Frédéric Ferraty and Philippe Vieu

Reference manual for implementing NonParametric Functional Data Analysis (NPFDA)

Companion manual of the book: NonParametric Functional Data Analysis: Theory and Practice, Springer-Verlag (New York), ISBN 0387303693 (2006).

June 2, 2006

Contents

List of Figures

Introduction

This document contains most of the *pdf* files that you can download from the NPFDA's website. This reference manual gives an overview on the various materials available for implementing the NPFDA-type statistical methods. In particular, you will find descriptions of functional datasets, case studies, R/S-plus routines, help files,...

Recall that methodologies and their mathematical supports are provided in the NPFDA's book by Ferraty and Vieu (2006, Springer Series in Statistics).

This document is regularly updated and this is the reason why some of the data and case studies presented here were not included in the NPFDA book.

So, have a good reading and good learning!

Part I

Functional Datasets

1

Electricity consumption

1.1 Brief description

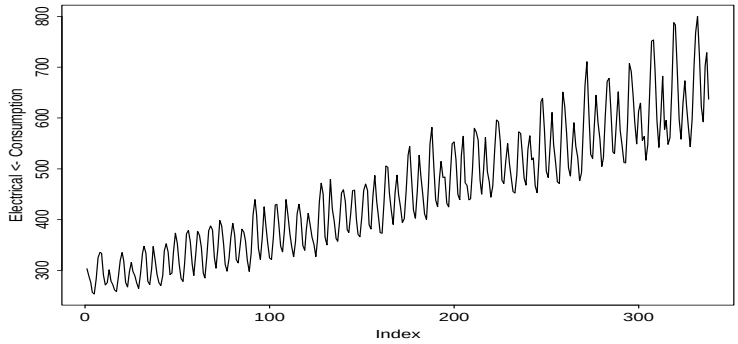
This dataset is a part of the original one which can be found at *http://www.economagic.com*. This time series focus on the US monthly electricity consumed by the residential and commercial sectors from January 1973 up to February 2001 (338 months). For our purpose we retain the last 28 years (336 months) and we eliminated the heteroscedasticity and the linear trend by differencing the log-data. Finally, this time series $\{z_i\}_{i=1,\ldots,336}$ is loaded into a 28×12 matrix, the ith corresponding to the ith year. The file "npfda-electricity.dat" is organized as follows:

	Col 1	$\cdots$	Col j	$\cdots$	Col 12
Row 1	z_1	$\cdots$	z_j	$\cdots$	z_{12}
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$
Row i	$z_{1+12(i-1)}$	$\cdots$	$z_{j+12(i-1)}$	$\cdots$	z_{12i}
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$
Row 28	z_{325}	$\cdots$	z_{324+j}	$\cdots$	z_{336}

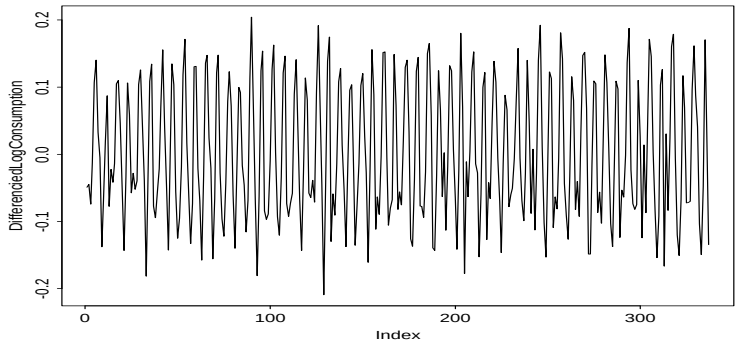
This time series can be viewed as dependent functional data (i.e population of 28 curves, 1 year = 1 curve).

1.2 Plot

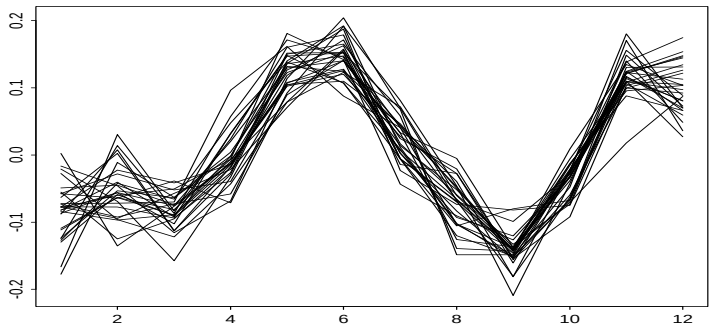
Original electricity consumption time series



Differentiated log electricity consumption time series



The 28 yearly differentiated log curves: time series as functional data (28 rows of the file "npfda-electricity.dat")



2

El Niño time series

2.1 Brief description

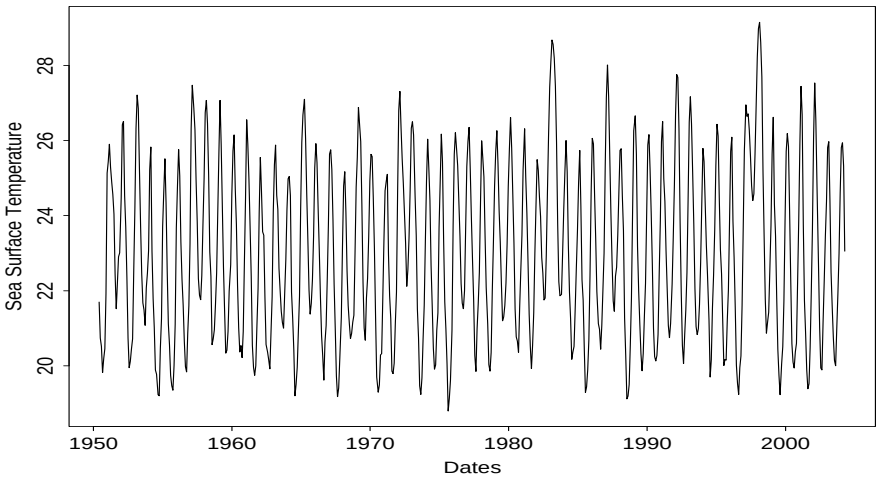
This dataset is a part of the original one which is available on line at *http://www.cpc.ncep.noaa.gov/data/indices/* and regularly updated. This dataset is not included in the NPFDA book. This time series concerns the monthly Sea surface Temperature from June, 1950 up to May, 2004 (648 months). This time series $\{z_i\}_{i=1,\ldots,648}$ is loaded into a 54×12 matrix, the ith corresponding to the ith year. The file "npfda-elnino.dat" is organized as follows:

	Col 1	$\cdots$	Col j	$\cdots$	Col 12
Row 1	z_1	$\cdots$	z_j	$\cdots$	z_{12}
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$
Row i	$z_{1+12(i-1)}$	$\cdots$	$z_{j+12(i-1)}$	$\cdots$	z_{12i}
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$
Row 54	z_{633}	$\cdots$	z_{632+j}	$\cdots$	z_{648}

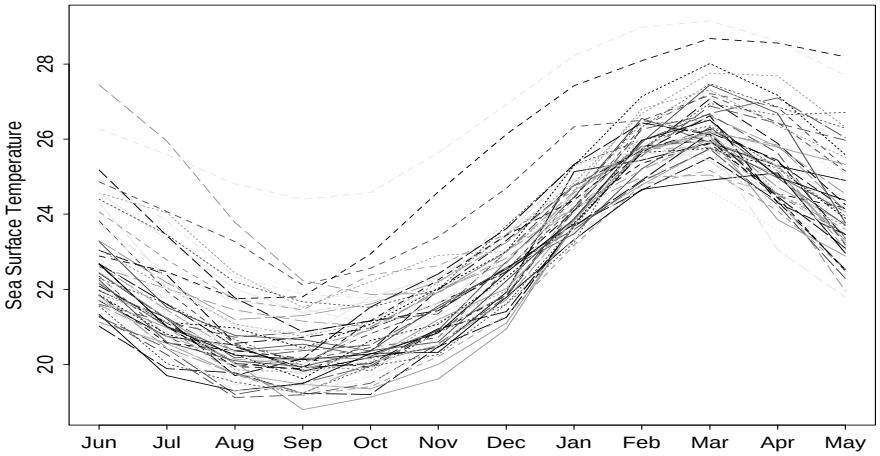
This time series can be viewed as dependent functional data (i.e population of 54 curves, 1 year = 1 curve).

2.2 Plot

El Niño time series (0-10South and 90West-80West)



The 54 yearly curves: time series as functional data
(54 rows of the file "npfda-elnino.dat")



3

Phoneme: log-periodograms

3.1 Brief description

This dataset is a part of the original one which can be found at *http://www-stat.stanford.edu/ElemStatLearn*. We observe $n = 2000$ pairs $(\boldsymbol{x}_i, y_i)_{i=1,\ldots,n}$ where the $\boldsymbol{x}_i$'s correspond to the discretized log-periodograms ($\boldsymbol{x}_i = (\chi(f_1), \chi(f_2), \ldots, \chi(f_{150}))$ is the ith discretized functional data) whereas the y_i's give the class membership (five phonemes):

$$y_i \in \{1,2,3,4,5\} \text{ with } \begin{cases} 1 \longleftrightarrow \text{“sh”} \\ 2 \longleftrightarrow \text{“iy”} \\ 3 \longleftrightarrow \text{“dcl”} \\ 4 \longleftrightarrow \text{“aa”} \\ 5 \longleftrightarrow \text{“ao”} \end{cases}$$

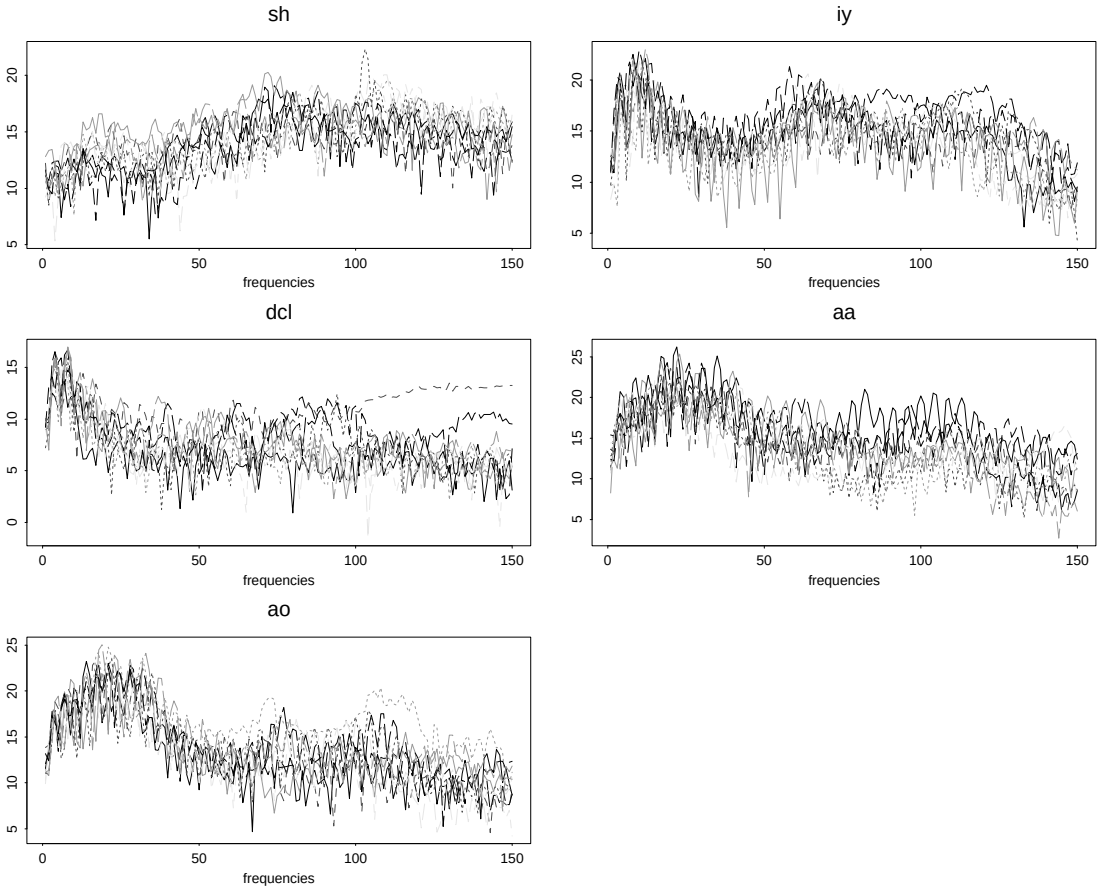
The phoneme dataset “npfda-phoneme.dat” contains the pairs $(\boldsymbol{x}_i, y_i)_{i=1,\ldots,2000}$ and is organized as follows:

	Col 1	$\cdots$	Col j	$\cdots$	Col 150	Col 151
Row 1	$\chi_1(f_1)$	$\cdots$	$\chi_1(f_j)$	$\cdots$	$\chi_1(f_{150})$	y_1
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$
Row i	$\chi_i(f_1)$	$\cdots$	$\chi_i(f_j)$	$\cdots$	$\chi_i(f_{150})$	y_i
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$
Row 2000	$\chi_{2000}(f_1)$	$\cdots$	$\chi_{2000}(f_j)$	$\cdots$	$\chi_{2000}(f_{150})$	y_{2000}

The first 150 columns correspond to the 150 frequencies whereas the last column contains the categorical responses (class number). Note that the size of each class is the same (400).

3.2 Plot

A sample of 10 log-periodograms for each phoneme class:



4

Spectrometric curves

4.1 Brief description

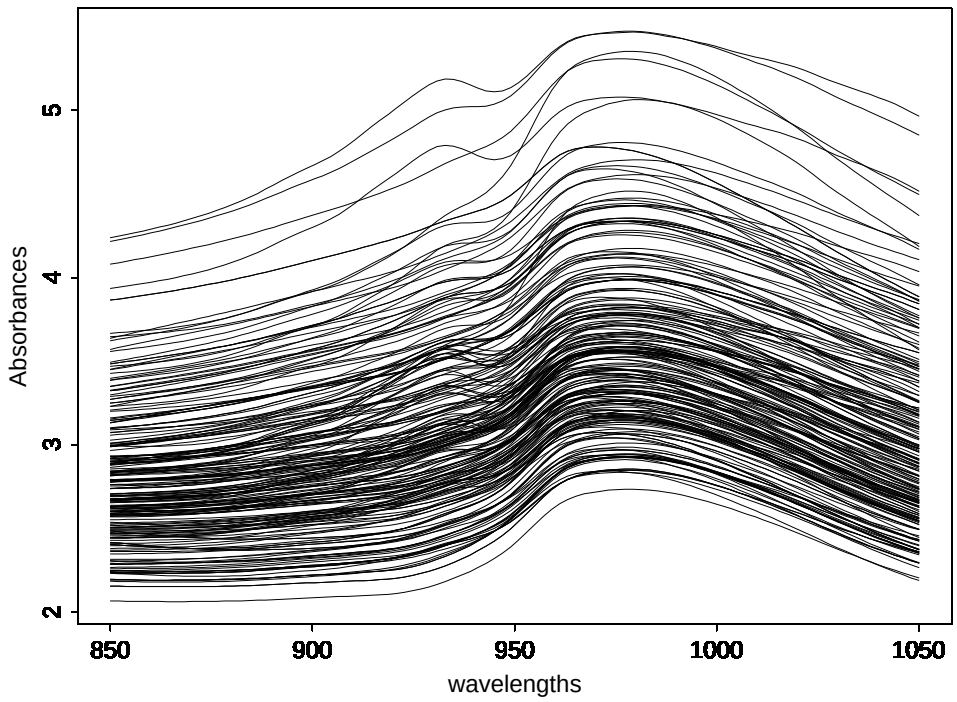
This dataset is a part of the original one which can be found at *http://lib.stat.cmu.edu/datasets/tecator*. For each unit i (among 215 pieces of finely chopped meat), we observe one spectrometric curve ($\boldsymbol{x}_i$) which corresponds to the absorbance measured at 100 wavelengths (i.e. $\boldsymbol{x}_i = (\chi_i(\lambda_1), \ldots, \chi_i(\lambda_{100}))$. Moreover, for each unit i, we have at hand its fat content y_i obtained by an analytical chemical processing. The file "npfda-spectrometric.dat" contains the pairs $(\boldsymbol{x}_i, y_i)_{i=1,\ldots,215}$ and is organized as follows:

	Col 1	$\cdots$	Col j	$\cdots$	Col 100	Col 101
Row 1	$\chi_1(\lambda_1)$	$\cdots$	$\chi_1(\lambda_j)$	$\cdots$	$\chi_1(\lambda_{100})$	y_1
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$
Row i	$\chi_i(\lambda_1)$	$\cdots$	$\chi_i(\lambda_j)$	$\cdots$	$\chi_i(\lambda_{100})$	y_i
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$
Row 215	$\chi_{215}(\lambda_1)$	$\cdots$	$\chi_{215}(\lambda_j)$	$\cdots$	$\chi_{215}(\lambda_{100})$	y_{215}

The first 100 columns correspond to the 100 channel spectrum whereas the last column contains the responses.

4.2 Plot

Spectrometric curves:



5

Satellite dataset: radar waveforms

5.1 Brief description

This dataset is a part of the original ones supplied by Frédéric Frappart and a deep description can be found in Frappart (2003). This dataset is not included in the NPFDA book.

Here, we have only $n = 472$ radar waveforms. The data were registered by the satellite Topex/Poseidon around an area of 25 kilometers upon the Amazon River. Each data is represented by its wave (i.e. curve) on the range $(0, 70)$, and the satellite is registering 10 curves each second. We observe their discretized version namely, for $i = 1, \ldots, 472$, $\boldsymbol{x}_i = (\chi_i(t_1), \ldots, \chi_i(t_{70}))$. Note that each wave is linked with the kind of ground treated by the satellite, and the idea for the Amazonian basin is to use these data for altimetric and hydrological purpose. The satellite dataset "npfda-sat.dat" contains the $\boldsymbol{x}_i$'s and is organized as follows:

	Col 1	$\cdots$	Col j	$\cdots$	Col 70
Row 1	$\chi_1(t_1)$	$\cdots$	$\chi_1(t_j)$	$\cdots$	$\chi_1(t_{70})$
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$
Row i	$\chi_i(t_1)$	$\cdots$	$\chi_i(t_j)$	$\cdots$	$\chi_i(t_{70})$
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$
Row 472	$\chi_{472}(t_1)$	$\cdots$	$\chi_{472}(t_j)$	$\cdots$	$\chi_{472}(t_{70})$

Frappart, F. (2003). *Catalogue des formes d'onde de l'altimètre Topex/Poséidon sur le bassin amazonien*. Technical Report, CNES, Toulouse, France.

5.2 Plot

Figure 5.1 contains a set of 20 randomly selected among these curves. It seems there are various forms of waves: curves with one heavy peak (for instance curves numbers 21 and 22); curves with one less heavy peak (for instance curves numbers 3 and 133); curves that look to have more than one peak (for instance curves numbers 1 and 7); curves that look without real peak (for instance curves numbers 5 and 24); "flat noised curves" (for instance curve number 4); .... In addition, one can remark an horizontal shift (see for instance curves 21 and 22; their shapes are very similar but the peaks are shifted).

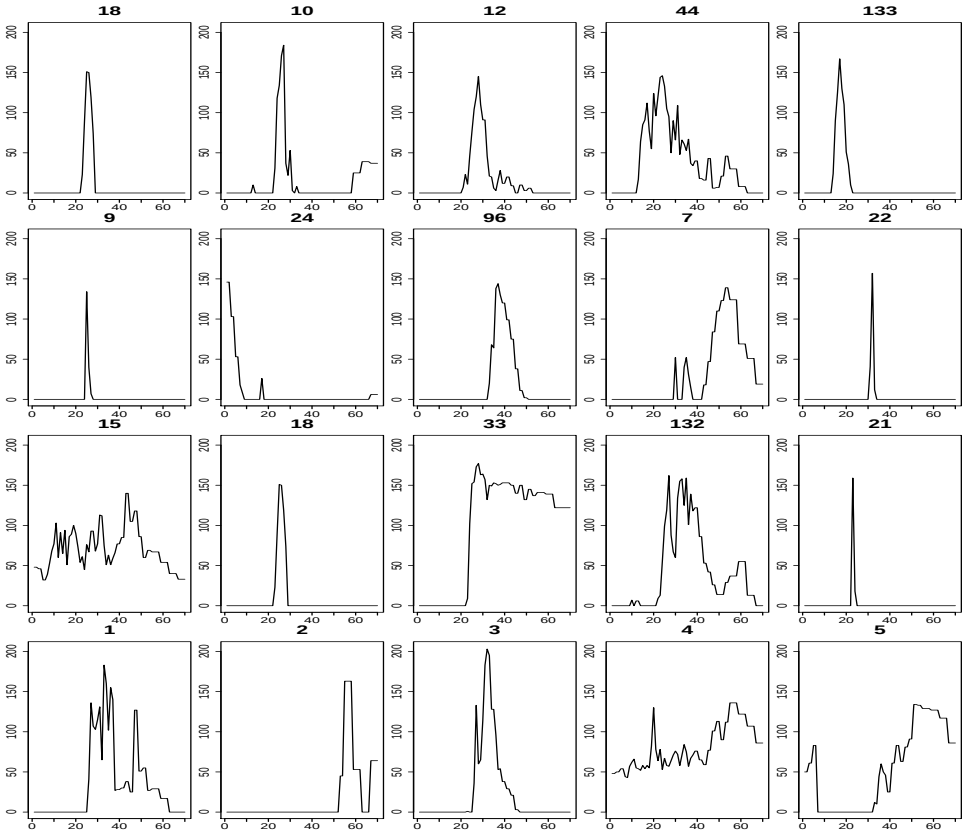


Fig. 5.1. Some radar waveforms

Part II

Case Studies

6

Prediction of a scalar response from curves (see Part II of the NPFDA book)

6.1 Predicting fat content from spectrometric curves

- **Statistical aim**

  We recall that for each unit i (among 215 pieces of finely chopped meat), we observe one spectrometric curve $(\boldsymbol{x}_i)$ which corresponds to the absorbance measured at 100 wavelengths (i.e. $\boldsymbol{x}_i = (\chi_i(\lambda_1), \ldots, \chi_i(\lambda_{100}))$. Moreover, for each unit i, we have at hand its fat content y_i obtained by an analytical chemical processing. The file "npfda-spectrometric.dat" contains the pairs $(\boldsymbol{x}_i, y_i)_{i=1,\ldots,215}$. Given a new spectrometric curve $\boldsymbol{x}$, our main task is to predict the corresponding fat content $\widehat{y}$.

- **Measuring performance**

  In order to highlight the performance of our functional prediction methods, we split our original sample into two samples. The first one, called *learning sample*, contains the first 160 units $((\boldsymbol{x}_i, y_i)_{i=1,\ldots,160})$. The second one, called *testing sample*, contains the last 55 units $((\boldsymbol{x}_i, y_i)_{i=161,\ldots,215})$. The learning sample allows to build the functional kernel estimator with optimal smoothing parameter(s); both the $\boldsymbol{x}_i$'s and the corresponding y_i's are used at this stage. The testing sample is useful for achieving predictions and measuring their quality; we evaluate the functional kernel estimator (obtained with the learning sample) at $\boldsymbol{x}_{161}, \ldots, \boldsymbol{x}_{215}$ ($y_{161}, \ldots, y_{215}$ being ignored) which allows us to get the predicted responses $\widehat{y}_{161}, \ldots, \widehat{y}_{215}$. The smooth shape of the curves allows to use semi-metrics based on the derivatives. The results are presented here d_2^{deriv}.
  To measure the performance of each functional prediction method, we consider
  i) the distribution of the *Square Errors*: $se_i = (y_i - \widehat{y}_i)^2$, $i = 161, \ldots, 215$,

ii) the *Mean Square Errors*: $MSE = \frac{1}{55}\sum_{i=161}^{215} se_i$.

Finally, we run the three routines `funopare.knn.lcv`, `funopare.mode.lcv` and `funopare.quantile.lcv` on the spectrometric dataset, corresponding to the three prediction methods: the conditional expectation (i.e. regression) method, the conditional mode one and the conditional median one.

Remark: the commandlines for R or S+ are the same.

- **Entering spectrometric data**

```
SPECDAT <- as.matrix(read.table("npfda-spectrometric.dat"))
attributes(SPECDAT)$dimnames[[1]] <- character(0)
SPECURVES <- SPECDAT[,1:100]           # sample of curves
Response <- SPECDAT[,101]              # sample of responses
learning <- 1:160
testing <- 161:215
SPECURVES1 <- SPECURVES[learning,]     # learning sample of curves
SPECURVES2 <- SPECURVES[testing,]      # testing sample of curves
Specresp1 <- Response[learning]        # learning sample of responses
Specresp2 <- Response[testing]         # testing sample of responses
```

- **Computing predicted fat content**

```
result.reg <- funopare.knn.lcv(Specresp1, SPECURVES1, SPECURVES2, 2,
                               nknot=20, c(0,1))
result.mode <- funopare.mode.lcv(Specresp1, SPECURVES1, SPECURVES2, 2,
                                 nknot=20, c(0,1))
result.quantile <- funopare.quantile.lcv(Specresp1, SPECURVES1, SPECURVES2, 2,
                                         nknot=20, c(0,1), alpha=0.5)
```

- **Computing square errors**

```
Se.reg <- (Specresp2 - result.reg$Predicted.values)^2
Se.mode <- (Specresp2 - result.mode$Predicted.values)^2
Se.quantile <- (Specresp2 - result.quantile$Predicted.values)^2
mse.reg <- round(sum(Se.reg)/length(Specresp2),2)
mse.mode <-  round(sum(Se.mode)/length(Specresp2),2)
mse.quantile <-  round(sum(Se.quantile)/length(Specresp2),2)
```

- **Plotting predicted responses**

```
par(mfrow=c(1,3), pty="s") # figure will be drawn row-by-row in an 1 by 3
                           # matrix on the current graphical device,
                           # pty = "s"  generates a square plotting region.
xlimits <- range(Specresp2)
ylimits <- range(c(result.mode$Predicted.values,
                   result.quantile$Predicted.values,
                   result.reg$Predicted.values))
```

```
plot(Specresp2, result.reg$Predicted.values, xlim=xlimits, ylim=ylimits,
     main=paste("Cond. Expect.: MSE=",mse.reg,""),
     xlab="Responses of testing sample", ylab="Predicted responses")
abline(0,1)
plot(Specresp2, result.mode$Predicted.values, xlim=xlimits, ylim=ylimits,
     main=paste("Cond. Mode: MSE=",mse.mode, sep=""),
     xlab="Responses of testing sample", ylab="Predicted responses")
abline(0,1)
plot(Specresp2, result.quantile$Predicted.values, xlim=xlimits, ylim=ylimits,
     main=paste("Cond. Median: MSE=",mse.quantile,""),
     xlab="Responses of testing sample", ylab="Predicted responses")
abline(0,1)
```

These last command allow to build Figure 6.1:

- **Improving the predictions by averaging the three predicted values (three methods)**

```
result.multimethods <- (result.mode$Predicted.values +
                        result.quantile$Predicted.values +
                        result.reg$Predicted.values)/3
Se.multimethods <- (Specresp2 - result.multimethods)^2
mse.multimethods <- round(sum(Se.multimethods)/length(Specresp2),2)
error.names <- c(paste("Cond. Expect.\n\n MSE=",mse.reg,sep=""),
                 paste("Cond. Mode\n\n MSE=",mse.mode,sep=""),
                 paste("Cond. Median\n\n MSE=",mse.quantile,sep=""),
                 paste("multimethods\n\n MSE=",mse.multipethods,sep=""))
par(mfrow=c(1,1))
boxplot(Se.reg, Se.mode, Se.quantile, Se.multimethods, names = error.names)
title("Squares Errors")
```

The 3 last commandlines allow to build Figure 6.2.

Finally, these three functional nonparametric prediction methods have to be viewed not as competitive ones but as complementary alternative ones.

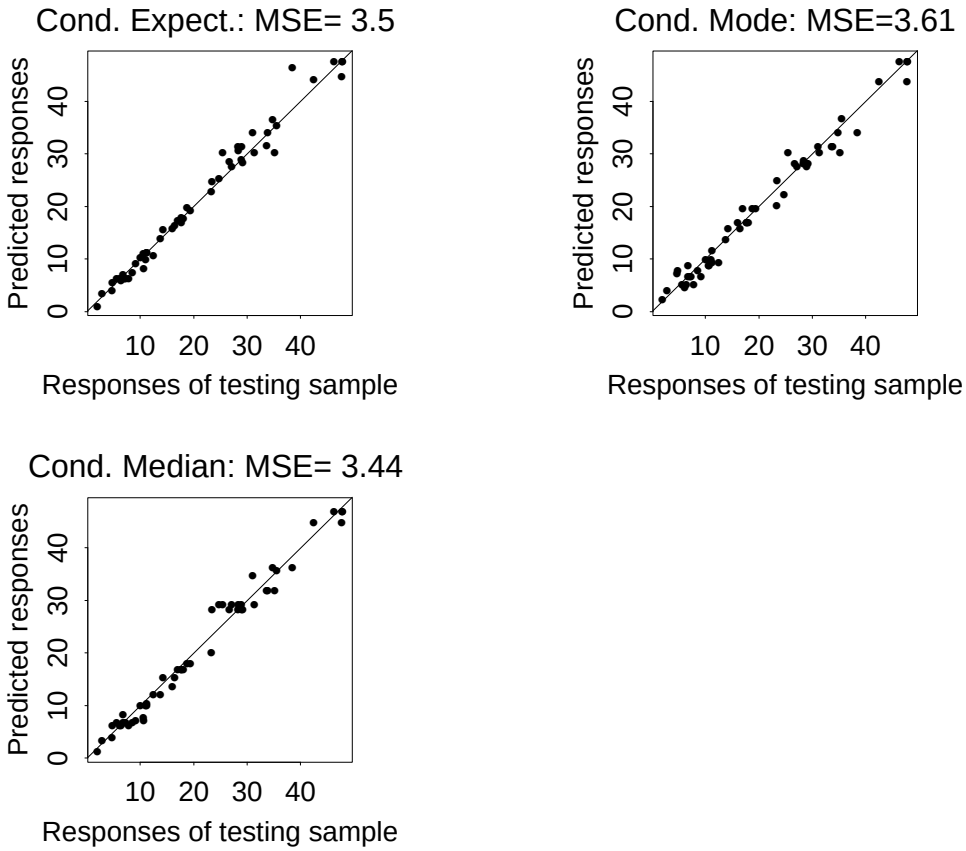


Fig. 6.1. Performance of the three functional prediction methods

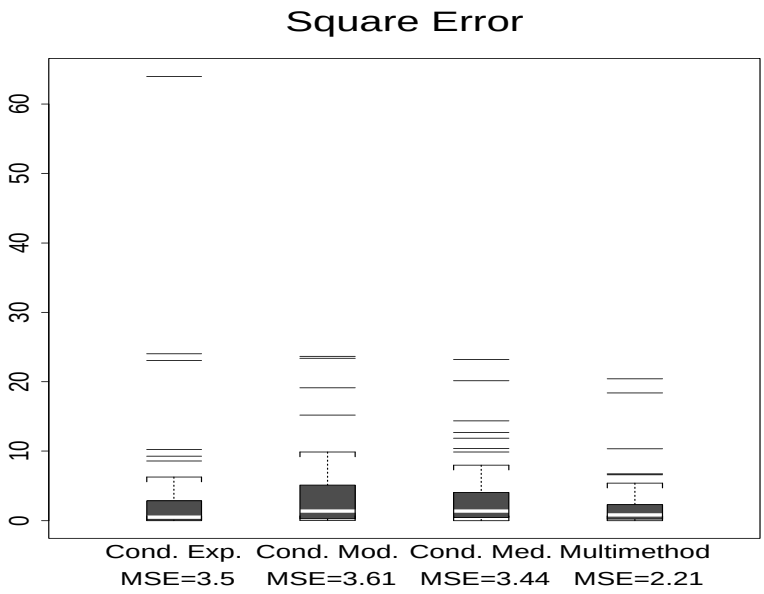


Fig. 6.2. Comparison between the three functional prediction methods and the multimethod one

7

Curves discrimination or supervised classification (see Chapter 8 of the NPFDA book)

7.1 A speech recognition problem: discriminating log-periodograms

- **Statistical aim**

  We recall that we observe $n = 2000$ pairs $(\boldsymbol{x}_i, y_i)_{i=1,\ldots,n}$ where the $\boldsymbol{x}_i$'s correspond to discretized log-periodograms ($\boldsymbol{x}_i = (\chi(f_1), \chi(f_2), \ldots, \chi(f_{150}))$ is the ith discretized functional data) whereas the y_i's give the associated class membership (five phonemes). The file "npfda-phoneme.dat" contains the pairs $(\boldsymbol{x}_i, y_i)_{i=1,\ldots,215}$. Given a new log-periodogram $\boldsymbol{x}$, our main task is to predict the corresponding class of phoneme y^{LCV}.

- **Measuring performance**

  For measuring the performance of our functional nonparametric discrimination method, we build two samples from the original dataset. The first one, the learning sample, contains the 5×50 units ($(\boldsymbol{x}_i, y_i)_{i \in \mathcal{L}}$, each group containing 50 observations). The second one is the testing sample and contains 5×50 units ($(\boldsymbol{x}_{i'}, y_{i'})_{\in i' \mathcal{T}}$ with 50 observations by group). The learning sample allows to estimate the posterior probabilities with optimal smoothing parameter ; both the $\boldsymbol{x}_i$'s and the corresponding y_i's are used at this stage. The testing sample is useful for measuring the discriminant power of such a method; we evaluate the posterior probabilities (obtained with the learning sample) at $\{\boldsymbol{x}_{i'}\}_{i' \in \mathcal{T}}$ ($\{y_{i'}\}_{i' \in \mathcal{T}}$ being ignored) which allows us to get the predicted class membership $\{y_{i'}^{LCV}\}_{i' \in \mathcal{T}}$. It remains to compute the misclassification rate

$$Misclas_{Test} \longleftarrow \frac{1}{250} \sum_{i' \in \mathcal{T}} 1_{[y_{i'} \neq y_{i'}^{LCV}]}.$$

  We repeat 50 times this procedure by building randomly 50 learning samples $\mathcal{L}_1, \ldots, \mathcal{L}_{50}$ and 50 testing samples $\mathcal{T}_1, \ldots, \mathcal{T}_{50}$. Finally, we perform

50 misclassification rates $Misclas_1, \ldots, Misclas_{50}$ and the distribution of these quantities gives a good idea on the discriminant power of such a functional nonparametric supervised classification. This procedure is entirely repeated, by running the routine `funopadi.knn.lcv`, for various semi-metrics in order to highlight the importance of such a proximity measure:

- pca-type semi-metrics (routine semimetric.pca) with a number of dimension taking its values in 4, 5, 6, 7 and 8 successively,
- pls-type semi-metrics (routine semimetric.mplsr) with a number of factors taking its values in 5, 6, 7, 8 and 9 successively,
- derivate-type semi-metrics (routine semimetric.deriv) with a number of derivatives equals to zero (classical L_2 norm; the results obtained with a larger number of derivatives are worse).

Remark: the commandlines for R or S+ are the same.

- **Entering phoneme data**

```
PHONDAT <- as.matrix(read.table("npfda-phoneme.dat"))
attributes(PHONDAT)$dimnames[[1]] <- character(0)
PHONCURVES <- PHONDAT[,1:150]          # sample of curves
Learn.sh <- sample(1:400,50)
Learn.iy <- sample(401:800,50)
Learn.dcl <- sample(801:1200,50)
Learn.aa <- sample(1201:1600,50)
Learn.ao <- sample(1601:2000,50)
Test.sh <- sample((1:400)[-Learn.sh],50)
ind <- (1:800)[-Learn.iy]
Test.iy <- sample(ind[ind>401],50)
ind <- (1:1200)[-Learn.dcl]
Test.dcl <- sample(ind[ind>801],50)
ind <- (1:1600)[-Learn.aa]
Test.aa <- sample(ind[ind>1201],50)
ind <- (1:2000)[-Learn.ao]
Test.ao <- sample(ind[ind>1601],50)
Learning <- c(Learn.sh,Learn.iy,Learn.dcl,Learn.aa,Learn.ao)
Testing <- c(Test.sh,Test.iy,Test.dcl,Test.aa,Test.ao)
PHONLEARN <- PHONCURVES[Learning,]     # learning sample of curves
PHONTEST <- PHONCURVES[Testing,]       # testing sample of curves
Classlearn <- sort(rep(1:5,50))        # learning class numbers
Classtest <-  sort(rep(1:5,50))        # testing class numbers
```

- **Computing predicted class membership and misclassification rates** (for various semi-metrics)

```
res.mplsr5 <- funopadi.knn.lcv(Classlearn,PHONLEARN,PHONTEST,5,
              kind.of.kernel = "quadratic",semimetric="mplsr")
Misclas.mplsr5 <- sum(res.mplsr5$Predicted.classnumber !=
```

```
                          Classtest)/250
res.mplsr6 <- funopadi.knn.lcv(Classlearn,PHONLEARN,PHONTEST,6,
              kind.of.kernel = "quadratic",semimetric="mplsr")
Misclas.mplsr6 <- sum(res.mplsr6$Predicted.classnumber !=
                          Classtest)/250
res.mplsr7 <- funopadi.knn.lcv(Classlearn,PHONLEARN,PHONTEST,7,
              kind.of.kernel = "quadratic",semimetric="mplsr")
Misclas.mplsr7 <- sum(res.mplsr7$Predicted.classnumber !=
                          Classtest)/250
res.mplsr8 <- funopadi.knn.lcv(Classlearn,PHONLEARN,PHONTEST,8,
              kind.of.kernel = "quadratic",semimetric="mplsr")
Misclas.mplsr8 <- sum(res.mplsr8$Predicted.classnumber !=
                          Classtest)/250
res.mplsr9 <- funopadi.knn.lcv(Classlearn,PHONLEARN,PHONTEST,9,
              kind.of.kernel = "quadratic",semimetric="mplsr")
Misclas.mplsr9 <- sum(res.mplsr9$Predicted.classnumber !=
                          Classtest)/250
res.pca4 <- funopadi.knn.lcv(Classlearn,PHONLEARN,PHONTEST,4,
            kind.of.kernel = "quadratic",semimetric="pca")
Misclas.pca4 <- sum(res.pca4$Predicted.classnumber !=
                        Classtest)/250
res.pca5 <- funopadi.knn.lcv(Classlearn,PHONLEARN,PHONTEST,5,
            kind.of.kernel = "quadratic",semimetric="pca")
Misclas.pca5 <- sum(res.pca5$Predicted.classnumber !=
                        Classtest)/250
res.pca6 <- funopadi.knn.lcv(Classlearn,PHONLEARN,PHONTEST,6,
            kind.of.kernel = "quadratic",semimetric="pca")
Misclas.pca6 <- sum(res.pca6$Predicted.classnumber !=
                        Classtest)/250
res.pca7 <- funopadi.knn.lcv(Classlearn,PHONLEARN,PHONTEST,7,
            kind.of.kernel = "quadratic",semimetric="pca")
Misclas.pca7 <- sum(res.pca7$Predicted.classnumber !=
                        Classtest)/250
res.pca8 <- funopadi.knn.lcv(Classlearn,PHONLEARN,PHONTEST,8,
            kind.of.kernel = "quadratic",semimetric="pca")
Misclas.pca8 <- sum(res.pca8$Predicted.classnumber !=
                        Classtest)/250
res.deriv0 <- funopadi.knn.lcv(Classlearn,PHONLEARN,PHONTEST,0,20,c(0,1),
              kind.of.kernel = "quadratic",semimetric="deriv")
Misclas.deriv0 <- sum(res.deriv0$Predicted.classnumber !=
                        Classtest)/250
```

- **Plotting misclassification rates over 1 run**
  The following commandlines allow to obtain Figure 7.1 (don't forget to active your graphics device!):

```
Misclas.rates <- c(Misclas.mplsr5,Misclas.mplsr6,Misclas.mplsr7,
                   Misclas.mplsr8,Misclas.mplsr9,Misclas.pca4,
                   Misclas.pca5,Misclas.pca6,Misclas.pca7,Misclas.pca8,
                   Misclas.deriv0)
Misclas.names <- c("mplsr5","mplsr6","mplsr7","mplsr8","mplsr9","pca4",
                   "pca5","pca6","pca7","pca8","deriv0")
%par(mai=c(0.8,.1,0.1,0.1))
dotchart(Misclas.rates, Misclas.names, cex=1, xlab="MISCLASSIFICATION RATES")
```

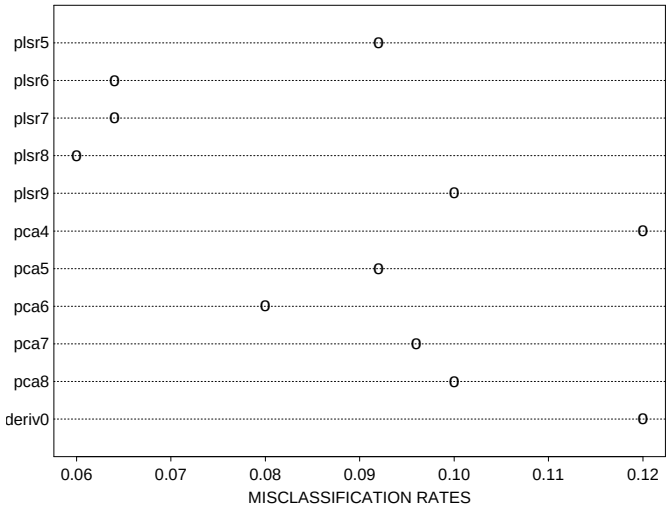


Fig. 7.1. Results for only one run

- **Computing and plotting misclassification rates over 50 runs**
  We repeat 50 times the previous commandlines by means of a loop:

```
Misclas.of.phon.over.50.samples.with.pca4 <- 0
Misclas.of.phon.over.50.samples.with.pca5 <- 0
Misclas.of.phon.over.50.samples.with.pca6 <- 0
Misclas.of.phon.over.50.samples.with.pca7 <- 0
Misclas.of.phon.over.50.samples.with.pca8 <- 0
Misclas.of.phon.over.50.samples.with.mplsr5 <- 0
Misclas.of.phon.over.50.samples.with.mplsr6 <- 0
Misclas.of.phon.over.50.samples.with.mplsr7 <- 0
Misclas.of.phon.over.50.samples.with.mplsr8 <- 0
Misclas.of.phon.over.50.samples.with.mplsr9 <- 0
Misclas.of.phon.over.50.samples.with.deriv0 <- 0
```

```
for(i in 1:50){
        set.seed(sample(0:1000,1))
        Learn.sh <- sample(1:400,50)
        Learn.iy <- sample(401:800,50)
        Learn.dcl <- sample(801:1200,50)
        Learn.aa <- sample(1201:1600,50)
        Learn.ao <- sample(1601:2000,50)
        Test.sh <- sample((1:400)[-Learn.sh],50)
        ind <- (1:800)[-Learn.iy]
        Test.iy <- sample(ind[ind>401],50)
        ind <- (1:1200)[-Learn.dcl]
        Test.dcl <- sample(ind[ind>801],50)
        ind <- (1:1600)[-Learn.aa]
        Test.aa <- sample(ind[ind>1201],50)
        ind <- (1:2000)[-Learn.ao]
        Test.ao <- sample(ind[ind>1601],50)
        Learning <- c(Learn.sh,Learn.iy,Learn.dcl,Learn.aa,Learn.ao)
        Testing <- c(Test.sh,Test.iy,Test.dcl,Test.aa,Test.ao)
        PHONLEARN <- PHONCURVES[Learning,]
        PHONTEST <- PHONCURVES[Testing,]
        res <- funopadi.knn.lcv(Classlearn,PHONLEARN,PHONTEST,4,
          kind.of.kernel = "quadratic",semimetric="pca")
        Misclas.of.phon.over.50.samples.with.pca4[i] <-
          sum(res$Predicted.classnumber != Classtest)/length(Classtest)
        res <- funopadi.knn.lcv(Classlearn,PHONLEARN,PHONTEST,5,
          kind.of.kernel = "quadratic",semimetric="pca")
        Misclas.of.phon.over.50.samples.with.pca5[i] <-
          sum(res$Predicted.classnumber != Classtest)/length(Classtest)
        res <- funopadi.knn.lcv(Classlearn,PHONLEARN,PHONTEST,6,
          kind.of.kernel = "quadratic",semimetric="pca")
        Misclas.of.phon.over.50.samples.with.pca6[i] <-
          sum(res$Predicted.classnumber != Classtest)/length(Classtest)
        res <- funopadi.knn.lcv(Classlearn,PHONLEARN,PHONTEST,7,
          kind.of.kernel = "quadratic",semimetric="pca")
        Misclas.of.phon.over.50.samples.with.pca7[i] <-
          sum(res$Predicted.classnumber != Classtest)/length(Classtest)
        res <- funopadi.knn.lcv(Classlearn,PHONLEARN,PHONTEST,8,
          kind.of.kernel = "quadratic",semimetric="pca")
        Misclas.of.phon.over.50.samples.with.pca8[i] <-
          sum(res$Predicted.classnumber != Classtest)/length(Classtest)
        res <- funopadi.knn.lcv(Classlearn,PHONLEARN,PHONTEST,5,
          kind.of.kernel = "quadratic",semimetric="mplsr")
        Misclas.of.phon.over.50.samples.with.mplsr5[i] <-
          sum(res$Predicted.classnumber != Classtest)/length(Classtest)
        res <- funopadi.knn.lcv(Classlearn,PHONLEARN,PHONTEST,6,
```

```
          kind.of.kernel = "quadratic",semimetric="mplsr")
        Misclas.of.phon.over.50.samples.with.mplsr6[i] <-
          sum(res$Predicted.classnumber != Classtest)/length(Classtest)
        res <- funopadi.knn.lcv(Classlearn,PHONLEARN,PHONTEST,7,
          kind.of.kernel = "quadratic",semimetric="mplsr")
        Misclas.of.phon.over.50.samples.with.mplsr7[i] <-
          sum(res$Predicted.classnumber != Classtest)/length(Classtest)
        res <- funopadi.knn.lcv(Classlearn,PHONLEARN,PHONTEST,8,
          kind.of.kernel = "quadratic",semimetric="mplsr")
        Misclas.of.phon.over.50.samples.with.mplsr8[i] <-
          sum(res$Predicted.classnumber != Classtest)/length(Classtest)
        res <- funopadi.knn.lcv(Classlearn,PHONLEARN,PHONTEST,9,
          kind.of.kernel = "quadratic",semimetric="mplsr")
        Misclas.of.phon.over.50.samples.with.mplsr9[i] <-
          sum(res$Predicted.classnumber != Classtest)/length(Classtest)
        res <- funopadi.knn.lcv(Classlearn,PHONLEARN,PHONTEST,0,30,c(0,1),
          kind.of.kernel = "quadratic",semimetric="deriv")
        Misclas.of.phon.over.50.samples.with.deriv0[i] <-
          sum(res$Predicted.classnumber != Classtest)/length(Classtest)
}
Misclas.mplsr5 <- Misclas.of.phon.over.50.samples.with.mplsr5
Misclas.mplsr6 <- Misclas.of.phon.over.50.samples.with.mplsr6
Misclas.mplsr7 <- Misclas.of.phon.over.50.samples.with.mplsr7
Misclas.mplsr8 <- Misclas.of.phon.over.50.samples.with.mplsr8
Misclas.mplsr9 <- Misclas.of.phon.over.50.samples.with.mplsr9
Misclas.pca4 <- Misclas.of.phon.over.50.samples.with.pca4
Misclas.pca5 <- Misclas.of.phon.over.50.samples.with.pca5
Misclas.pca6 <- Misclas.of.phon.over.50.samples.with.pca6
Misclas.pca7 <- Misclas.of.phon.over.50.samples.with.pca7
Misclas.pca8 <- Misclas.of.phon.over.50.samples.with.pca8
Misclas.deriv0 <- Misclas.of.phon.over.50.samples.with.deriv0
```

Now, `Misclas.mplsr5`, `Misclas.mplsr6`,...,`Misclas.deriv0` are vectors of length 50. The following commandlines allow to obtain Figure 7.2:

```
Misclas.names <- c("plsr5","plsr6","plsr7","plsr8","plsr9","pca4",
  "pca5","pca6","pca7","pca8","deriv0")
boxplot(Misclas.mplsr5, Misclas.mplsr6,Misclas.mplsr7,Misclas.mplsr8,
  Misclas.mplsr9,Misclas.pca4,Misclas.pca5,Misclas.pca6,Misclas.pca7,
  Misclas.pca8,Misclas.deriv0, names=Misclas.names, xlab="SEMI-METRICS",
  ylab="MISCLSSIFICATION RATES", cex=0.8)
```

It is clear that the semi-metric based on the multivariate PLS regression allow to obtain a good discrimination (see Figure 7.2).

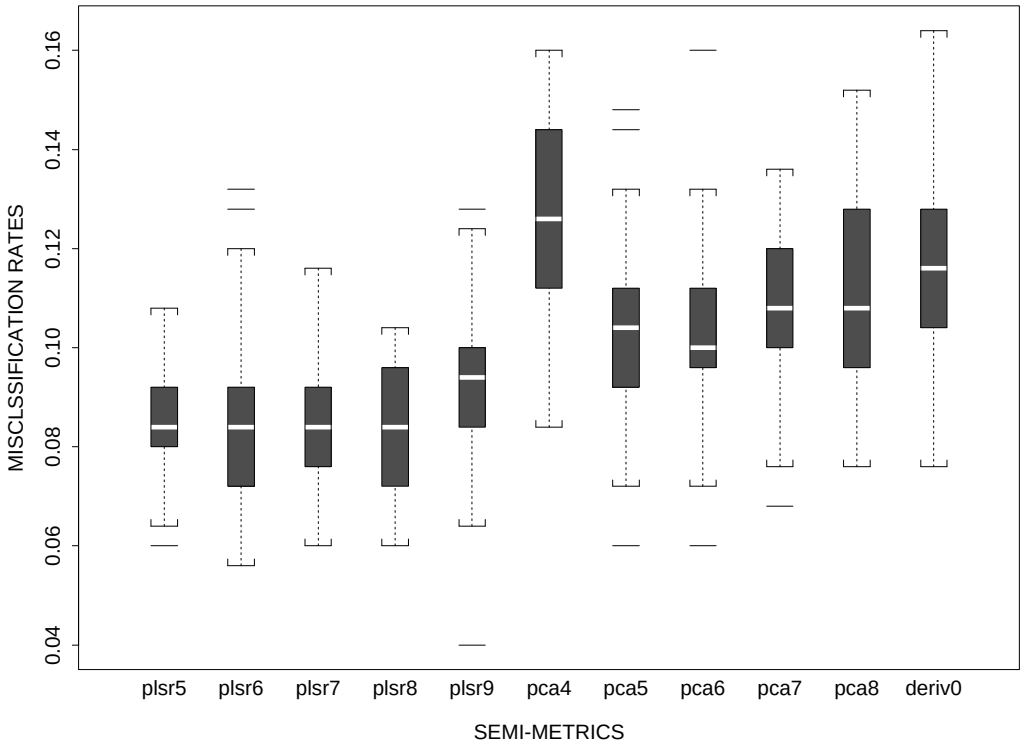


Fig. 7.2. Results over 50 runs

7.2 Discriminating spectrometric curves in relation with fat content

- **Statistical aim**

  We recall that for each unit i (among 215 pieces of finely chopped meat), we observe one spectrometric curve $(\boldsymbol{x}_i)$ which corresponds to the absorbance measured at 100 wavelengths (i.e. $\boldsymbol{x}_i = (\chi_i(\lambda_1), \ldots, \chi_i(\lambda_{100}))$. Moreover, for each unit i, we have at hand its fat content y_i obtained by an analytical chemical processing. The file "npfda-spectrometric.dat" contains the pairs $(\boldsymbol{x}_i, y_i)_{i=1,\ldots,215}$. But, in a discrimination setting, we have to consider a categorical response instead of a scalar one. Therefore, the observed responses $y_1, \ldots, y_{215}$ (column 101) are replaced with $y_1^*, \ldots, y_{215}^*$ where

  $$\forall i = 1, \ldots, 215, \quad y_i^* = \begin{cases} 1 \text{ if } y_i < 20 \\ 2 \text{ else.} \end{cases}$$

  The curves in the group labelled "1" (resp. "2") correspond to a fat content smaller (larger) than 20 %. Given a new sctrometric curve $\boldsymbol{x}$, our main task is to predict the corresponding class membership.

- **Measuring performance**

  For measuring the performance of the functional discrimination procedure, we follow the same methodology than the one used in the speech recognition problem. More precisely, we built 50 learning and testing samples (the ratio between groups being preserved) which allow us to get 50 misclassification rates. The smooth shape of the curves allows to use semimetrics based on the derivatives. We give here directly the results with the semi-metric d_2^{deriv}.

Remark: the commandlines for R or S+ are the same.

- **Entering spectrometric data**

```
SPECDAT <- as.matrix(read.table("npfda-spectrometric.dat"))
attributes(SPECDAT)$dimnames[[1]]_character(0)
SPECURVES <- SPECDAT[,1:100]           # sample of curves
Dichotomous <- SPECDAT[,101]           # sample of scalar responses
Dichotomous[Dichotomous < 20] <- 1     # Class 1: fat content < 20%
Dichotomous[Dichotomous >= 20] <- 2    # Class 2: fat content >= 20%
```

- **Computing predicted class number over 50 samples**

```
Misclas.lcv.of.spec.over.50.samples <- 0
subject1 <- (1:215)[Dichotomous==1]
subject2 <- (1:215)[-subject1]
for(i in 1:50){
```

```
        set.seed(sample(0:1000,1))
        learn1 <- sample(subject1,77)
        learn2 <- sample(subject2,43)
        learning <- c(learn1,learn2)
        testing <- (1:215)[-learning]
        SPECURVESL <- SPECURVES[learning,]
        SPECURVEST <- SPECURVES[testing,]
        Specdichl <- Dichotomous[learning]
        Specdicht <- Dichotomous[testing]
        res <- funopadi.knn.lcv(Specdichl,SPECURVESL,SPECURVEST,
          2,20,c(0,1),kind.of.kernel = "quadratic",semimetric="deriv")
        Misclas.lcv.of.spec.over.50.samples[i] <-
          sum(res$Predicted.classnumber != Specdicht)/length(Specdicht)
}
```

- **Plotting misclassification rates over 50 runs**

  The following commandline allows to obtain Figure 7.3 which summarizes the results. The semi-metric d_2^{deriv} leads clearly to good discrimination (the mean of the misclassification rates equals to 2 %).

```
boxplot(Misclas.lcv.of.spec.over.50.samples, ylab="Misclassification rates",
cex=1)
```

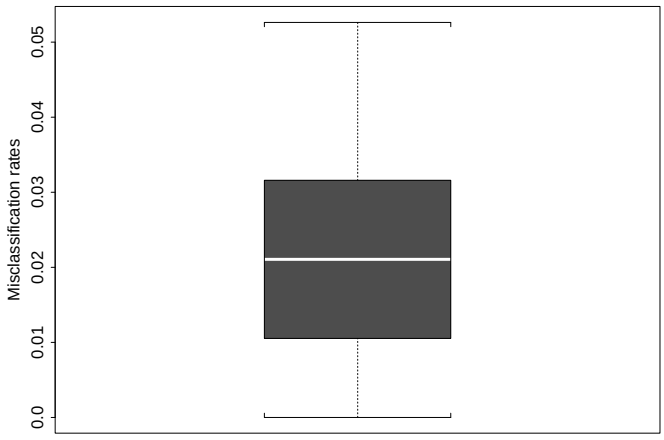


Fig. 7.3. Results over 50 runs

8

Unsupervised classification of curves or clustering (see Chapter 9 of the NPFDA book)

8.1 Classifying radar waveforms from the satellite Topex/Poseidon

- **Statistical aim**

  We recall that for each unit i, we observe one waveform (i.e. curve) measured at 70 points (i.e. $\boldsymbol{x}_i = (\chi_i(t_1), \ldots, \chi_i(t_{70}))$. The file "npfda-sat.dat" contains the observed curves $\{\boldsymbol{x}_i\}_{i=1,\ldots,472}$. This case study is not included in the NPFDA book and for more details on this study, see Dabo-Niang, Ferraty and Vieu (2006, Far East J. Theor. Stat., 18, 93-119).
- **R/S+ commandlines**:
  - **Entering spectrometric data**

    ```
    SATDAT <- as.matrix(read.table("npfda-sat.dat"))
    attributes(SATDAT)$dimnames <- NULL
    ```
  - **Automatical classification: the following commandline allows to get directly the groups**:

    ```
    sat.classif <- classif.npfda(SATDAT, 1:ncol(SATDAT),
                    kind.of.kernel="quadratic", semimetric="hshift",
                    threshold=0.05, nb.bw=50, nss=0, mspg=10,
                    centrality="median")
    ```

    Remark: `nss=0` means that the criterion HI is used instead of SHI whereas `centrality="mean"` implies that HI is based on differences between median and modal curves.
  - **Loading the groups**

    ```
    Group11 <- sat.classif$Partition[[1]]
    Group12 <- sat.classif$Partition[[2]]
    Group21 <- sat.classif$Partition[[3]]
    Group22 <- sat.classif$Partition[[4]]
    Group23 <- sat.classif$Partition[[5]]
    ```

    Remark: `spec.classif$Labels` contains the history of the splitting procedure namely '11', '12' '21', '22' and '23'.

- **Plotting terminal leaves: Group11, Group12, Group21, Group22 and Group23**
  - **Displaying a sample of 16 waveforms for each groups**

    The following R/S+ commandlines allow to build Figure 8.1:

```
s11 <- sample(Group11,16)
s12 <- sample(Group12,16)
s21 <- sample(Group21,16)
s22 <- sample(Group22,16)
s23 <- sample(Group23,16)
x <- 1:ncol(SATDAT)
close.screen(all = TRUE)
screen1 <- c(0.14,0.41,0.55,0.88)
screen2 <- c(0.58,0.85,0.55,0.88)
screen3 <- c(0,0.32,0.11,0.44)
screen4 <- c(0.34,0.66,0.11,0.44)
screen5 <- c(0.68,1,0.11,0.44)
split.screen(rbind(screen1, screen2, screen3, screen4, screen5))
screen(1)
par(mar=c(0,0,0,0)+0.2, oma=c(0,0,2,0), xaxt="n", yaxt="n")
plot(x,x,type="n",bty="n")
mtext(c("GROUP 11","GROUP 12"), side=3, outer=T, line=-4,
      at=c(0.28,.72))
##
# GROUP11
#########
nbscreen <- split.screen(c(4,4),1)
screen(nbscreen[1])
plot(x, SATDAT[s11[1],],ylim=range(SATDAT), type="l",
     xlab="",ylab="")
count <- 0
for(i in s11[2:16]){
        count <- count + 1
        screen(nbscreen[count+1])
        plot(x,SATDAT[i,],ylim=range(SATDAT), type="l",
             xlab="",ylab="")
}
mtext(c("GROUP 21","GROUP 22", "GROUP 23"), side=3, outer=T,
      line=-22, at=c(0.16,0.5,0.84))
##
# GROUP12
#########
nbscreen <- split.screen(c(4,4),2)
screen(nbscreen[1])
plot(x, SATDAT[s12[1],],ylim=range(SATDAT), type="l",
     xlab="",ylab="")
```

```
count <- 0
for(i in s12[2:16]){
        count <- count + 1
        screen(nbscreen[count+1])
        plot(x,SATDAT[i,],ylim=range(SATDAT), type="l",
             xlab="",ylab="")
}
##
# GROUP21
#########
nbscreen <- split.screen(c(4,4),3)
screen(nbscreen[1])
plot(x, SATDAT[s21[1],],ylim=range(SATDAT), type="l",
     xlab="",ylab="")
count <- 0
for(i in s21[2:16]){
        count <- count + 1
        screen(nbscreen[count+1])
        plot(x,SATDAT[i,],ylim=range(SATDAT), type="l",
             xlab="",ylab="")
}
##
# GROUP22
#########
nbscreen <- split.screen(c(4,4),4)
screen(nbscreen[1])
plot(x, SATDAT[s22[1],],ylim=range(SATDAT), type="l",
     xlab="",ylab="")
count <- 0
for(i in s22[2:16]){
        count <- count + 1
        screen(nbscreen[count+1])
        plot(x,SATDAT[i,],ylim=range(SATDAT), type="l",
             xlab="",ylab="")
}
##
# GROUP23
#########
nbscreen <- split.screen(c(4,4),5)
screen(nbscreen[1])
plot(x, SATDAT[s23[1],],ylim=range(SATDAT), type="l",
     xlab="",ylab="")
count <- 0
for(i in s23[2:16]){
        count <- count + 1
```

```
            screen(nbscreen[count+1])
            plot(x,SATDAT[i,],ylim=range(SATDAT), type="l",
                 xlab="",ylab="")
    }
```

•• **Loading mean and modal curves for each terminal group**

```
    MODAL.CURVES <- spec.classif$MODES
    MEAN.CURVES <- spec.classif$MEAN
```

•• **Displaying modal curves for each terminal group**

The following commandlines allow to perform Figure 8.2:

```
    par(mfrow=c(1,6), mar=c(1,2,2,0)+.2, oma=c(1,1,0,0), pty="s")
    Labels <- sat.classif$Labels
    for(j in 1:2)
        plot(x, MODAL.CURVES[j,],ylim=range(SATDAT), type="l",
             xlab="",ylab="", main=paste("GROUP",Labels[j],sep=""),
             cex.main=2)
    plot(x, MODAL.CURVES[j,],ylim=range(SATDAT), type="n", xlab="",
         ylab="",bty="n", xaxs=F, yaxs=F)
    for(j in 3:5)
        plot(x, MODAL.CURVES[j,],ylim=range(SATDAT), type="l", xlab="",
             ylab="", main=paste("GROUP",Labels[j],sep=""), cex.main=2 )
```

•• **Displaying mean curves for each terminal group**

The following commandlines allow to perform Figure 8.3:

```
    par(mfrow=c(1,6), mar=c(1,2,2,0)+.2, oma=c(1,1,0,0), pty="s")
    Labels <- sat.classif$Labels
    for(j in 1:2)
        plot(x, MEAN.CURVES[j,],ylim=range(SATDAT), type="l", xlab="",
             ylab="", main=paste("GROUP",Labels[j],sep=""), cex.main=2)
    plot(x, MEAN.CURVES[j,],ylim=range(SATDAT), type="n", xlab="",
         ylab="",bty="n", xaxs=F, yaxs=F)
    for(j in 3:5)
        plot(x, MEAN.CURVES[j,],ylim=range(SATDAT), type="l", xlab="",
             ylab="", main=paste("GROUP",Labels[j],sep=""), cex.main=2 )
```

•• **Computing mean and median curves for Groups 1 and 2**

```
    nb.groups <- length(sat.classif$Partition)
    MEAN.CURVES <- matrix(0, 2, ncol(SATDAT))
    MEDIAN.CURVES <- matrix(0, 2, ncol(SATDAT))
    MODAL.CURVES <- matrix(0, 2, ncol(SATDAT))
    Group1 <- c(sat.classif$Partition[[1]],
                sat.classif$Partition[[2]])
    Group2 <- c(sat.classif$Partition[[3]],
                sat.classif$Partition[[4]],
                sat.classif$Partition[[5]])
    )
    MEAN.CURVES[1,] <- apply(SATDAT[Group1,], 2, mean)
    MEAN.CURVES[2,] <- apply(SATDAT[Group2,], 2, mean)
```

```
MEDIAN.CURVES[1,] <- SATDAT[Group1[median.npfda(SEMIMETRIC
                            [Group1, Group1])],]
MEDIAN.CURVES[2,] <- SATDAT[Group2[median.npfda(SEMIMETRIC
                            [Group2, Group2])],]
```

•• **Computing modal curves for groups 1 and 2**

```
grid <- 1:ncol(SATDAT)
SM.SAT <- semimetric.hshift(SATDAT, SATDAT, grid)
Hrange <- range(SM.SAT)
Bw.seq <- seq(Hrange[1], Hrange[2] * 0.5, length = 50 + 1)[-1]
PROBCURVES <- matrix(0, n, 50)
for(i in 1:n){
        PROBCURVES[i,] <- prob.curve(i, SM.SAT, Bw.seq)
}
res.classif.bw <- classif.bw(PROBCURVES, Bw.seq, 1:n)
Bw.opt <- res.classif.bw$bw
index <- res.classif.bw$index
shi <- classif.hi(SATDAT, SM.SAT, "quadratic", Bw.opt, 1:n,
                  "hshift", "median", grid)
first.split.sat <- classif.part(SATDAT, PROBCURVES, SM.SAT,
                                "quadratic", index, Bw.seq, 1:n,
                                shi, "hshift", 0.05, 0, 10,
                                "median", grid)
Group1 <- first.split.sat$Groups[[1]]
Group2 <- first.split.sat$Groups[[2]]
Band <- first.split.sat$Bw.opt
rank.mode1 <- funopa.mode(Band[1], SM.SAT[Group1,Group1],
                          "quadratic")
MODAL.CURVES[1,] <- CURVES[Group1[rank.mode1],]
rank.mode2 <- funopa.mode(Band[2], SM.SAT[Group2, Group2],
                          "quadratic")
MODAL.CURVES[2,] <- CURVES[Group2[rank.mode2],]
```

•• **Displaying modal, median and mean curves for groups 1 and 2**

The following commandlines allow to perform Figure 8.4:

```
par(mfrow=c(2,3), mar=c(1,2,2,0)+.2, oma=c(1,1,0,0), pty="s")
for(j in 1:2){
    plot(x, MODAL.CURVES[j,],ylim=range(SATDAT), type="l",
         xlab="",ylab="", main=paste("Modal curve of GROUP",j,sep=""),
         cex.main=1.5)
    plot(x, MEDIAN.CURVES[j,],ylim=range(SATDAT), type="l", xlab="",
         ylab="", main=paste("Median curve of GROUP",j,sep=""),
         cex.main=1.5)
    plot(x, MEAN.CURVES[j,],ylim=range(SATDAT), type="l", xlab="",
         ylab="", main=paste("Mean curve of GROUP",j,sep=""),
         cex.main=1.5)
```

```
}
```

•• **Displaying the splitting score behavior along the procedure**
The following commandlines allow to perform Figure 8.5:

```
Splitting.score <- sat.classif$Ssc
Split.names <- c(paste("GROUPS","\n","1 & 2"),
                 paste("GROUPS", "\n","11 & 12"),
                 paste("GROUPS", "\n","21, 22 & 23"))
par(mfrow=c(1,1))
barplot(Splitting.score, names=Split.names,
        main="Splitting scores", ylab="Percentages")
abline(h=0.05, lwd=3)
```

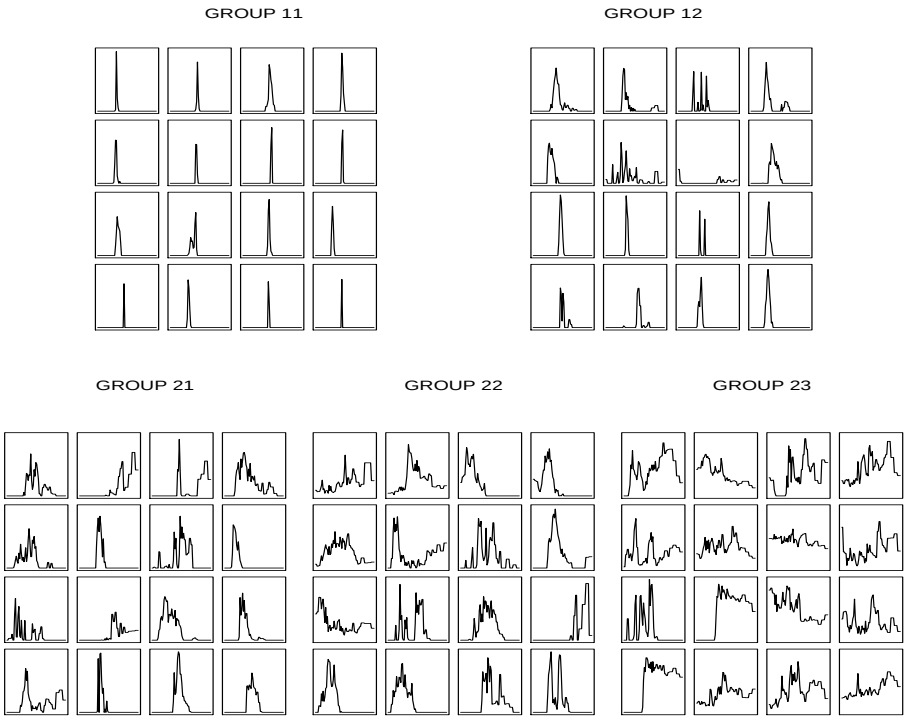


Fig. 8.1. Samples of radar waveforms for each group

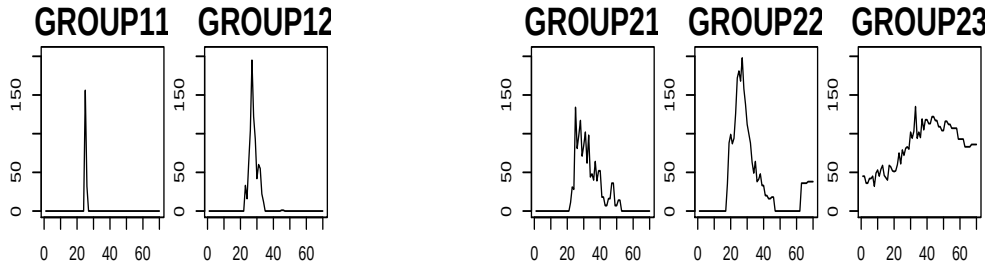


Fig. 8.2. Modal curves for the five terminal groups

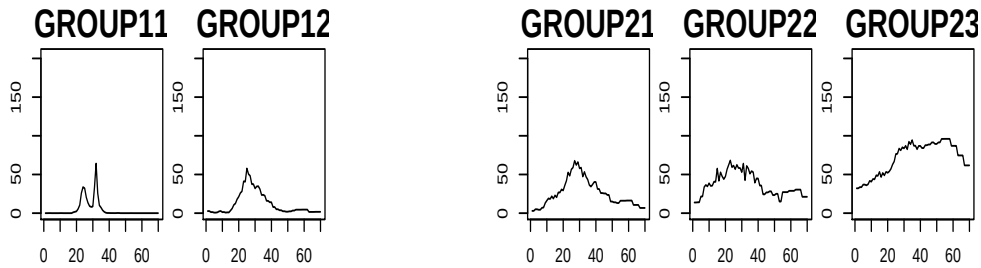


Fig. 8.3. Mean curves for the five terminal groups

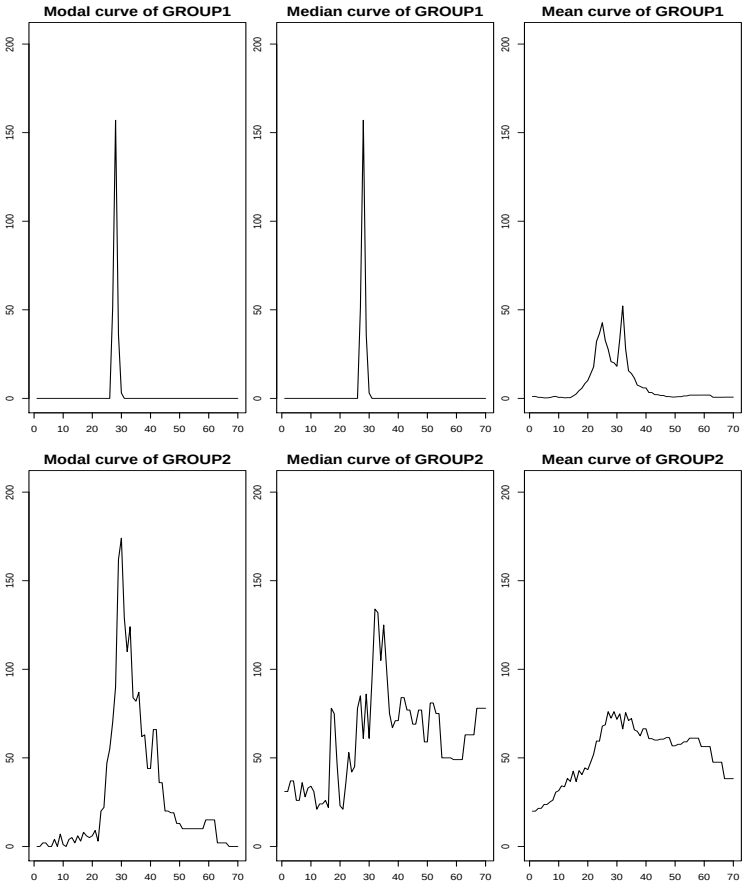


Fig. 8.4. First splitting into $GROUP\ 1$ and $GROUP\ 2$: comparison between modal, median and mean curves

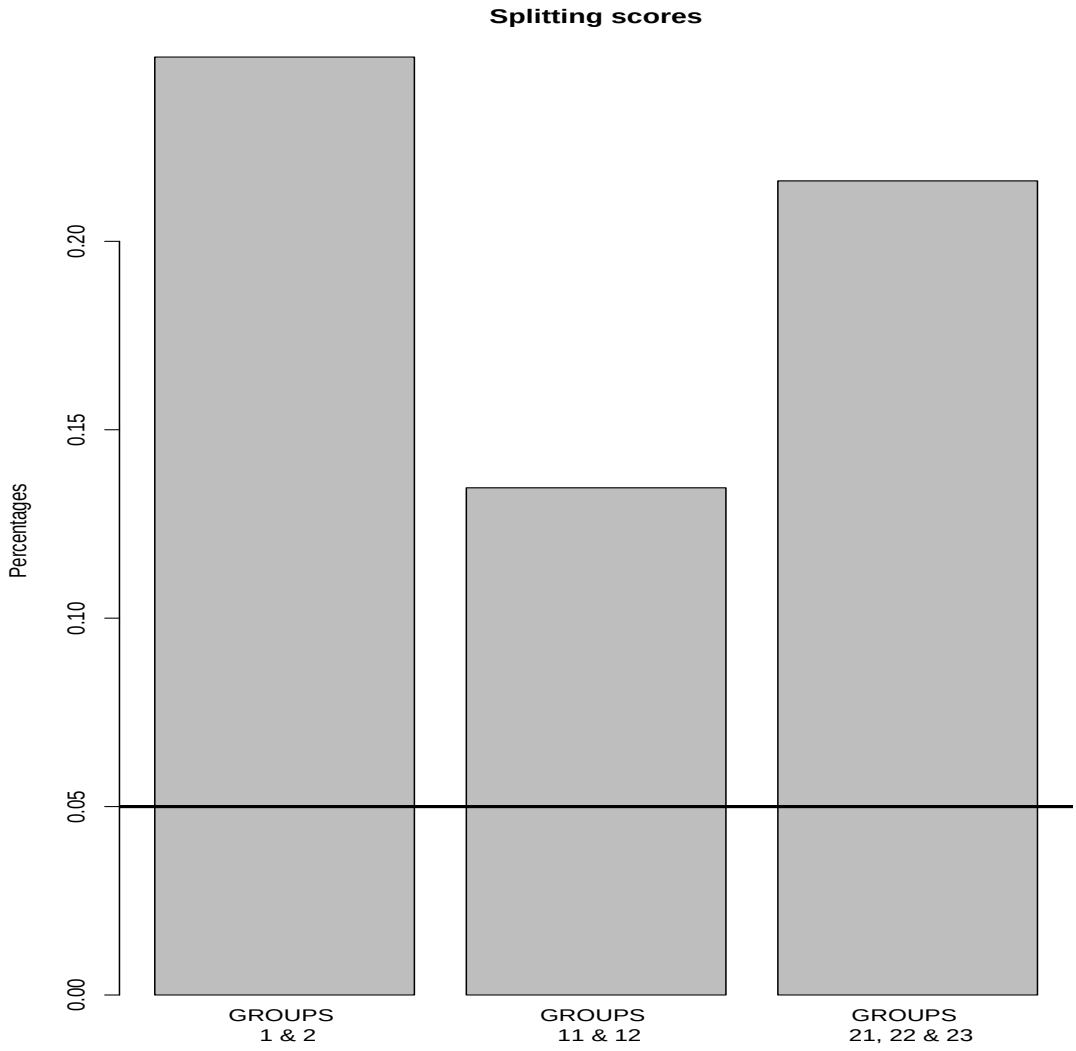


Fig. 8.5. Behavior of splitting score for the satellite waveforms; the horizontal line corresponds to the threshold of splitting score

8.2 Classifying spectrometric curves

- **Statistical aim**

  We recall that for each unit i (among 215 pieces of finely chopped meat), we observe one spectrometric curve ($\boldsymbol{x}_i$) which corresponds to the absorbance measured at 100 wavelengths (i.e. $\boldsymbol{x}_i = (\chi_i(\lambda_1), \ldots, \chi_i(\lambda_{100}))$. The first 100 columns of the file "npfda-spectrometric.dat" contain the observed curves $\{\boldsymbol{x}_i\}_{i=1,\ldots,215}$. It is worth noting that the reponse (fat content) is completely ignored for this classification analysis.
- **R/S+ commandlines**:
  - **Entering spectrometric data**
    ```
    SPECDAT <- as.matrix(read.table("npfda-spectrometric.dat"))
    attributes(SPECDAT)$dimnames[[1]] <- character(0)
    SPECURVES <- SPECDAT[,1:100]           # sample of curves
    ```
  - **Automatical classification: the following commandline allows to get directly the groups**:
    ```
    spec.classif <- classif.npfda(SPECURVES, 2, 20, c(0,1),
                    semimetric="deriv", threshold=0.05, nb.bw=100,
                    nss=0, mspg=10, centrality="mean")
    ```
    Remark: `nss=0` means that the criterion HI is used instead of SHI whereas `centrality="mean"` implies that HI is based on differences between mean and modal curves.
  - **Loading the groups**
    ```
    Group1 <- spec.classif$Partition[[1]]
    Group21 <- spec.classif$Partition[[2]]
    Group22 <- spec.classif$Partition[[3]]
    ```
    Remark: `spec.classif$Labels` contains the history of the splitting procedure namely '1', '21' and '22'.

- **Plotting terminal leaves: Group1, Group21 and Group22**
  - **Computing the second derivatives of the spectrometric curves**

```
SPECD2 <- t(approx.spline.deriv(SPECURVES, 2, 20,
            c(0,1))$APPROX)
```

  - **Loading the modal curve for each group**

```
MODAL.CURVES <- spec.classif$MODES
```

  - **Computing the second derivatives of the modal curves**

```
MODAL.CURVESD2 <- t(approx.spline.deriv(
                    MODAL.CURVES, 2, 20, c(0,1))$APPROX)
```

  - **Displaying groups: original data, second derivatives and corresponding modal curves**

The following R/S+ commandlines allow to build Figure 8.6:

```
x <- seq(850, 1050, length=100)
##
# GROUP1
########
par(mfrow=c(3,3))
plot(x,SPECURVES[Group1[1],],ylim=range(SPECURVES), type="l",
     xlab="Wavelenghts",ylab="Absorbances", main="GROUP 1")
for(i in Group1[-1]){
        par(new=T)
        plot(x,SPECURVES[i,],ylim=range(SPECURVES), type="l",
             xlab="", ylab="", axes=F)
}
plot(x,SPECD2[Group1[1],],ylim=range(SPECD2), type="l",
     xlab="Wavelenghts", ylab="Absorbances",
     main="GROUP 1: second derivatives")
for(i in Group1[-1]){
        par(new=T)
        plot(x,SPECD2[i,],ylim=range(SPECD2), type="l",
             xlab="",ylab="", axes=F)
}
plot(x, MODAL.CURVESD2[1,], ylim=range(SPECD2), type="l",
     xlab="Wavelenghts", ylab="Absorbances",
     main="GROUP 1: modal curve")
##
# GROUP21
#########
plot(x,SPECURVES[Group21[1],],ylim=range(SPECURVES), type="l",
     xlab="Wavelenghts",ylab="Absorbances", main="GROUP 21")
for(i in Group21[-1]){
        par(new=T)
        plot(x,SPECURVES[i,],ylim=range(SPECURVES), type="l",
             xlab="", ylab="", axes=F)
```

```
}
plot(x,SPECD2[Group21[1],],ylim=range(SPECD2), type="l",
     xlab="Wavelenghts", ylab="Absorbances",
     main="GROUP 21: second derivatives")
for(i in Group21[-1]){
        par(new=T)
        plot(x,SPECD2[i,],ylim=range(SPECD2), type="l", xlab="",
             ylab="", axes=F)
}
plot(x, MODAL.CURVESD2[2,], ylim=range(SPECD2), type="l",
     xlab="Wavelenghts", ylab="Absorbances",
     main="GROUP 21: modal curve")
##
# GROUP22
#########
plot(x,SPECURVES[Group22[1],],ylim=range(SPECURVES), type="l",
     xlab="Wavelenghts",ylab="Absorbances", main="GROUP 22")
for(i in Group22[-1]){
        par(new=T)
        plot(x,SPECURVES[i,],ylim=range(SPECURVES), type="l",
             xlab="", ylab="", axes=F)
}
plot(x,SPECD2[Group22[1],],ylim=range(SPECD2), type="l",
     xlab="Wavelenghts", ylab="Absorbances",
     main="GROUP 22: second derivatives")
for(i in Group22[-1]){
        par(new=T)
        plot(x,SPECD2[i,],ylim=range(SPECD2), type="l", xlab="",
             ylab="", axes=F)
}
plot(x, MODAL.CURVESD2[3,], ylim=range(SPECD2), type="l",
     xlab="Wavelenghts", ylab="Absorbances",
     main="GROUP 22: modal curve")
```

•• Displaying the splitting score behavior along the procedure

The following commandlines allow to perform Figure 8.7:

```
Splitting.score <- spec.classif$Ssc
Split.names <- c(paste("GROUPS","\n","1 & 2"),
                 paste("GROUPS", "\n","21 & 22"))
par(mfrow=c(1,1))
barplot(Splitting.score, names=Split.names, main="Splitting scores",
        ylab="Percentages")
abline(h=0.05, lwd=3)
```

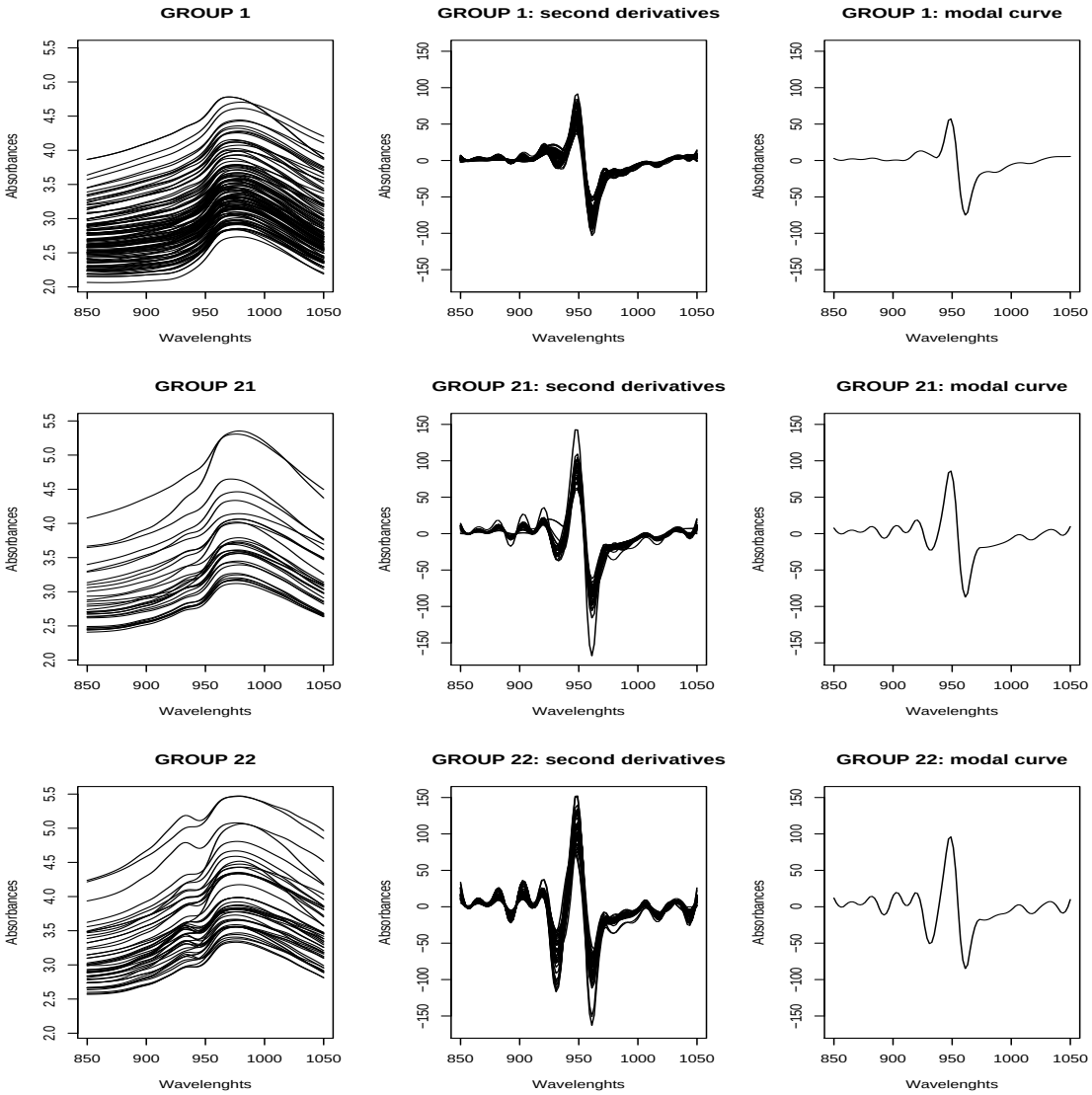


Fig. 8.6. For each group: left column displays the original spectrometric curves, middle column plots their second derivatives and right column displays the corresponding functional mode

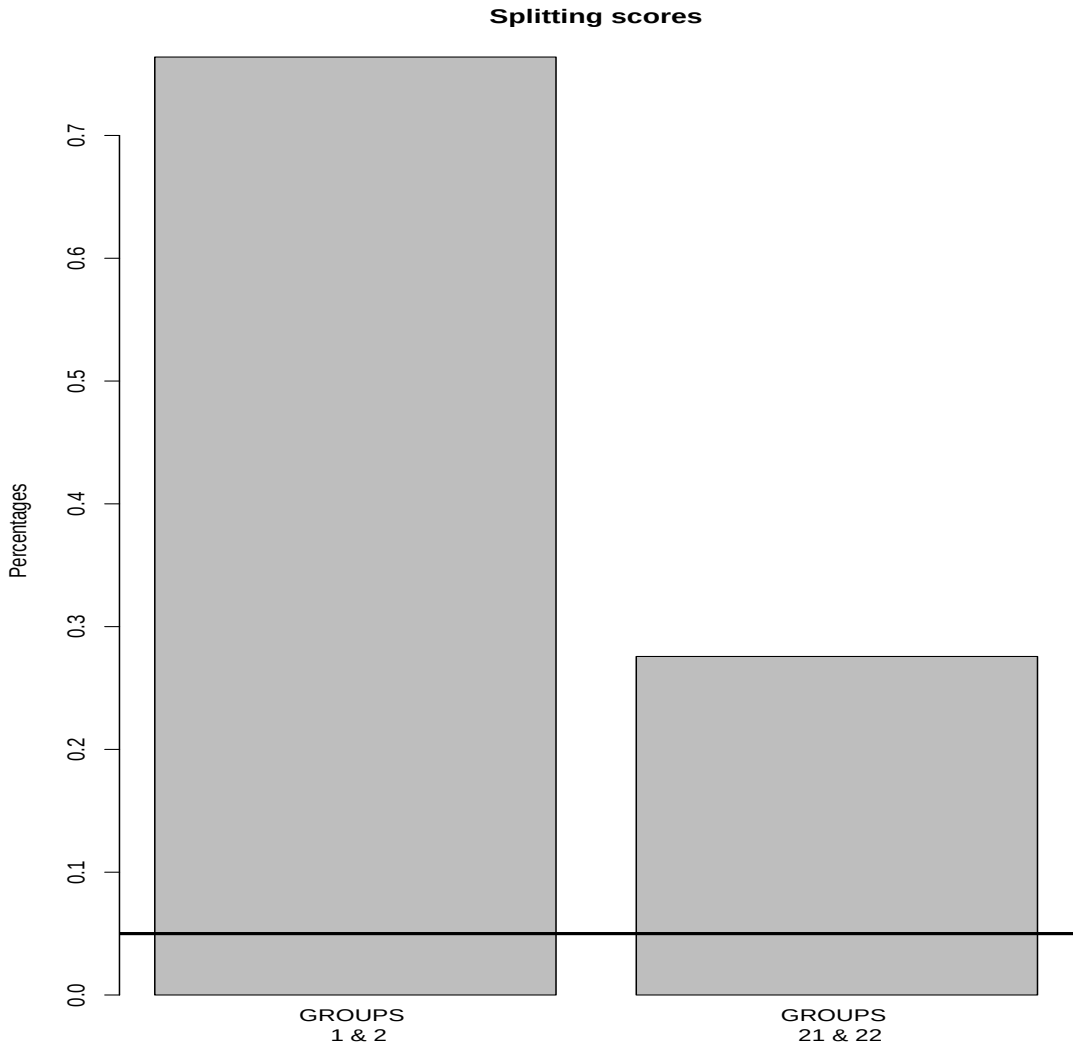


Fig. 8.7. Behavior of splitting score for the spectrometric data; the horizontal line corresponds to the threshold of splitting score

9

Forecasting functional time series (see NPFDA's Part IV)

9.1 Forecasting El Niño time series in a functional way

- **Statistical aim**

We focus on the El Niño time series which gives monthly Sea Surface temperatures. This case study is not included in the NPFDA book and for more details, see Ferraty, Rabhi and Vieu (2005, Sankhya, 67, 378-398). The data are recorded as a sequence of real numbers. Here, the dataset is componed of $N = 648$ real values $\{z_i, i = 1, \ldots, 648\}$ and organized as follows (see description of the datasets):

	Col 1	$\cdots$	Col j	$\cdots$	Col 12
Row 1	z_1	$\cdots$	z_j	$\cdots$	z_{12}
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$
Row i	$z_{1+12(i-1)}$	$\cdots$	$z_{j+12(i-1)}$	$\cdots$	z_{12i}
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$
Row 54	z_{633}	$\cdots$	z_{632+j}	$\cdots$	z_{648}

Firstly, one has to decide which past values have to be taken into account for prediction. In order to apply the functional methodology, one has to cut the original time series in a set of functional data. Here we have decided to predict future sea surface temperatures by using the El Niño data for the whole last year. That means that, with notations of our book, we have choosen $\tau = 12$. In order to illustrate our purpose, we will not use the 54^{th} year and we will predict it by mean of the data corresponding of the 53 previous years. To use the nonparametric functional methods, one has first to decide what is the horizon s of prediction that is wished. Then, for fixed s, the data will be reorganized into a functional explanatory sample $\{\chi_i, i = 1, \ldots, 52\}$ which will be loaded in the following 52×12 matrix:

z_1	$\cdots$	z_j	$\cdots$	z_{12}
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$
$z_{1+12(i-1)}$	$\cdots$	$z_{j+12(i-1)}$	$\cdots$	z_{12i}
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$
z_{613}	$\cdots$	z_{612+j}	$\cdots$	z_{624}

and a response real sample $\{y_i, i = 1, \ldots, 52\}$, which will be loaded in the following 52-dimensional vector:

z_{12+s}	$\cdots$	z_{12i+s}	$\cdots$	z_{612+s}

For fixed horizon s, we can predict the value $\widehat{z}_{624+s}$ by using any technique among the three ones which are described in the book. Our goal is not to make a full analysis of this dataset, and to make things clearer we will just present the results obtained with $R/S+$ routines involving automatic smoothing parameter choices. More precisely, each among the three $R/S+$ routines `funopare.knn.lcv`, `funopare.mode.lcv` and `funopare.quantile.lcv` have been used to compute the predicted value $\widehat{z}_{624+s}$ obtained respectively by the regression operator estimation technique, by the conditional mode estimation technique and by the conditional median estimation technique. Concerning the semi-metric involved in the nonparametric forecasting procedures, the small number of discretization points for each curve (exactly 12) suggested the use of a semi-metric based on functional principal components ideas. Precisely, we used the PCA semi-metric $\boldsymbol{d}_q^{PCA}$ defined in the book and we took the parameter q which allows to get the best mean squares error ($q = 4$ for `funopare.knn.lcv`, $q = 3$ for `funopare.mode.lcv` and $q = 2$ for `funopare.quantile.lcv`).

- **Entering and organizing El Niño dataset**

```
ELNINODAT <- as.matrix(read.table("npfda-elnino.dat"))
attributes(ELNINODAT)$dimnames[[1]] <- character(0)
learning <- 1:52
testing <- 53
elnino.past.learn <- ELNINODAT[learning,]  # sample of explanatory curves
elnino.past.testing <- ELNINODAT[testing,] # The 53th year
s <- 1                                     # forecasting horizon 1
elnino.futur.s <- ELNINODAT[2:53,s]        # sample of real responses
```

Now, the R/S+ routines for prediction of a scalar response from a functional sample can be easily used in the following way.

- **The functional nonparametric forecasting**

```
result.pred.reg.step.s <- funopare.knn.lcv(elnino.futur.s,
  elnino.past.learn,elnino.past.testing,4,
```

```
  kind.of.kernel="quadratic",semimetric="pca")
              # Kernel functional regression forecasting

result.pred.median.step.s <- funopare.quantile.lcv(
  elnino.futur.s,elnino.past.learn,elnino.past.testing,2,
  alpha=0.5, Knearest=NULL, kind.of.kernel="quadratic",
  semimetric="pca")
              # Kernel functional median forecasting

result.pred.mode.step.s <- funopare.mode.lcv(
  elnino.futur.s,elnino.past.learn,elnino.past.testing,3,
  Knearest=NULL, kind.of.kernel="quadratic",
  semimetric="pca")
               # Kernel functional mode forecasting

result.pred.quantiles.s <- funopare.quantile.lcv(
  elnino.futur.s,elnino.past.learn,elnino.past.testing,
  2,alpha=c(0.05,0.5,0.95), Knearest=NULL,
  kind.of.kernel="quadratic",semimetric="pca")
               # Median estimation and 90% prediction band
```

- **Collecting the forecasting results**

  These R/S+ routines are recording several different results. The most important ones are the predicted responses which can be obtained in the following manner.

```
result.pred.reg.step.s$Predicted.values
              # Forecasted value with regression method

result.pred.median.step.s$Predicted.values
              # Forecasted value with median method

result.pred.mode.step.s$Predicted.values
              # Forecasted value with mode method

result.pred.quantiles.s$Predicted.values
              #.05, .5 and .95 estimated quantiles
```

- **Forecasting the 54th year**

  To do that, it suffices to repeat the previous stages for $s = 1, \ldots, 12$ (horizons):

```
pred.reg <- 0
```

```
pred.median <- 0
pred.mode <- 0
for(s in 1:12){
        elnino.futur.s <- ELNINODAT[2:53,s]
        result.pred.step.s <- funopare.knn.lcv(elnino.futur.s,
          elnino.past.learn,elnino.past.testing,4,
          kind.of.kernel="quadratic",semimetric="pca")
        pred.reg[s] <- result.pred.step.s$Predicted.values
        result.pred.median.step.s <- funopare.quantile.lcv(elnino.futur.s,
          elnino.past.learn,elnino.past.testing,2,alpha=0.5,
          Knearest=NULL, kind.of.kernel="quadratic", semimetric="pca")
        pred.median[s] <- result.pred.median.step.s$Predicted.values
        result.pred.mode.step.s <- funopare.mode.lcv(elnino.futur.s,
          elnino.past.learn,elnino.past.testing,3,Knearest=NULL,
          kind.of.kernel="quadratic",semimetric="pca")
        pred.mode[s] <- result.pred.mode.step.s$Predicted.values
}
```

- **Plotting the forecasted values**

  The following commandlines allow to display the forecasted 54th year obtained (Figure 9.1) by the various functional prediction methods and we compare them with the observed values (54th year):

```
year54 <- ELNINODAT[54,]
mse.reg <- round(sum((pred.reg-year54)^2)/12,4)
mse.median <- round(sum((pred.median-year54)^2)/12,4)
mse.mode <- round(sum((pred.mode-year54)^2)/12,4)
par(mfrow=c(2,2))
plot(1:12,pred.reg,xlab='54th year', ylab='',
  main=paste('Regression: MSE=',mse.reg,sep=''),
  type='l',lty=2,ylim=range(c(pred.reg,year54)))
par(new=T)
plot(1:12,year54,type='l',lty=1,axes=F,xlab='', ylab='')
plot(1:12,pred.median,xlab='54th year', ylab='',
  main=paste('Median: MSE=',mse.median,sep=''),
  type='l',lty=2,ylim=range(c(pred.median,year54)))
par(new=T)
plot(1:12,year54,type='l',lty=1,axes=F,xlab='', ylab='',
  ylim=range(c(pred.median,year54)))
plot(1:12,pred.mode,xlab='54th year', ylab='',
  main=paste('Mode: MSE=',mse.mode,sep=''),
  type='l',lty=2,ylim=range(c(pred.mode,year54)))
par(new=T)
plot(1:12,year54,type='l',lty=1,axes=F,xlab='', ylab='',
  ylim=range(c(pred.mode,year54)))
```

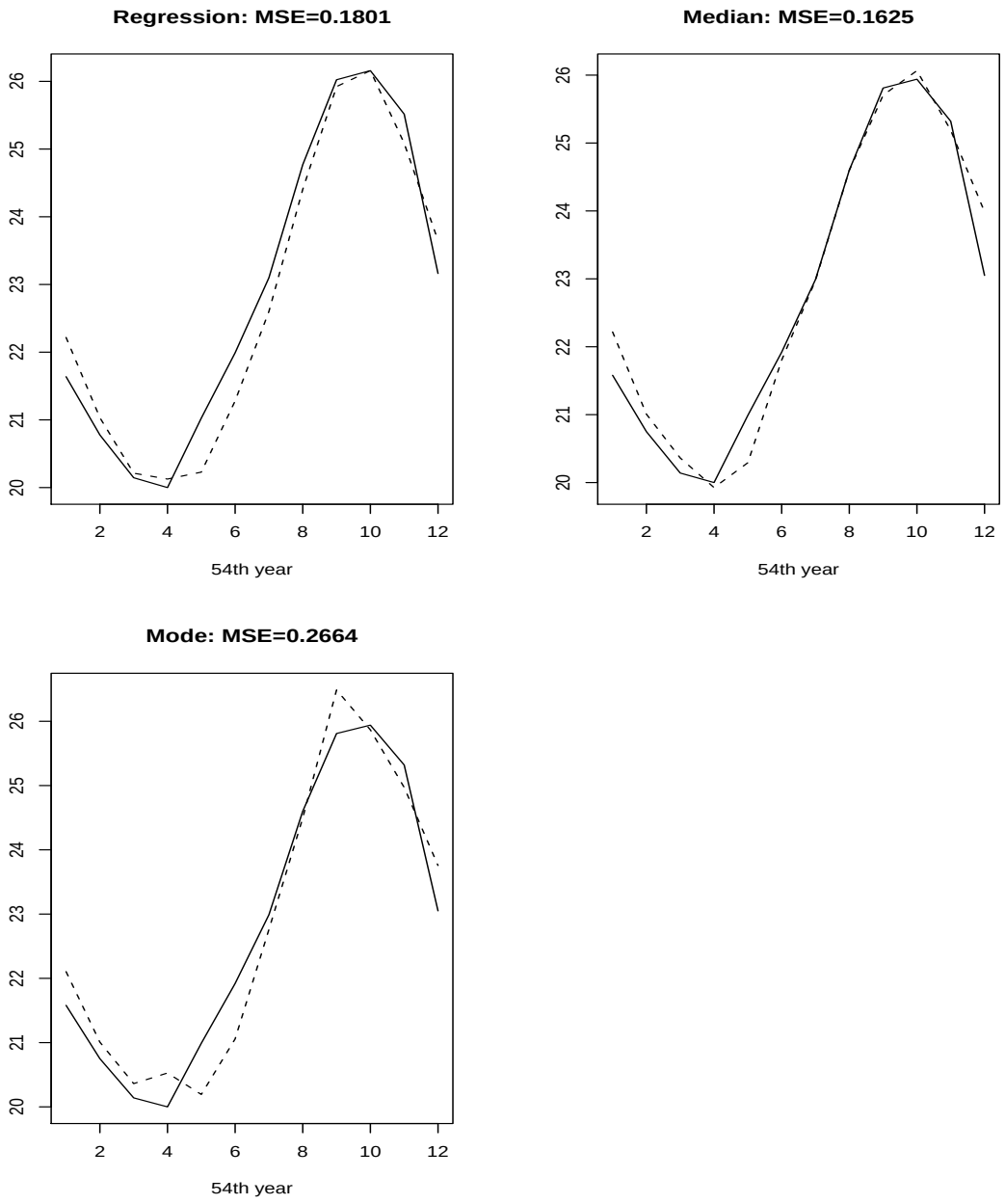


Fig. 9.1. Forecasted 54th year for the three prediction methods

Remark: it is possible to get pointwise confidence prediction band with the routine `funopare.quantile.lcv`.

9.2 Forecasting electricity consumption in a functional way

- **Statistical aim**

We focus on the electricity consumption time series. The data are recorded as a sequence of real numbers. Here, the dataset is componed of $N = 336$ real values $\{z_i, i = 1, \ldots, 336\}$ (not to mask the main of purpose, we will directly work with the differenciated log data) and organized as follows (see description of the datasets):

	Col 1	$\cdots$	Col j	$\cdots$	Col 12
Row 1	z_1	$\cdots$	z_j	$\cdots$	z_{12}
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$
Row i	$z_{1+12(i-1)}$	$\cdots$	$z_{j+12(i-1)}$	$\cdots$	z_{12i}
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$
Row 28	z_{325}	$\cdots$	z_{324+j}	$\cdots$	z_{336}

Firstly, one has to decide which past values have to be taken into account for prediction. In order to apply the functional methodology, one has to cut the original time series in a set of functional data. Here we have decided to predict future electrical consumption by using the consumption data for the whole last year. That means that, with notations of our book, we have choosen $\tau = 12$. In order to illustrate our purpose, we will not use the 28^{th} year and we will predict it by mean of the data corresponding of the 27 previous years. To use the nonparametric functional methods, one has first to decide what is the horizon s of prediction that is wished. Then, for fixed s, the data will be reorganized into a functional explanatory sample $\{\chi_i, i = 1, \ldots, 26\}$ which will be loaded in the following 26×12 matrix:

z_1	$\cdots$	z_j	$\cdots$	z_{12}
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$
$z_{1+12(i-1)}$	$\cdots$	$z_{j+12(i-1)}$	$\cdots$	z_{12i}
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$
z_{301}	$\cdots$	z_{300+j}	$\cdots$	z_{312}

and a response real sample $\{y_i, i = 1, \ldots, 26\}$, which will be loaded in the following 26-dimensional vector:

z_{12+s}	$\cdots$	z_{12i+s}	$\cdots$	z_{312+s}

For fixed horizon s, we can predict the value $\widehat{z}_{324+s}$ by using any technique among the three ones which are described in the book. Our goal is not to make a full analysis of this economic dataset, and to make things clearer we will just present the results obtained with $R/S+$ routines involving automatic smoothing parameter choices. More precisely, each among the three $R/S+$ routines `funopare.knn.lcv`, `funopare.mode.lcv` and `funopare.quantile.lcv` have been used to compute the predicted value $\widehat{z}_{324+s}$ obtained respectively by the regression operator estimation technique, by the conditional mode estimation technique and by the conditional median estimation technique. Concerning the semi-metric involved in the nonparametric forecasting procedures, the small number of discretization points for each curve (exactly 12) suggested the use of a semi-metric based on functional principal components ideas. Precisely, we used the PCA semi-metric $\boldsymbol{d}_q^{PCA}$ defined in the book and we took the parameter q which allows to get the best mean square errors ($q = 5$ for `funopare.knn.lcv`, $q = 2$ for `funopare.mode.lcv` and `funopare.quantile.lcv`).

- **Organizing electrical data**

```
CONSELDAT <- as.matrix(read.table("npfda-electricity.dat"))
attributes(CONSELDAT)$dimnames[[1]] <- character(0)
learning <- 1:26
testing <- 27
elec.past.learn<-CONSELDAT[learning,]  # sample of explanatory curves
elec.past.testing<-CONSELDAT[testing,] # The 27th year
s <- 1                                 # forecasting horizon 1
elec.futur.s <- CONSELDAT[2:27,s]      # sample of real responses
```

Now, the R/S+ routines for prediction of a scalar response from a functional sample can be easily used in the following way.

- **The functional nonparametric forecasting**

```
result.pred.reg.step.s <- funopare.knn.lcv(elec.futur.s,
  elec.past.learn,elec.past.testing,5,
  kind.of.kernel="quadratic",semimetric="pca")
              # Kernel functional regression forecasting

result.pred.median.step.s <- funopare.quantile.lcv(
  elec.futur.s,elec.past.learn,elec.past.testing,2,
  alpha=0.5, Knearest=NULL, kind.of.kernel="quadratic",
  semimetric="pca")
              # Kernel functional median forecasting

result.pred.mode.step.s <- funopare.mode.lcv(
  elec.futur.s,elec.past.learn,elec.past.testing,2,
  Knearest=NULL, kind.of.kernel="quadratic",
```

```
  semimetric="pca")
              # Kernel functional mode forecasting

result.pred.quantiles.s<-funopare.quantile.lcv(
  elec.futur.s,elec.past.learn,elec.past.testing,
  2,alpha=c(0.05,0.5,0.95), Knearest=NULL,
  kind.of.kernel="quadratic",semimetric="pca")
              # Median estimation and 90% prediction band
```

- **Collecting the forecasting results**

  These R/S+ routines are recording several different results. The most important ones are the predicted responses which can be obtained in the following manner.

```
result.pred.reg.step.s$Predicted.values
              # Forecasted value with regression method

result.pred.median.step.s$Predicted.values
              # Forecasted value with median method

result.pred.mode.step.s$Predicted.values
              # Forecasted value with mode method

result.pred.quantiles.s$Predicted.values
              #.05, .5 and .95 estimated quantiles
```

- **Forecasting the 28th year**

  To do that, it suffices to repeat the previous stages for $s = 1, \ldots, 12$ (horizons):

```
pred.reg <- 0
pred.median <- 0
pred.mode <- 0
for(s in 1:12){
        elec.futur.s <- CONSELDAT[2:27,s]
        result.pred.step.s <- funopare.knn.lcv(elec.futur.s,
          elec.past.learn,elec.past.testing,5,
          kind.of.kernel="quadratic",semimetric="pca")
        pred.reg[s] <- result.pred.step.s$Predicted.values
        result.pred.median.step.s <- funopare.quantile.lcv(elec.futur.s,
          elec.past.learn,elec.past.testing,2,alpha=0.5,
          Knearest=NULL, kind.of.kernel="quadratic", semimetric="pca")
        pred.median[s] <- result.pred.median.step.s$Predicted.values
```

```
        result.pred.mode.step.s <- funopare.mode.lcv(elec.futur.s,
          elec.past.learn,elec.past.testing,2,Knearest=NULL,
          kind.of.kernel="quadratic",semimetric="pca")
        pred.mode[s] <- result.pred.mode.step.s$Predicted.values
}
```

- **Plotting the forecasted values**

  The following commandlines allow to display the forecasted 28th year obtained (figure 9.2) by the various functional prediction methods and we compare them with the observed values (28th year):

```
year28 <- CONSELDAT[28,]
mse.reg <- round(sum((pred.reg-year28)^2)/12,4)
mse.median <- round(sum((pred.median-year28)^2)/12,4)
mse.mode <- round(sum((pred.mode-year28)^2)/12,4)
par(mfrow=c(2,2))
plot(1:12,pred.reg,xlab='28th year', ylab='',
  main=paste('Regression: MSE=',mse.reg,sep=''),
  type='l',lty=2,ylim=range(c(pred.reg,year28)))
par(new=T)
plot(1:12,year28,type='l',lty=1,axes=F,xlab='', ylab='')
plot(1:12,pred.median,xlab='28th year', ylab='',
  main=paste('Median: MSE=',mse.median,sep=''),
  type='l',lty=2,ylim=range(c(pred.median,year28)))
par(new=T)
plot(1:12,year28,type='l',lty=1,axes=F,xlab='', ylab='',
  ylim=range(c(pred.median,year28)))
plot(1:12,pred.mode,xlab='28th year', ylab='',
  main=paste('Mode: MSE=',mse.mode,sep=''),
  type='l',lty=2,ylim=range(c(pred.mode,year28)))
par(new=T)
plot(1:12,year28,type='l',lty=1,axes=F,xlab='', ylab='',
  ylim=range(c(pred.mode,year28)))
```

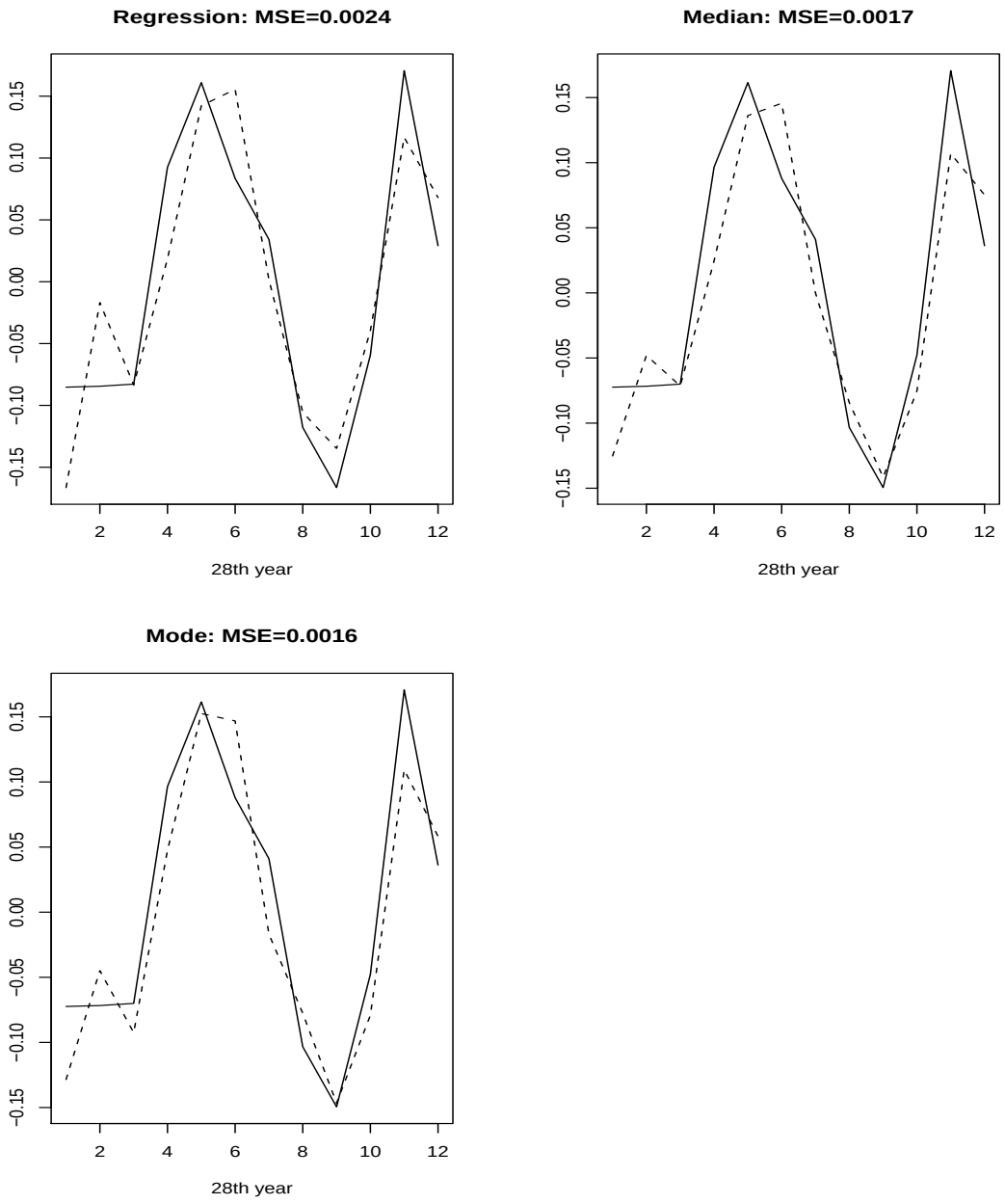


Fig. 9.2. Forecasted 28th year for the three prediction methods

Part III

R/S-plus Routines Help Files

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.

10

Indexes of R/S-plus routines

10.1 Alphabetical Index for all routines

10.2 Index by Category

10.2.1 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

10.2.2 discriminating a sample of curves (supervised classification)

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

10.2.3 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

10.2.4 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

10.2.5 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

10.2.6 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

11

Help Files for main routines

11.1 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. $\{\chi_i\}_{i=1,\ldots,n}$ at the J points $\{t_j\}_{j=1,\ldots,J}$ 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 $\boldsymbol{x}_i = (\chi_i(t_1), \ldots, \chi_i(t_J))$. Therefore, in the following help files, we suppose that we have at hand matrices `DATA1` (or `CURVES`) and `DATA2` (or `PRED`) such that:

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

and,

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

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).

11.2 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]` $= R_{LCV}^{kNN}(\boldsymbol{x}_i)$.

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

$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 $(y_1, \ldots, y_n)$.

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

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,]`$= t^{kNN}_{alpha[1]}(\boldsymbol{x}_i), t^{kNN}_{alpha[2]}(\boldsymbol{x}_i), \ldots$

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

$Response.values vector of length card(I_2) such that, for all `i` in I_2, `Response.values[i]`$= 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