Package ‘gmfd’

April 6, 2018

**Type**  Package

**Title**  Inference and Clustering of Functional Data

**Version**  1.0.1

**Author**  Andrea Martino [aut, cre],
Andrea Ghiglietti [aut],
Francesca Ieva [aut],
Anna Maria Paganoni [aut]

**Maintainer**  Andrea Martino <andrea.martino@polimi.it>


**Depends**  R (>= 3.3.0)

**License**  GPL-3

**LazyData**  true

**Encoding**  UTF-8

**RoxygenNote**  6.0.1.9000

**Imports**  graphics, stats

**Suggests**  knitr, rmarkdown

**VignetteBuilder**  knitr

**NeedsCompilation**  no

**Repository**  CRAN

**Date/Publication**  2018-04-06 09:34:16 UTC

---

**R topics documented:**

- funData ................................................................. 2
- funDist ............................................................... 3
- gmfd_diss ............................................................ 4
funData

S3 Class for functional datasets. A class for univariate or multivariate functional dataset.

Description

S3 Class for functional datasets. A class for univariate or multivariate functional dataset.

Usage

funData(grid, data)

Arguments

grid
  the grid over which the functional dataset is defined.

data
  a vector, a matrix or a list of vectors or matrices containing the functional data.

Value

The function returns a S3 object of class funData, containing the grid over which the functional dataset is defined and a matrix or a list of vectors or matrices containing the functional data.

See Also
gmfd_simulate

Examples

# Define parameters
n <- 50
P <- 100
K <- 150

# Grid of the functional dataset
t <- seq(0, 1, length.out = P)

# Define the means and the parameters to use in the simulation
m1 <- t^2 * (1 - t)
m2 <- t * (1 - t)^2
rho <- rep(0, K)
theta <- matrix(0, K, P)
for (k in 1:K) {

funDist

\[
\text{rho}[k] \leftarrow 1 / (k + 1)^2 \\
\text{if ( k%%2 == 0 )} \\
\quad \text{theta}[k,] \leftarrow \sqrt{2} \ast \sin( k \ast \pi \ast t ) \\
\text{else if ( k%%2 != 0 \&\& k != 1 )} \\
\quad \text{theta}[k,] \leftarrow \sqrt{2} \ast \cos( (k - 1) \ast \pi \ast t ) \\
\text{else} \\
\quad \text{theta}[k,] \leftarrow \text{rep}(1, P)
\]

# Simulate the functional data
x1 <- gmfd_simulate(n, m1, rho = rho, theta = theta)
x2 <- gmfd_simulate(n, m2, rho = rho, theta = theta)
FD <- funData(t, list(x1, x2))

---

funDist

**Distance function**

**Description**

This function allows you to compute the distance between two curves with the chosen metric.

**Usage**

funDist(FD1, FD2, metric, p = NULL, lambda = NULL, phi = NULL, k_trunc = NULL)

**Arguments**

- **FD1**
  - a functional data object of type funData for the first curve
- **FD2**
  - a functional data object of type funData for the second curve
- **metric**
  - the chosen distance to be used: "l2" for the classical L2-distance, "trunc" for the truncated Mahalanobis semi-distance, "mahalanobis" for the generalized Mahalanobis distance.
- **p**
  - a positive numeric value containing the parameter of the regularizing function for the generalized Mahalanobis distance.
- **lambda**
  - a vector containing the eigenvalues in descending order of the functional data from which the curves are extracted.
- **phi**
  - a matrix containing the eigenfunctions of the functional data in its columns from which the curves are extracted.
- **k_trunc**
  - a positive numeric value representing the number of components at which the truncated mahalanobis distance must be truncated

**Value**

The function returns a numeric value indicating the distance between the two curves.
References


Examples

```r
# Define parameters:
n <- 50
P <- 100
K <- 150

# Grid of the functional dataset
t <- seq( 0, 1, length.out = P )

# Define the means and the parameters to use in the simulation
m1 <- t^2 * ( 1 - t )
rho <- rep( K, K )
theta <- matrix( K, P )
for ( k in 1:K ) {
  rho[k] <- 1 / ( k + 1 )^2
  if ( k%%2 == 0 )
    theta[k, ] <- sqrt( 2 ) * sin( k * pi * t )
  else if ( k%%2 != 0 && k != 1 )
    theta[k, ] <- sqrt( 2 ) * cos( ( k - 1 ) * pi * t )
  else
    theta[k, ] <- rep( 1, P )
}

# Simulate the functional data
z <- gmfd_simulate( n, m1, rho, theta = theta )

# Extract two rows of the functional data
x <- funData( t, z[, 1] )
y <- funData( t, z[, 2] )

lambda <- eigen(cov(z))$values
phi <- eigen(cov(z))$vectors

d <- fundist( x, y, metric = "mahalanobis", p = 1, lambda = lambda, phi = phi )
```

---

**Dissimilarity matrix function**

This function computes the dissimilarity matrix containing the distances between the curves of the functional dataset.
Usage

gmfd_diss(FD, metric, p = NULL, k_trunc = NULL)

Arguments

FD
a functional data object of type funData

metric
the chosen distance to be used. Choose "L2" for the classical L2-distance, "trunc" for the truncated Mahalanobis semi-distance, "mahalanobis" for the generalized Mahalanobis distance.

p
a positive numeric value containing the parameter of the regularizing function for the generalized Mahalanobis distance.

k_trunc
a positive numeric value representing the number of components at which the truncated mahalanobis distance must be truncated

Value

The function returns a matrix of numeric values containing the distances between the curves.

References


Examples

```r
# Define parameters
n <- 50
P <- 100
K <- 150

t <- seq(0, 1, length.out = P)

# Define the means and the parameters to use in the simulation
m1 <- t^2 * (1 - t)

rho <- rep(0, K)
theta <- matrix(0, K, P)
for (k in 1:K) {
  rho[k] <- 1 / (k + 1)^2
  if (k%%2 == 0)
    theta[k,] <- sqrt(2) * sin(2 * pi * t)
  else if (k%%2 != 0 & k != 1)
    theta[k,] <- sqrt(2) * cos((k - 1) * pi * t)
  else
    theta[k,] <- rep(1, P)
}
```
gmfd_kmeans

# Simulate the functional data
x <- gmfd_simulate( n, m, rho = rho, theta = theta )

FD <- funData( t, x )

D <- gmfd_diss( FD, metric = "l2" )

Description

This function performs a k-means clustering algorithm on an univariate or multivariate functional data using a generalization of Mahalanobis distance.

Usage

gmfd_kmeans(FD, n.cl = 2, metric, p = NULL, k_trunc = NULL)

Arguments

FD          a functional data object of type funData.
n.cl        an integer representing the number of clusters.
metric      the chosen distance to be used: "L2" for the classical L2-distance, "trunc" for the truncated Mahalanobis semi-distance, "mahalanobis" for the generalized Mahalanobis distance.
p           a positive numeric value containing the parameter of the regularizing function for the generalized Mahalanobis distance.
k_trunc     a positive numeric value representing the number of components at which the truncated mahalanobis distance must be truncated

Value

The function returns a list with the following components: cluster: a vector of integers (from 1 to n.cl) indicating the cluster to which each curve is allocated; centers: a list of d matrices (k x T) containing the centroids of the clusters

References


gmfd_simulate

Simulation of a functional sample

Description

Simulate a univariate functional sample using a Karhunen Loeve expansion.
Usage

gmf_simulate(size, mean, covariance = NULL, rho = NULL, theta = NULL)

Arguments

- `size`: a positive integer indicating the size of the functional sample to simulate.
- `mean`: a vector representing the mean of the sample.
- `covariance`: a matrix from which the eigenvalues and eigenfunctions must be extracted.
- `rho`: a vector of the eigenvalues in descending order to be used for the simulation.
- `theta`: a matrix containing the eigenfunctions in its columns to be used for the simulation.

Value

The function returns a functional data object of type `funData`.

Examples

```r
# Define parameters
n <- 50
P <- 100
K <- 150

t <- seq(0, 1, length.out = P)

# Define the means and the parameters to use in the simulation
# with the Karhunen-Loève expansion
m1 <- t^2 * (1 - t)
rho <- rep(0, K)
theta <- matrix(0, K, P)
for (k in 1:K) {
  rho[k] <- 1 / (k + 1)^2
  if (k %% 2 == 0)
    theta[k,] <- sqrt(2) * sin(k * pi * t)
  else if (k %% 2 != 0 && k != 1)
    theta[k,] <- sqrt(2) * cos((k - 1) * pi * t)
  else
    theta[k,] <- rep(1, P)
}

# Simulate the functional data
x <- gmf_simulate(n, m1, rho = rho, theta = theta)
```
gmfd_test

Two-sample hypotesis tests

Description

Performs a two sample hypotesis tests on two samples of functional data.

Usage

gmfd_test(FD1, FD2, conf.level = 0.95, stat_test, p = NULL, k_trunc = NULL)

Arguments

FD1 a functional data object of type fundata of the first sample.
FD2 a functional data object of type fundata of the second sample.
conf.level confidence level of the test.
stat_test the chosen test statistic to be used: "l2" for the classical L2-distance, "l2_trunc" for the truncated L2-distance, "trunc" for the truncated Mahalanobis semi-distance, "mahalanobis" for the generalized Mahalanobis distance
p a vector of positive numeric value containing the parameters of the regularizing function for the generalized Mahalanobis distance.
k_trunc a positive numeric value representing the number of components at which the truncated mahalanobis distance must be truncated

Value

The function returns a list with the following components:

statistic the value of the test statistic.
quantile the value of the quantile.
p.value the p-value for the test.

References


See Also

funDist
Examples

# Define parameters
n <- 50
P <- 100
K <- 150

# Grid of the functional dataset
t <- seq(0, 1, length.out = P)

# Define the means and the parameters to use in the simulation
m1 <- t^2 * (1 - t)
rho <- rep(0, K)
theta <- matrix(0, K, P)
for (k in 1:K) {
    rho[k] <- 1/(k+1)^2
    if (k%%2 == 0)
        theta[k,] <- sqrt(2) * sin(k * pi * t)
    else if (k%%2 != 0 && k != 1)
        theta[k,] <- sqrt(2) * cos((k-1) * pi * t)
    else
        theta[k,] <- rep(1, P)
}
s <- 0
for (k in 1:K)
    s <- s + sqrt(rho[k]) * theta[k,]

m2 <- m1 + 0.1 * s

# Simulate the functional data
x1 <- gmfd_simulate(n, m1, rho = rho, theta = theta)
x2 <- gmfd_simulate(n, m2, rho = rho, theta = theta)
FD1 <- fundata(t, x1)
FD2 <- fundata(t, x2)
output <- gmfd_test(FD1, FD2, 0.95, "mahalanobis", p = 10^5)

---

plot.funData

A method to plot funData objects

Description

This function performs the plot of a functional dataset stored in an object of class funData.

Usage

## S3 method for class 'funData'
plot(x, ...)

Arguments

x  the univariate functional dataset in form of funData object.

...  additional graphical parameters to be used in plotting functions

See Also

funData

Examples

# Define parameters
n <- 50
P <- 100
K <- 150

t <- seq(0, 1, length.out = P)

# Define the means and the parameters to use in the simulation
m1 <- t^2 * (1 - t)
m2 <- t * (1 - t)^2

rho <- rep(0, K)
theta <- matrix(0, K, P)
for (k in 1:K) {
  rho[k] <- 1 / (k + 1)^2
  if (k%%2 == 0)
    theta[k,] <- sqrt(2) * sin(k * pi * t)
  else if (k%%2 != 0 && k != 1)
    theta[k,] <- sqrt(2) * cos((k - 1) * pi * t)
  else
    theta[k,] <- rep(1, P)
}

# Simulate the functional data
x1 <- gmfd_simulate(n, m1, rho = rho, theta = theta)
x2 <- gmfd_simulate(n, m2, rho = rho, theta = theta)

FD <- funData(t, list(x1, x2))

plot(FD)
Index

*Topic **Clustering**
gmfd_kmeans, 6

*Topic **Inference**
gmfd_test, 9

*Topic **Simulation**
gmfd_simulate, 7

*Topic **distance**
funDist, 3

funData, 2, 11
funDist, 3, 7, 9

gmfd_diss, 4
gmfd_kmeans, 6
gmfd_simulate, 2, 7
gmfd_test, 9

plot.funData, 10