples

```
# Example 1

v <- seq(from = 0, to = 1, length.out = 20)
set.seed(10)
curve <- sin(v) + rnorm(length(v))
operator_kernel <- 0.6*(v %*% t(v))
hat_curve <- integral_operator(operator_kernel, curve, v)
```

mat2fd

Obtain a fd object from a matrix

Description

This function returns a fd object obtained from the discretized functional observations contained in `mat_obj`.

It is assumed that the functional observations contained in `mat_obj` are real observations, hence a polygonal base will be used to obtain the functional object.

Usage

```
mat2fd(mat_obj, range_val = c(0, 1), argvals = NULL)
```

Arguments

mat_obj A matrix that contains the discretized functional observations.

range_val A numeric vector of length 2 that contains the range of the observed functional data. By default `range_val = c(0, 1)`.

argvals Optional argument that contains a strictly increasing vector of argument values at which line segments join to form a polygonal line. If `argvals = NULL`, it is assumed a equidistant discretization vector. By default `argvals = NULL`.

Value

A fd object obtained from the functional observations in `mat_obj`.

Examples

```
# Example 1
N <- 100
dv <- 30
v <- seq(from = 0, to = 1, length.out = dv)
set.seed(150) # For replication
mat_func_obs <- matrix(rnorm(N*dv),
                      nrow = N,
                      ncol = dv)
fd_func_obs <- mat2fd(mat_obj = mat_func_obs,
                    range_val = range(v),
                    argvals = v)

plot(fd_func_obs)
```

obtain_autocorrelation

Estimate the autocorrelation function of the series

Description

Obtain the empirical autocorrelation function for lags = 0, ..., nlags of the functional time series. Given Y_1, \dots, Y_T a functional time series, the sample autocovariance functions $\hat{C}_h(u, v)$ are given by:

$$\hat{C}_h(u, v) = \frac{1}{T} \sum_{i=1}^{T-h} (Y_i(u) - \bar{Y}_T(u))(Y_{i+h}(v) - \bar{Y}_T(v))$$

where $\bar{Y}_T(u) = \frac{1}{T} \sum_{i=1}^T Y_i(t)$ denotes the sample mean function. By normalizing these functions using the normalizing factor $\int \hat{C}_0(u, u) du$, the range of the autocovariance functions becomes $(0, 1)$; thus defining the autocorrelation functions of the series

Usage

```
obtain_autocorrelation(Y, v = seq(from = 0, to = 1, length.out =
  ncol(Y)), nlags)
```

Arguments

- | | |
|-------|---|
| Y | Matrix containing the discretized values of the functional time series. The dimension of the matrix is $(n \times m)$, where n is the number of curves and m is the number of points observed in each curve. |
| v | Discretization points of the curves, by default <code>seq(from = 0, to = 1, length.out = 100)</code> . |
| nlags | Number of lagged covariance operators of the functional time series that will be used to estimate the autocorrelation function. |

Value

Return a list with the lagged autocorrelation functions estimated from the data. Each function is given by a $(m \times m)$ matrix, where m is the number of points observed in each curve.

Examples

```
# Example 1

N <- 100
v <- seq(from = 0, to = 1, length.out = 10)
sig <- 2
bbridge <- simulate_iid_brownian_bridge(N, v, sig)
nlags <- 1
lagged_autocor <- obtain_autocorrelation(Y = bbridge,
                                       nlags = nlags)
image(x = v, y = v, z = lagged_autocor$Lag0)

# Example 2
require(fields)
N <- 500
v <- seq(from = 0, to = 1, length.out = 50)
sig <- 2
bbridge <- simulate_iid_brownian_bridge(N, v, sig)
nlags <- 4
lagged_autocov <- obtain_autocovariance(Y = bbridge,
                                       nlags = nlags)
lagged_autocor <- obtain_autocorrelation(Y = bbridge,
                                       v = v,
                                       nlags = nlags)

opar <- par(no.readonly = TRUE)
par(mfrow = c(1,2))
z_lims <- range(lagged_autocov$Lag1)
colors <- heat.colors(12)
image.plot(x = v,
          y = v,
          z = lagged_autocov$Lag1,
          legend.width = 2,
          zlim = z_lims,
          col = colors,
          xlab = "u",
          ylab = "v",
          main = "Autocovariance")
z_lims <- range(lagged_autocor$Lag1)
image.plot(x = v,
          y = v,
          z = lagged_autocor$Lag1,
          legend.width = 2,
          zlim = z_lims,
          col = colors,
          xlab = "u",
```

```

      ylab = "v",
      main = "Autocorrelation")
par(opar)

```

obtain_autocovariance *Estimate the autocovariance function of the series*

Description

Obtain the empirical autocovariance function for lags = 0, ..., nlags of the functional time series. Given Y_1, \dots, Y_T a functional time series, the sample autocovariance functions $\hat{C}_h(u, v)$ are given by:

$$\hat{C}_h(u, v) = \frac{1}{T} \sum_{i=1}^{T-h} (Y_i(u) - \bar{Y}_T(u))(Y_{i+h}(v) - \bar{Y}_T(v))$$

where $\bar{Y}_T(u) = \frac{1}{T} \sum_{i=1}^T Y_i(t)$ denotes the sample mean function.

Usage

```
obtain_autocovariance(Y, nlags)
```

Arguments

Y	Matrix containing the discretized values of the functional time series. The dimension of the matrix is $(n \times m)$, where n is the number of curves and m is the number of points observed in each curve.
nlags	Number of lagged covariance operators of the functional time series that will be used to estimate the autocorrelation function.

Value

Return a list with the lagged autocovariance functions estimated from the data. Each function is given by a $(m \times m)$ matrix, where m is the number of points observed in each curve.

Examples

```

# Example 1

N <- 100
v <- seq(from = 0, to = 1, length.out = 10)
sig <- 2
bbridge <- simulate_iid_brownian_bridge(N, v, sig)
nlags <- 1
lagged_autocov <- obtain_autocovariance(Y = bbridge,
                                       nlags = nlags)
image(x = v, y = v, z = lagged_autocov$Lag0)

```

```

# Example 2

N <- 500
v <- seq(from = 0, to = 1, length.out = 50)
sig <- 2
bbridge <- simulate_iid_brownian_bridge(N, v, sig)
nlags <- 10
lagged_autocov <- obtain_autocovariance(Y = bbridge,
                                       nlags = nlags)

image(x = v, y = v, z = lagged_autocov$Lag0)
image(x = v, y = v, z = lagged_autocov$Lag10)

# Example 3

require(fields)
N <- 500
v <- seq(from = 0, to = 1, length.out = 50)
sig <- 2
bbridge <- simulate_iid_brownian_bridge(N, v, sig)
nlags <- 4
lagged_autocov <- obtain_autocovariance(Y = bbridge,
                                       nlags = nlags)

z_lims <- range(lagged_autocov$Lag0)
colors <- heat.colors(12)
opar <- par(no.readonly = TRUE)
par(mfrow = c(1,5))
par(oma=c( 0,0,0,6))
for(k in 0:nlags){
  image(x=v,
        y=v,
        z = lagged_autocov[[paste0("Lag",k)]],
        main = paste("Lag",k),
        col = colors,
        xlab = "u",
        ylab = "v")
}
par(oma=c( 0,0,0,2.5)) # reset margin to be much smaller.
image.plot( legend.only=TRUE, legend.width = 2,zlim=z_lims, col = colors)
par(opar)

```

obtain_autocov_eigenvalues

Estimate eigenvalues of the autocovariance function

Description

Estimate the eigenvalues of the sample autocovariance function \hat{C}_0 . This function returns the eigenvalues which are greater than the value epsilon.

Usage

```
obtain_autocov_eigenvalues(v, Y, epsilon = 1e-04)
```

Arguments

v	Discretization points of the curves, by default <code>seq(from = 0, to = 1, length.out = 100)</code> .
Y	Matrix containing the discretized values of the functional time series. The dimension of the matrix is $(n \times m)$, where n is the number of curves and m is the number of points observed in each curve.
epsilon	Value used to determine how many eigenvalues will be returned. The eigenvalues $\lambda_j/\lambda_1 > \text{epsilon}$ will be returned. By default <code>epsilon = 0.0001</code> .

Value

A vector containing the k eigenvalues greater than epsilon.

Examples

```
N <- 100
v <- seq(from = 0, to = 1, length.out = 10)
sig <- 2
Y <- simulate_iid_brownian_bridge(N, v, sig)
lambda <- obtain_autocov_eigenvalues(v = v, Y = Y)
```

obtain_FACF

Obtain the autocorrelation function for a given functional time series.

Description

Estimate the lagged autocorrelation function for a given functional time series and its distribution under the hypothesis of strong functional white noise. This graphic tool can be used to identify seasonal patterns in the functional data as well as auto-regressive or moving average terms. i.i.d. bounds are included to test the presence of serial correlation in the data.

Usage

```
obtain_FACF(Y, v, nlags, ci = 0.95, estimation = "MC", figure = TRUE,
...)
```

Arguments

Y	Matrix containing the discretized values of the functional time series. The dimension of the matrix is $(n \times m)$, where n is the number of curves and m is the number of points observed in each curve.
v	Discretization points of the curves, by default <code>seq(from = 0, to = 1, length.out = 100)</code> .
nlags	Number of lagged covariance operators of the functional time series that will be used to estimate the autocorrelation function.
ci	A value between 0 and 1 that indicates the confidence interval for the i.i.d. bounds of the autocorrelation function. By default <code>ci = 0.95</code> .
estimation	Character specifying the method to be used when estimating the distribution under the hypothesis of functional white noise. Accepted values are: <ul style="list-style-type: none"> • "MC": Monte-Carlo estimation. • "Imhof": Estimation using Imhof's method. By default, <code>estimation = "MC"</code> .
figure	Logical. If TRUE, plots the estimated autocorrelation function with the specified i.i.d. bound.
...	Further arguments passed to the <code>plot_FACF</code> function.

Value

Return a list with:

- `BlueLine`: The upper prediction bound for the i.i.d. distribution.
- `rho`: Autocorrelation values for each lag of the functional time series.

References

Mestre, G., Portela, J., Muñoz-San Roque, A., Alonso, E. (2020). *Forecasting hourly supply curves in the Italian Day-Ahead electricity market with a double-seasonal SARMAHX model*. International Journal of Electrical Power & Energy Systems, 121, 1–13. <https://www.sciencedirect.com/science/article/pii/S0142061519337135>

Kokoszka, P., Rice, G., Shang, H.L. (2017). *Inference for the autocovariance of a functional time series under conditional heteroscedasticity* Journal of Multivariate Analysis, 162, 32–50. <https://www.sciencedirect.com/science/article/pii/S0142061519337135>

Examples

```
# Example 1

N <- 100
v <- seq(from = 0, to = 1, length.out = 5)
sig <- 2
Y <- simulate_iid_brownian_bridge(N, v, sig)
obtain_FACF(Y,v,20)
```

```
# Example 2

data(elec_prices)
v <- seq(from = 1, to = 24)
nlags <- 30
obtain_FACF(Y = as.matrix(elec_prices),
v = v,
nlags = nlags,
ci = 0.95,
figure = TRUE)
```

obtain_FPACF

Obtain the partial autocorrelation function for a given FTS.

Description

Estimate the partial autocorrelation function for a given functional time series and its distribution under the hypothesis of strong functional white noise.

Usage

```
obtain_FPACF(Y, v, nlags, n_harm, ci = 0.95, estimation = "MC",
figure = TRUE, ...)
```

Arguments

Y	Matrix containing the discretized values of the functional time series. The dimension of the matrix is $(n \times m)$, where n is the number of curves and m is the number of points observed in each curve.
v	Discretization points of the curves.
nlags	Number of lagged covariance operators of the functional time series that will be used to estimate the partial autocorrelation function.
n_harm	Number of principal components that will be used to fit the ARH(p) models.
ci	A value between 0 and 1 that indicates the confidence interval for the i.i.d. bounds of the partial autocorrelation function. By default $ci = 0.95$.
estimation	Character specifying the method to be used when estimating the distribution under the hypothesis of functional white noise. Accepted values are: <ul style="list-style-type: none"> "MC": Monte-Carlo estimation. "Imhof": Estimation using Imhof's method. By default, $estimation = "MC"$.
figure	Logical. If TRUE, plots the estimated partial autocorrelation function with the specified i.i.d. bound.
...	Further arguments passed to the <code>plot_FACF</code> function.

Value

Return a list with:

- BlueLine: The upper prediction bound for the i.i.d. distribution.
- rho: Partial autocorrelation coefficients for each lag of the functional time series.

Examples

```
# Example 1

N <- 100
v <- seq(from = 0, to = 1, length.out = 5)
sig <- 2
Y <- simulate_iid_brownian_bridge(N, v, sig)
obtain_FPACF(Y,v,20, n_harm = 2)

# Example 2

data(elec_prices)
v <- seq(from = 1, to = 24)
nlags <- 30
obtain_FPACF(Y = as.matrix(elec_prices),
v = v,
nlags = nlags,
n_harm = 5,
ci = 0.95,
figure = TRUE)
```

obtain_suface_L2_norm *Obtain L2 norm of the autocovariance functions*

Description

Returns the L2 norm of the lagged autocovariance functions \hat{C}_h . The L2 norm of these functions is defined as

$$\sqrt{\int \int \hat{C}_h^2(u, v) dudv}$$

Usage

```
obtain_suface_L2_norm(v, autocovSurface)
```

Arguments

v Discretization points of the curves, by default `seq(from = 0, to = 1, length.out = 100)`.

autocovSurface An $(m \times m)$ matrix with the discretized values of the autocovariance operator \hat{C}_0 , obtained by calling the function `obtain_autocovariance`. The value m indicates the number of points observed in each curve.

Value

A vector containing the L2 norm of the lagged autocovariance functions autocovSurface.

Examples

```
# Example 1

N <- 100
v <- seq(from = 0, to = 1, length.out = 10)
sig <- 2
Y <- simulate_iid_brownian_bridge(N, v, sig)
nlags <- 1
autocovSurface <- obtain_autocovariance(Y=Y,nlags = nlags)
norms <- obtain_surface_L2_norm(v = v,autocovSurface = autocovSurface)
plot_autocovariance(fun.autocovariance = autocovSurface,lag = 1)
title(sub = paste0("Lag ",1," - L2 Norm: ",norms[2]))

# Example 2

N <- 400
v <- seq(from = 0, to = 1, length.out = 50)
sig <- 2
Y <- simulate_iid_brownian_bridge(N, v, sig)
nlags <- 2
autocovSurface <- obtain_autocovariance(Y=Y,nlags = nlags)
norms <- obtain_surface_L2_norm(v = v,autocovSurface = autocovSurface)
opar <- par(no.readonly = TRUE)
par(mfrow = c(1,3))
plot_autocovariance(fun.autocovariance = autocovSurface,lag = 0)
title(sub = paste0("Lag ",0," - L2 Norm: ",norms[1]))
plot_autocovariance(fun.autocovariance = autocovSurface,lag = 1)
title(sub = paste0("Lag ",1," - L2 Norm: ",norms[2]))
plot_autocovariance(fun.autocovariance = autocovSurface,lag = 2)
title(sub = paste0("Lag ",2," - L2 Norm: ",norms[3]))
par(opar)
```

plot_autocovariance *Generate a 3D plot of the autocovariance surface of a given FTS*

Description

Obtain a 3D plot of the autocovariance surfaces of a given functional time series. This visualization is useful to detect any kind of dependency between the discretization points of the series.

Usage

```
plot_autocovariance(fun.autocovariance, lag = 0, ...)
```

Arguments

- `fun.autocovariance` A list obtained by calling the function `obtain_autocovariance`.
- `lag` An integer between 0 and `nlags`, indicating the lagged autocovariance function to be plotted. By default 0.
- `...` Further arguments passed to the `persp` function.

Examples

```
# Example 1

N <- 100
v <- seq(from = 0, to = 1, length.out = 10)
sig <- 2
bbridge <- simulate_iid_brownian_bridge(N, v, sig)
nlags <- 1
lagged_autocov <- obtain_autocovariance(Y = bbridge, nlags = nlags)
plot_autocovariance(lagged_autocov, 1)

# Example 2

N <- 500
v <- seq(from = 0, to = 1, length.out = 50)
sig <- 2
bbridge <- simulate_iid_brownian_bridge(N, v, sig)
nlags <- 4
lagged_autocov <- obtain_autocovariance(Y = bbridge, nlags = nlags)
opar <- par(no.readonly = TRUE)
par(mfrow = c(1,5))
for(k in 0:nlags){
  plot_autocovariance(lagged_autocov, k)
}
par(opar)
```

plot_FACF

Plot the autocorrelation function of a given FTS

Description

Plot a visual representation of the autocorrelation function of a given functional time series, including the upper i.i.d. bound.

Usage

```
plot_FACF(rho, BlueLine, ci, ...)
```

Arguments

rho	Autocorrelation values for each lag of the functional time series obtained by calling the function obtain_FACF.
BlueLine	The upper prediction bound for the i.i.d. distribution obtained by calling the function obtain_FACF.
ci	Value between 0 and 1 that was used when calling the function obtain_FACF. This value is only used to display information in the figure.
...	Further arguments passed to the plot function.

Examples

```
# Example 1

N <- 100
v <- seq(from = 0, to = 1, length.out = 10)
sig <- 2
bbridge <- simulate_iid_brownian_bridge(N, v, sig)
nlags <- 15
upper_bound <- 0.95
fACF <- obtain_FACF(Y = bbridge, v = v, nlags = nlags, ci = upper_bound, figure = FALSE)
plot_FACF(rho = fACF$rho, BlueLine = fACF$BlueLine, ci = upper_bound)

# Example 2

N <- 200
v <- seq(from = 0, to = 1, length.out = 30)
sig <- 2
bbridge <- simulate_iid_brownian_bridge(N, v, sig)
nlags <- 15
upper_bound <- 0.95
fACF <- obtain_FACF(Y = bbridge, v = v, nlags = nlags, ci = upper_bound, figure = FALSE)
plot_FACF(rho = fACF$rho, BlueLine = fACF$BlueLine, ci = upper_bound)
```

```
reconstruct_fd_from_PCA
```

Obtain the reconstructed curves after PCA

Description

This function reconstructs the functional curves from a score vector using the basis obtained after applying functional PCA. This allows the user to draw estimations from the joint density of the FPCA scores and reconstruct the curves for those new scores.

Usage

```
reconstruct_fd_from_PCA(pca_struct, scores, centerfns = T)
```

Arguments

pca_struct	List obtained after calling function <code>pca.fd</code> .
scores	Numerical vector that contains the scores of the fPCA decomposition for one functional observation.
centerfns	Logical value specifying wheter the FPCA performed used <code>centerfns = T</code> or <code>centerfns = F</code> . By default <code>centerfns = T</code> .

Value

Returns a object of type `fd` that contains the reconstructed curve.

Examples

```
# Example 1

# Simulate fd
nobs <- 200
dv <- 10
basis<-fda::create.bspline.basis(rangeval=c(0,1),nbasis=10)
set.seed(5)
C <- matrix(rnorm(nobs*dv), ncol = dv, nrow = nobs)
fd_sim <- fda::fd(coef=t(C),basis)

# Perform FPCA
pca_struct <- fda::pca.fd(fd_sim,nharm = 6)

# Reconstruct first curve
fd_rec <- reconstruct_fd_from_PCA(pca_struct = pca_struct, scores = pca_struct$scores[1,])
plot(fd_sim[1])
plot(fd_rec, add = TRUE, col = "red")
legend("topright",
      legend = c("Real Curve", "PCA Reconstructed"),
      col = c("black","red"),
      lty = 1)

# Example 2 (Perfect reconstruction)

# Simulate fd
nobs <- 200
dv <- 7
basis<-fda::create.bspline.basis(rangeval=c(0,1),nbasis=dv)
set.seed(5)
C <- matrix(rnorm(nobs*dv), ncol = dv, nrow = nobs)
fd_sim <- fda::fd(coef=t(C),basis)

# Perform FPCA
pca_struct <- fda::pca.fd(fd_sim,nharm = dv)

# Reconstruct first curve
fd_rec <- reconstruct_fd_from_PCA(pca_struct = pca_struct, scores = pca_struct$scores[1,])
plot(fd_sim[1])
```

```
plot(fd_rec, add = TRUE, col = "red")
legend("topright",
      legend = c("Real Curve", "PCA Reconstructed"),
      col = c("black", "red"),
      lty = 1)
```

```
simulate_iid_brownian_bridge
```

Simulate a FTS from a brownian bridge process

Description

Generate a functional time series from a Brownian Bridge process. If $W(t)$ is a Wiener process, the Brownian Bridge is defined as $W(t) - tW(1)$. Each functional observation is discretized in the points indicated in v . The series obtained is i.i.d. and does not exhibit any kind of serial correlation.

Usage

```
simulate_iid_brownian_bridge(N, v = seq(from = 0, to = 1, length.out =
  100), sig = 1)
```

Arguments

<code>N</code>	The number of observations of the simulated data.
<code>v</code>	Discretization points of the curves, by default <code>seq(from = 0, to = 1, length.out = 100)</code> .
<code>sig</code>	Standard deviation of the Brownian Motion process, by default 1.

Value

Return the simulated functional time series as a matrix.

Examples

```
N <- 100
v <- seq(from = 0, to = 1, length.out = 20)
sig <- 2
bbridge <- simulate_iid_brownian_bridge(N, v, sig)
matplot(v,t(bbridge), type = "l", xlab = "v", ylab = "Value")
```

`simulate_iid_brownian_motion`*Simulate a FTS from a brownian motion process*

Description

Generate a functional time series from a Brownian Motion process. Each functional observation is discretized in the points indicated in v . The series obtained is i.i.d. and does not exhibit any kind of serial correlation

Usage

```
simulate_iid_brownian_motion(N, v = seq(from = 0, to = 1, length.out =
  100), sig = 1)
```

Arguments

<code>N</code>	The number of observations of the simulated data.
<code>v</code>	Discretization points of the curves, by default <code>seq(from = 0, to = 1, length.out = 100)</code> .
<code>sig</code>	Standard deviation of the Brownian Motion process, by default 1.

Value

Return the simulated functional time series as a matrix.

Examples

```
N <- 100
v <- seq(from = 0, to = 1, length.out = 20)
sig <- 2
bmotion <- simulate_iid_brownian_motion(N, v, sig)
matplot(v,t(bmotion), type = "l", xlab = "v", ylab = "Value")
```

Index

* data

- elec_prices, 2

- elec_prices, 2
- estimate_iid_distr_Imhof, 3
- estimate_iid_distr_MC, 5

- fdaACF, 6
- fdaACF-package (fdaACF), 6
- fit_ARHp_FPCA, 7

- integral_operator, 8

- mat2fd, 9

- obtain_autocorrelation, 10
- obtain_autocov_eigenvalues, 13
- obtain_autocovariance, 12
- obtain_FACF, 14
- obtain_FPACF, 16
- obtain_sufact_L2_norm, 17

- plot_autocovariance, 18
- plot_FACF, 19

- reconstruct_fd_from_PCA, 20

- simulate_iid_brownian_bridge, 22
- simulate_iid_brownian_motion, 23