Frédéric Ferraty and Philippe Vieu

R/S-plus Functions Help Files

NonParametric Functional Data Analysis

June 2, 2006

Contents

Introduction

This part proposes indexes and help files for the R/S-plus routines needed for implementing NonParametric Functional Data Analysis (NPFDA). As you will see, two kinds of indexes are available (the first corresponds to the standard alphabetical order whereas the second is listed by category). Afterthen, the reader will find help files for the main routines.

1

Indexes of R/S-plus routines

1.1 Alphabetical Index for all routines

`funopare.kernel`	Performs functional prediction (regression) of a scalar response from a sample of curves via the functional kernel estimator. A global bandwidth is considered without automatic selection
`funopare.kernel.cv`	Performs functional prediction (regression) of a scalar response from a sample of curves via the functional kernel estimator. A global bandwidth is automatically selected with a cross-validation procedure
`funopare.knn`	Performs functional prediction (regression) of a scalar response from a sample of curves via the functional kernel estimator. A bandwidth corresponding to number of neighbours has to be given
`funopare.knn.gcv`	Performs functional prediction (regression) of a scalar response from a sample of curves via the functional kernel estimator. A global bandwidth (i.e. a number of neighbours) is selected by a cross-validation procedure
`funopare.knn.lcv`	Performs functional prediction (regression) of a scalar response from a sample of curves via the functional kernel estimator. A local bandwidth (i.e. number of neighbours depending on the curve where the estimator is evaluated) is selected by a cross-validation procedure
`funopare.mode.lcv`	Performs functional prediction of a scalar response from a sample of curves by computing the functional conditional mode. A local bandwidth (i.e. local number of neighbours) is selected by a "trivial" cross-validation procedure
`funopare.quantile.lcv`	Performs functional prediction of a scalar response from a sample of curves by computing the functional conditional mode. A local bandwidth (i.e. local number of neighbours) is selected by a "trivial" cross-validation procedure

`hshift`	Returns the "horizontal shifted proximity" between two curves
`indicator`	Uniform kernel function
`integrated.quadratic`	Integrated quadratic kernel function
`integrated.triangle`	Integrated triangle kernel function
`median.npfda`	Performs the median curve from the semimetric matrix of a sample of curves (classification)
`mplsr`	Computes PLS regression coefficients
`prob.curve`	Returns a probabilty curve (classification)
`quadratic`	Quadratic kernel function
`rank.minima`	Returns the rank of the local minima of a numeric strictly positive discretized function
`semimetric.deriv`	Computes between curves a semimetric based on their derivative (via a B-spline expansion)
`semimetric.fourier`	Computes between curves a semimetric based on their fourier expansion
`semimetric.hshift`	Computes between curves a semimetric taking into account an horizontal shift effect
`semimetric.mplsr`	Computes between curves a semimetric based on the partial least squares method
`semimetric.pca`	Computes between curves a semimetric based on the functional principal component analysis
`symsolve`	Solves linear systems A x = b for x where A is symmetric
`triangle`	Triangle kernel function
`unbal2equibal`	Transforms unbalanced functional data into Balanced functional data

1.2 Index by Category

1.2.1 classifying a sample of curves (clustering)

`classif.bw`	Computes a bandwidth among a fixed sequence of bandwidths
`classif.hi`	Returns an Heterogeneity Index (HI) for a set of curves
`classif.npfda`	Performs the nonparametric unsupervised classification (or clustering) method of a sample of curves
`classif.part`	From a set of curves, it performs a partition and computes the corresponding splitting score
`classif.shi`	Returns a Subsampling Heterogeneity Index (SHI) from a set of curves
`classif.split`	Returns a partition of a set of curves
`entropy`	Returns the entropy of any density function
`funopa.mode`	Returns the rank of the modal curve in a set of curves
`median.npfda`	Performs the median curve from the semimetric matrix of a sample of curves
`prob.curve`	Returns a probabilty curve

1.2.2 discriminating a sample of curves (supervised classification)

`funopadi.knn.lcv`	Performs functional discrimination of a sample of curves

1.2.3 Kernel functions/integrated kernel functions

`indicator`	Uniform kernel function
`integrated.quadratic`	Integrated quadratic kernel function
`integrated.triangle`	Integrated triangle kernel function
`quadratic`	Quadratic kernel function
`triangle`	Triangle kernel function

1.2.4 Predicting a scalar response from curves or Forecasting time series

`funopare.kernel`	Performs functional prediction (regression) of a scalar response from a sample of curves via the functional kernel estimator. A global bandwidth is considered without automatic selection
`funopare.kernel.cv`	Performs functional prediction (regression) of a scalar response from a sample of curves via the functional kernel estimator. A global bandwidth is automatically selected with a cross-validation procedure
`funopare.knn`	Performs functional prediction (regression) of a scalar response from a sample of curves via the functional kernel estimator. A bandwidth corresponding to number of neighbours has to be given
`funopare.knn.gcv`	Performs functional prediction (regression) of a scalar response from a sample of curves via the functional kernel estimator. A global bandwidth (i.e. a number of neighbours) is selected by a cross-validation procedure
`funopare.knn.lcv`	Performs functional prediction (regression) of a scalar response from a sample of curves via the functional kernel estimator. A local bandwidth (i.e. number of neighbours depending on the curve where the estimator is evaluated) is selected by a cross-validation procedure
`funopare.mode.lcv`	Performs functional prediction of a scalar response from a sample of curves by computing the functional conditional mode. A local bandwidth (i.e. local number of neighbours) is selected by a "trivial" cross-validation procedure
`funopare.quantile.lcv`	Performs functional prediction of a scalar response from a sample of curves by computing the functional conditional mode. A local bandwidth (i.e. local number of neighbours) is selected by a "trivial" cross-validation procedure

1.2.5 Preprocessing functional data

`approx.fourier`	Fourier approximation of curves
`approx.spline.deriv`	B-spline approximation of the successive derivatives of a set curves
`approx.spline`	B-spline approximation of a set of curves
`fourier`	Fourier approximation of the successive derivatives of a set of curves
`unbal2equibal`	Transforms unbalanced functional data into Balanced functional data

1.2.6 Proximities between curves (semi-metrics)

`hshift`	Returns the "horizontal shifted proximity" between two curves (called by `semimetric.hshift`)
`semimetric.deriv`	Computes between curves a semimetric based on their derivative (via a B-spline expansion)
`semimetric.fourier`	Computes between curves a semimetric based on their fourier expansion
`semimetric.hshift`	Computes between curves a semimetric taking into account an horizontal shift effect
`semimetric.mplsr`	Computes between curves a semimetric based on the partial least squares method
`semimetric.pca`	Computes between curves a semimetric based on the functional principal component analysis

2

Help Files for main routines

2.1 Introduction and notations

We have written R and S+ routines for implementing nonparametric methods for functional datasets. Most of main routines involve one functional dataset (called `DATA1` or `CURVES`) allowing to build estimates and a second functional dataset (called `DATA2` or `PRED`) for obtaining predictions for new observations. Notations are those introduced in our book.

For simplifying our purpose (dataframe, programs,...), we consider only the case of balanced data with equally spaced measurements. It means that we observe a sample of f.r.v. $\{\chi_i\}_{i=1,\dots,n}$ at the J points $\{t_j\}_{j=1,\dots,J}$ with $t_j = t_1 + (j-1)(t_J - t_1)/(J-1)$ and we can build a data matrix $n \times J$, the *ith* row containing the quantities $\boldsymbol{x}_i = (\chi_i(t_1), \dots, \chi_i(t_J))$. Therefore, in the following help files, we suppose that we have at hand matrices `DATA1` (or `CURVES`) and `DATA2` (or `PRED`) such that:

$$\forall i \in \{1,\dots,n\},\ \texttt{DATA1[i,]} = \boldsymbol{x}_i = (\texttt{CURVES[i,]}),$$

and,

$$\forall i' \in \{1,\dots,n'\},\ \texttt{DATA2[i',]} = \boldsymbol{z}_{i'} = (\texttt{PRED[i',]}).$$

Generally, the first matrix `CURVES` (or `DATA1`) corresponds to a first sample of curves using for the estimating procedure. The second matrix `PRED` (or `DATA2`) corresponds to a second sample of curves for which predictions are computed.

In addition, when we have at hand *standard* unbalanced functional data, the routine `unbal2equibal` proposes a pre-processing stage for transforming such unbalanced functional data into equally spaced balanced ones in a simple way (throughout a linear interpolation method).

2.2 Help files in alphabetical order

funopadi.knn.lcv

DESCRIPTION

Performs functional discrimination of a sample of curves when a categorical response is observed (supervised classification). A local bandwidth (i.e. local number of neighbours) is selected by a cross-validation procedure.

USAGE

```
funopadi.knn.lcv(Classes, CURVES, PRED,...,
                 kind.of.kernel = "quadratic",
                 semimetric="deriv")
```

REQUIRED ARGUMENTS

Classes vector containing the categorical responses giving the group number for each curve in the matrix CURVES (if nbclass is the number of groups, the vector "Classes" contains the class numbers 1,2,...,nbclass).

CURVES matrix $n \times J$ containing the curves stored row by row; `CURVES[i,]`$= \boldsymbol{x}_i$.

PRED matrix $n' \times J$ containing new curves stored row by row; `PRED[i,]`$= \boldsymbol{z}_i$.

... arguments needed for the call of the function computing the semimetric.

OPTIONAL ARGUMENTS

kind.of.kernel the kernel function K used for the computation of the kernel estimator; you can choose `indicator`, `triangle` or `quadratic` (default).

semimetric character string allowing to choose the function computing the semimetric $\boldsymbol{d}(.,.)$; you can choose `"deriv"` (default), `''fourier''`, `"hshift"`, `"mplsr"`, and `"pca"`.

OUTPUT: a list containing

$Estimated.classnumber vector of length n containing the estimated class number for each curve of CURVES.

$Predicted.classnumber if the argument `PRED`$\neq$ `CURVES`, this contains a vector of length n' containing the estimated class number for each curve of PRED.

$Bandwidths vector of length n containing the local data-driven bandwidths (`Bandwidths[i]`$= h_{k_{opt}(\boldsymbol{x}_i)}$).

$Mse mean squared error between estimated values and observed values

CALLED SUBROUTINE:

one among the kernel functions: `indicator`, `triangle`, `quadratic`.

one among the semimetric routines: `semimetric.deriv`, `semimetric.fourier`, `semimetric.hshift`, `semimetric.pca`.

funopare.kernel

DESCRIPTION

Performs functional prediction (regression) of a scalar response from a sample of curves via the functional kernel estimator. A global bandwidth is considered without automatic selection.

USAGE

funopare.kernel(Response, CURVES, PRED, bandwidth,...,
kind.of.kernel = "quadratic",
semimetric="deriv")

REQUIRED ARGUMENTS

Response vector containing the observations of the scalar response $(y_1, \ldots, y_n)$.

CURVES matrix $n \times J$ containing the curves stored row by row; `CURVES[i,]`$= \boldsymbol{x}_i$.

PRED matrix $n' \times J$ containing new curves stored row by row; `PRED[i,]`$= \boldsymbol{z}_i$.

bandwidth the value of the bandwidth (h).

... arguments needed for the call of the function computing the semimetric.

OPTIONAL ARGUMENTS

kind.of.kernel the kernel function K used for the computation of the kernel estimator; you can choose `indicator`, `triangle` or `quadratic` (default).

semimetric character string allowing to choose the function computing the semimetric $\boldsymbol{d}(.,.)$; you can choose `"deriv"` (default), `''fourier''`, `"hshift"`, `"mplsr"`, and `"pca"`.

OUTPUT: a list containing

$Estimated.values vector of length n such that `response.estimated[i]`$= R^{kernel}(\boldsymbol{x}_i)$.

$Predicted.values if the argument `PRED`$\neq$ `CURVES`, this contains a vector of length n' such that `Predicted.values[i]`$= R^{kernel}(\boldsymbol{z}_i)$.

$band value of the current bandwidth.

$Mse mean square error between estimated values and observed values.

CALLED SUBROUTINE:

one among the kernel functions: `indicator`, `triangle`, `quadratic`.

one among the semimetric routines `semimetric.deriv`, `semimetric.fourier`, `semimetric.hshift`, `semimetric.pca`.

SEE ALSO: funopare.kernel.cv, funopare.knn, funopare.knn.gcv, funopare.knn.lcv

funopare.kernel.cv

DESCRIPTION

Performs functional prediction (regression) of a scalar response from a sample of curves via the functional kernel estimator. A global bandwidth is automatically selected with a cross-validation procedure.

USAGE

```
funopare.kernel.cv(Response, CURVES, PRED,...,
                   kind.of.kernel = "quadratic",
                   semimetric="deriv",
                   h.range = NULL)
```

REQUIRED ARGUMENTS

Response vector containing the observations of the scalar response $(y_1, \ldots, y_n)$.

CURVES matrix $n \times J$ containing the curves stored row by row; `CURVES[i,]`$= \boldsymbol{x}_i$.

PRED matrix $n' \times J$ containing new curves stored row by row; `PRED[i,]`$= \boldsymbol{z}_i$.

... arguments needed for the call of the function computing the semimetric.

OPTIONAL ARGUMENTS

kind.of.kernel the kernel function K used for the computation of the kernel estimator; you can choose `indicator`, `triangle` or `quadratic` (default).

semimetric character string allowing to choose the function computing the semimetric $\boldsymbol{d}(.,.)$; you can choose `"deriv"` (default), `''fourier''`, `"hshift"`, `"mplsr"`, and `"pca"`.

h.range vector of length 2 giving the range for the bandwidth. By default, the procedure defines a sequence of candidates for bandwidth according to the values of the matrix `CURVES`.

OUTPUT: a list containing

$Estimated.values vector of length n such that `response.estimated[i]`$= R_{CV}^{kernel}(\boldsymbol{x}_i)$.

$Predicted.values if the argument `PRED`$\neq$ `CURVES`, this contains a vector of length n' such that `Predicted.values[i]`$= R_{CV}^{kernel}(\boldsymbol{z}_i)$.

$hopt value of the optimal bandwidth

$hseq used sequence of possible bandwidths

$Mse mean squared error between estimated values and observed values

CALLED SUBROUTINE:

one among the kernel functions: `indicator`, `triangle`, `quadratic`.

one among the semimetric routines: `semimetric.deriv`, `semimetric.fourier`, `semimetric.hshift`, `semimetric.pca`.

SEE ALSO: funopare.kernel, funopare.knn, funopare.knn.gcv, funopare.knn.lcv

funopare.knn

DESCRIPTION

Performs functional prediction (regression) of a scalar response from a sample of curves via the functional kernel estimator. A bandwidth corresponding to number of neighbours is given.

USAGE

funopare.knn(Response, CURVES, PRED, neighbour,...,
kind.of.kernel = "quadratic",
semimetric="deriv")

REQUIRED ARGUMENTS

Response vector containing the observations of the scalar response $(y_1, \ldots, y_n)$.

CURVES matrix $n \times J$ containing the curves stored row by row; `CURVES[i,]`$= \boldsymbol{x}_i$.

PRED matrix $n' \times J$ containing new curves stored row by row; `PRED[i,]`$= \boldsymbol{z}_i$.

neighbour the number (k) of neighbours fixed for computing the functional kernel estimator.

... arguments needed for the call of the function computing the semimetric.

OPTIONAL ARGUMENTS

kind.of.kernel the kernel function K used for the computation of the kernel estimator; you can choose `indicator`, `triangle` or `quadratic` (default).

semimetric character string allowing to choose the function computing the semimetric $\boldsymbol{d}(.,.)$; you can choose `"deriv"` (default), `''fourier''`, `"hshift"`, `"mplsr"`, and `"pca"`.

OUTPUT: a list containing

$Estimated.values vector of length n such that `response.estimated[i]`$= R^{kNN}(\boldsymbol{x}_i)$.

$Predicted.values if the argument `PRED`$\neq$ `CURVES`, this contains a vector of length n' such that `Predicted.values[i]`$= R^{kNN}(\boldsymbol{z}_i)$.

$kNN value of the current argument `neighbour`.

$Mse mean squared error between estimated values and observed values

CALLED SUBROUTINE:

one among the kernel functions: `indicator`, `triangle`, `quadratic`.

one among the semimetric routines: `semimetric.deriv`, `semimetric.fourier`, `semimetric.hshift`, `semimetric.pca`.

SEE ALSO: funopare.kernel, funopare.kernel.cv, funopare.knn.gcv, funopare.knn.lcv

funopare.knn.gcv

DESCRIPTION

Performs functional prediction (regression) of a scalar response from a sample of curves via the functional kernel estimator. A global bandwidth (i.e. a number of neighbours) is selected by a cross-validation procedure.

USAGE

```
funopare.knn.gcv(Response, CURVES, PRED,...,
                 kind.of.kernel = "quadratic",
                 semimetric="deriv")
```

REQUIRED ARGUMENTS

Response vector containing the observations of the scalar response $(y_1, \ldots, y_n)$.

CURVES matrix $n \times J$ containing the curves stored row by row; `CURVES[i,]`$= \boldsymbol{x}_i$.

PRED matrix $n' \times J$ containing new curves stored row by row; `PRED[i,]`$= \boldsymbol{z}_i$.

... arguments needed for the call of the function computing the semimetric.

OPTIONAL ARGUMENTS

kind.of.kernel the kernel function K used for the computation of the kernel estimator; you can choose `indicator`, `triangle` or `quadratic` (default).

semimetric character string allowing to choose the function computing the semimetric $\boldsymbol{d}(.,.)$; you can choose `"deriv"` (default), `''fourier''`, `"hshift"`, `"mplsr"`, and `"pca"`.

OUTPUT: a list containing

$Estimated.values vector of length n such that `response.estimated[i]`$= R^{kNN}_{GCV}(\boldsymbol{x}_i)$.

$Predicted.values if the argument `PRED`$\neq$ `CURVES`, this contains a vector of length n' such that `Predicted.values[i]`$= R^{kNN}_{GCV}(\boldsymbol{z}_i)$.

$Bandwidths vector of length n containing the data-driven bandwidths (`Bandwidths[i]`$= h_{k_{opt}}(\boldsymbol{x}_i)$).

$knearest.opt contains the optimal number of neighbours (`knearest`$= k_{opt}$).

$Mse mean squared error between estimated values and observed values

CALLED SUBROUTINE:

one among the kernel functions: `indicator`, `triangle`, `quadratic`.
one among the semimetric routines: `semimetric.deriv`, `semimetric.fourier`, `semimetric.hshift`, `semimetric.pca`.

SEE ALSO: funopare.kernel, funopare.kernel.cv, funopare.knn, funopare.knn.lcv

funopare.knn.lcv

DESCRIPTION

Performs functional prediction (regression) of a scalar response from a sample of curves via the functional kernel estimator. A local bandwidth (i.e. local number of neighbours) is selected by a cross-validation procedure.

USAGE

```
funopare.knn.lcv(Response, CURVES, PRED,...,
                 kind.of.kernel = "quadratic",
                 semimetric="deriv")
```

REQUIRED ARGUMENTS

Response vector containing the observations of the scalar response $(y_1, \ldots, y_n)$.

CURVES matrix $n \times J$ containing the curves stored row by row; `CURVES[i,]`$= \boldsymbol{x}_i$.

PRED matrix $n' \times J$ containing new curves stored row by row; `PRED[i,]`$= \boldsymbol{z}_i$.

... arguments needed for the call of the function computing the semimetric.

OPTIONAL ARGUMENTS

kind.of.kernel the kernel function K used for the computation of the kernel estimator; you can choose `indicator`, `triangle` or `quadratic` (default).

semimetric character string allowing to choose the function computing the semimetric $\boldsymbol{d}(.,.)$; you can choose `"deriv"` (default), `''fourier''`, `"hshift"`, `"mplsr"`, and `"pca"`.

OUTPUT: a list containing

$Estimated.values vector of length n such that `response.estimated[i]`$= R^{kNN}_{LCV}(\boldsymbol{x}_i)$.

$Predicted.values if the argument `PRED`$\neq$ `CURVES`, this contains a vector of length n' such that `Predicted.values[i]`$= R^{kNN}_{LCV}(\boldsymbol{z}_i)$.

$Bandwidths vector of length n containing the data-driven bandwidths (`Bandwidths[i]`$= h_{k_{opt}(\boldsymbol{x}_i)}$).

$Mse mean squared error between estimated values and observed values

CALLED SUBROUTINE:

one among the kernel functions: `indicator`, `triangle`, `quadratic`.

one among the semimetric routines: `semimetric.deriv`, `semimetric.fourier`, `semimetric.hshift`, `semimetric.pca`.

SEE ALSO: funopare.kernel, funopare.kernel.cv, funopare.knn, funopare.knn.gcv

funopare.mode.lcv

DESCRIPTION

Performs functional prediction of a scalar response from a sample of curves by computing the functional conditional mode. A local bandwidth (i.e. local number of neighbours) is selected by a “trivial” cross-validation procedure.

USAGE

```
funopare.mode.lcv(Response, CURVES, PRED,...,
                  Knearest = NULL,
                  kind.of.kernel = "quadratic",
                  semimetric="deriv")
```

REQUIRED ARGUMENTS

Response vector containing the observations of the scalar response $(y_1, \ldots, y_n)$.

CURVES matrix $n \times J$ containing the curves stored row by row; `CURVES[i,]`$= \boldsymbol{x}_i$.

PRED matrix $n' \times J$ containing new curves stored row by row; `PRED[i,]`$= \boldsymbol{z}_i$.

... arguments needed for the call of the function computing the semimetric.

OPTIONAL ARGUMENTS

Knearest vector giving the the sequence of successive authorized integers for k_{opt} and κ_{opt}. By default (i.e. `Knearest=NULL`), the vector `Knearest` contains a sequence of 10 integers taking into account $card(I_1)$.

kind.of.kernel the kernel function K used for the computation of the kernel estimator; you can choose `indicator`, `triangle` or `quadratic` (default).

semimetric character string allowing to choose the function computing the semimetric $\boldsymbol{d}(.,.)$; you can choose `"deriv"` (default), `''fourier''`, `"hshift"`, `"mplsr"`, and `"pca"`.

OUTPUT: a list containing

$Estimated.values vector of length n such that `response.estimated[i]`$= \theta^{kNN}(\boldsymbol{x}_i)$.

$Predicted.values if the argument `PRED`$\neq$ `CURVES`, this contains a vector of length n' such that `Predicted.values[i]`$= \theta^{kNN}(\boldsymbol{z}_i)$.

$Response.values vector of length card(I_2) such that, for all `i` in I_2, `Response.values[i]`$= y_i$ (i.e. observed responses corresponding to the second learning subsample).

$Mse mean squared error between estimated values and observed values

CALLED SUBROUTINE:

one among the kernel functions: `indicator`, `triangle`, `quadratic`.

one among the integrated kernel functions: `integrated.triangle`, `integrated.quadratic`.
one among the semimetric routines: `semimetric.deriv`, `semimetric.fourier`, `semimetric.hshift`, `semimetric.pca`.

funopare.quantile.lcv

DESCRIPTION

Performs functional prediction of a scalar response from a sample of curves by computing the functional conditional quantiles. A local bandwidth (i.e. local number of neighbours) is selected by a "trivial" cross-validation procedure.

USAGE

```
funopare.quantile.lcv(Response, CURVES, PRED,...,
                      alpha = c(0.05, 0.5, 0.95),
                      Knearest = NULL,
                      kind.of.kernel = "quadratic",
                      semimetric="deriv")
```

REQUIRED ARGUMENTS

Response vector containing the observations of the scalar response $(y_1, \dots, y_n)$.

CURVES matrix $n \times J$ containing the curves stored row by row; `CURVES[i,]`$= \boldsymbol{x}_i$.

PRED matrix $n' \times J$ containing new curves stored row by row; `PRED[i,]`$= \boldsymbol{z}_i$.

... arguments needed for the call of the function computing the semimetric.

OPTIONAL ARGUMENTS

alpha vector giving the quantiles to be computed. By default, the 5-percentile, median and 95-percentile are computed.

Knearest vector giving the the sequence of successive authorized integers for k_{opt} and κ_{opt}. By default (i.e. `Knearest=NULL`), the vector `Knearest` contains a sequence of 10 integers taking into account $card(I_1)$.

kind.of.kernel the kernel function K used for the computation of the kernel estimator; you can choose `indicator`, `triangle` or `quadratic` (default).

semimetric character string allowing to choose the function computing the semimetric $\boldsymbol{d}(.,.)$; you can choose `"deriv"` (default), `''fourier''`, `"hshift"`, `"mplsr"`, and `"pca"`.

OUTPUT: a list containing

$Estimated.values a $card(I_2)\times$length(alpha)-matrix such that, for all `i` in the set I_2, `Estimated.values[i,]`$= t^{kNN}_{alpha[1]}(\boldsymbol{x}_i), t^{kNN}_{alpha[2]}(\boldsymbol{x}_i), \dots$

$Predicted.values if the argument `PRED`$\neq$ `CURVES`, this contains a $n'\times$length(alpha)-matrix such that `Predicted.values[i,]`$= t^{kNN}_{alpha[1]}(\boldsymbol{z}_i), t^{kNN}_{alpha[2]}(\boldsymbol{x}_i), \dots$.

$Response.values vector of length card(I_2) such that, for all `i` in I_2, `Response.values[i]`$= y_i$ (i.e. observed responses corresponding to the second learning subsample).

$Mse mean squared error between estimated values and observed values

CALLED SUBROUTINE:

one among the kernel functions: `indicator`, `triangle`, `quadratic`.
one among the integrated kernel functions: `integrated.triangle`, `integrated.quadratic`.
one among the semimetric routines: `semimetric.deriv`, `semimetric.fourier`, `semimetric.hshift`, `semimetric.pca`.

semimetric.deriv

DESCRIPTION

Achieves a semimetric based on the succesive derivatives.

USAGE

semimetric.deriv(DATA1, DATA2, q, nknot, range.grid)

REQUIRED ARGUMENTS

DATA1 matrix $n \times J$ containing a first set of curves stored row by row; `DATA1[i,j]`$= \chi_i(t_j)$.

DATA2 matrix $n' \times J$ containing a second set of curves stored row by row; `DATA2[i',j]`$= \chi_{i'}(t_j)$.

q order of derivation.

nknot number of interior knots (needed for defining the B-spline basis).

Range.grid vector of length 2 containing the range of the grid $t_1, \ldots, t_J$ (`Range.grid[1]`$= t_1$ and `Range.grid[2]`$= t_J$).

OUTPUT: a matrix `SEMIMETRIC` such that:

if `DATA2` and `DATA1` are the same, the $n \times n$ matrix
`SEMIMETRIC[i,i']`$= \boldsymbol{d}_q^{deriv}(\boldsymbol{x}_i, \boldsymbol{x}_{i'})$,
else this function returns the $n \times n'$ matrix
`SEMIMETRIC[i,i']`$= \boldsymbol{d}_q^{deriv}(\boldsymbol{x}_i, \boldsymbol{z}_{i'})$.

CALLED SUBROUTINE: symsolve

SEE ALSO

semimetric.fourier, semimetric.hshift, semimetric.mplsr, semimetric.pca

semimetric.hshift

DESCRIPTION

Computes between curves a semimetric taking into account an horizontal shift effect.

USAGE

semimetric.deriv(DATA1, DATA2, grid)

REQUIRED ARGUMENTS

DATA1 matrix $n \times J$ containing a first set of curves stored row by row; `DATA1[i,j]`$= \chi_i(t_j)$.

DATA2 matrix $n' \times J$ containing a second set of curves stored row by row; `DATA2[i',j]`$= \chi_{i'}(t_j)$.

grid vector of length J which defines the grid $t_1, \ldots, t_J$ (because one considers only equispaced grid, `1:J` can be chosen).

OUTPUT: a matrix `SEMIMETRIC` such that:

if `DATA2` and `DATA1` are the same, the $n \times n$ matrix
`SEMIMETRIC[i,i']`$= \boldsymbol{d}_q^{deriv}(\boldsymbol{x}_i, \boldsymbol{x}_{i'})$,
else this function returns the $n \times n'$ matrix
`SEMIMETRIC[i,i']`$= \boldsymbol{d}_q^{deriv}(\boldsymbol{x}_i, \boldsymbol{z}_{i'})$.

CALLED SUBROUTINE: hshift

SEE ALSO

semimetric.deriv, semimetric.fourier, semimetric.mplsr, semimetric.pca

semimetric.mplsr

DESCRIPTION

Achieves a mplsr-type semimetric based on the multivariate partial least-squares regression method.

USAGE

semimetric.mplsr(Classe1, DATA1, DATA2, q)

REQUIRED ARGUMENTS

Classe1 vector of length n containing a categorical response which corresponds to class number for units stored in `DATA1`.

DATA1 matrix $n \times J$ containing a first set of curves stored row by row; `DATA1[i,j]`$= \chi_i(t_j)$.

DATA2 matrix $n' \times J$ containing a second set of curves stored row by row; `DATA2[i',j]`$= \chi_{i'}(t_j)$.

q the retained number of factors.

OUTPUT: a matrix `SEMIMETRIC` such that:

if `DATA2` and `DATA1` are the same, the $n \times n$ matrix

`SEMIMETRIC[i,i']`$= \boldsymbol{d}_q^{PLS}(\boldsymbol{x}_i, \boldsymbol{x}_{i'})$,

else this function returns the $n \times n'$ matrix

`SEMIMETRIC[i,i']`$= \boldsymbol{d}_q^{PLS}(\boldsymbol{x}_i, \boldsymbol{z}_{i'})$.

CALLED SUBROUTINE: mplsr

SEE ALSO

semimetric.deriv, semimetric.fourier, semimetric.hshift, semimetric.pca

semimetric.pca

DESCRIPTION

Achieves a pca-type semimetric based on the functional principal components analysis method.

USAGE

semimetric.pca(DATA1, DATA2, q)

REQUIRED ARGUMENTS

DATA1 matrix $n \times J$ containing a first set of curves stored row by row; `DATA1[i,j]`$= \chi_i(t_j)$.

DATA2 matrix $n' \times J$ containing a second set of curves stored row by row; `DATA2[i',j]`$= \chi_{i'}(t_j)$.

q the retained dimension for the reduced dimensional space.

OUTPUT: a matrix `SEMIMETRIC` such that:

if `DATA2` and `DATA1` are the same, the $n \times n$ matrix
`SEMIMETRIC[i,i']`$= \boldsymbol{d}_q^{PCA}(\boldsymbol{x}_i, \boldsymbol{x}_{i'})$,
else this function returns the $n \times n'$ matrix
`SEMIMETRIC[i,i']`$= \boldsymbol{d}_q^{PCA}(\boldsymbol{x}_i, \boldsymbol{z}_{i'})$.

SEE ALSO

semimetric.deriv, semimetric.fourier, semimetric.hshift, semimetric.mplsr.

unbal2equibal

DESCRIPTION

Transforms an unbalanced functional dataset into an equally spaced balanced via linear interpolation.

USAGE

unbal2equibal(DATA, INSTANTS, Range.grid, lnewgrid)

REQUIRED ARGUMENTS

DATA matrix $n \times J_{max}$ containing the set of unbalanced curves stored row by row; `DATA[i,j]`$= \chi_i(t_{i,j})$ and $J_{max} = max_i J_i$. As soon as $J_i < J_{max}$, the *ith* row of `DATA` is completed with NA's.

INSTANTS matrix $n \times J_{max}$ containing the set of design points stored row by row; `INSTANTS[i,j]`$= t_{i,j}$ ($J_{max} = max_i J_i$). As soon as $J_i < J_{max}$, the *ith* row of `INSTANTS` is completed with NA's.

Range.grid vector of length 2 containing the range of the desired grid (`Range.grid[1]`$= t_1$ and `Range.grid[2]`$= t_J$).

lnewgrid length of the desired equally spaced grid $t_1, t_2, \ldots, t_J$.

OUTPUT: a $n \times J$ matrix `EQUIBAL` containing the approximated functional data in an equally balanced way (`EQUIBAL[i,j]`$= \widetilde{\chi}_i(t_j)$).

CALLED SUBROUTINE: approx

Index