7

Computational Issues

This chapter is devoted to the implementation of the functional nonparametric prediction methods based on regression, conditional quantiles and conditional mode with special attention to the regression ones. It concerns mainly users/practitioners wishing to test such functional statistical techniques on their datasets. The main goal consists in presenting several routines written in S+ or R in order to make any user familiar with such statistical methods. In particular, we build procedures with automatic choice of the smoothing parameters (bandwidths), which is especially interesting for practitioners. This chapter is written to be self-contained. However, in order to make this chapter easier to understand, we recommend the reading of the "nontheoretical" Chapters 2, 3, 4 and 5. After the description of the various routines, we propose a short case study which allows one to understand how work such procedures and how they can be easily implemented. Finally, the source codes, functional datasets, descriptions of the $R/S+$ routines and guidelines for use are given with much more detail in the companion website *http://www.lsp.ups-tlse.fr/staph/npfda*.

7.1 Computing Estimators

We focus on the various functional nonparametric prediction methods. For each of them, we present the kernel estimator and its corresponding implementations through $R/S+$ subroutine. Most of the programs propose an automatic method for selecting the smoothing parameter (i.e., for choosing the bandwidths), which makes these procedures particularly attractive for practitioners. Concerning the functional nonparametric regression method, special attention is paid because it was the first one developed from an historical point of view and because it is the most popular from a general statistical point of view. Therefore, in this regression context we propose several kernel estimators with various automatic selections of the smoothing parameter.

Note that we consider only two families of semi-metrics (computed via `semimetric.pca` or `semimetric.deriv`) described in Chapter 3, two basic kernel functions (see routines `triangle` or `quadratic` in the companion website[1]) described in Chapter 4 and the corresponding integrated ones (see routines `integrated.quadratic` and `integrated.triangle` in the website[1]) described in Chapter 5. However, the following package of subroutines can be viewed as a basic library; any user, according to his statistical and programmer's level, can increase this library by adding his own kernels, integrated kernels or semi-metrics routines.

We recall that we focus on the prediction problem which corresponds to the situation when we observe n pairs $(\boldsymbol{x}_i, y_i)_{i=1,\ldots,n}$ independently and identically distributed: $\boldsymbol{x}_i = \{\chi_i(t_1), \ldots, \chi_i(t_J)\}$ is the discretized version of the curve $\chi_i = \{\chi_i(t);\ t \in T\}$ measured at J points $t_1, \ldots, t_J$ whereas the y_i's are scalar responses. In addition, $\boldsymbol{d}_q(\boldsymbol{x}_i, \boldsymbol{x}_{i'})$ denotes any semi-metric (index of proximity) between the observed curves $\boldsymbol{x}_i$ and $\boldsymbol{x}_{i'}$. So, the statistical problem consists in predicting the responses from the curves.

7.1.1 Prediction via Regression

First, we consider the kernel estimators defined previously in (5.23). It achieves the prediction at an observed curve $\boldsymbol{x}_{i'}$ by building a weighted average of the y_i's for which the corresponding $\boldsymbol{x}_i$ is such that the quantity $\boldsymbol{d}_d(\boldsymbol{x}_i, \boldsymbol{x}_{i'})$ is smaller than a positive real parameter h called bandwidth. In a second attempt, we will consider a slightly modified version in which we replace the bandwidth h by the number k of $\boldsymbol{x}_i$'s that are taken into account to compute the weighted average; such methods use the terminology k-Nearest Neighbours . For both kernel and k-NN estimators, we propose various procedures, the most basic one being the case when the user fixes himself the smoothing parameter h or k. Any other routine achieves an automatic selection of the smoothing parameter. So, if the practitioner wishes to test several different bandwidths, the basic routines can be used, or, in the opposite case, let the other routines automatically choose them.

- **Functional kernel estimator without bandwidth selection**

  The main goal is to compute the quantity:

$$R^{kernel}(\boldsymbol{x}) \;=\; \frac{\sum_{i=1}^{n} y_i K\left(\boldsymbol{d}_q(\boldsymbol{x}_i, \boldsymbol{x})/h\right)}{\sum_{i=1}^{n} K\left(\boldsymbol{d}_q(\boldsymbol{x}_i, \boldsymbol{x})/h\right)},$$

  where $(\boldsymbol{x}_i, y_i)_{i=1,\ldots,n}$ are the observed pairs and $\boldsymbol{x}$ is an observed curve at which the regression is estimated. The user has to fix the bandwidth h, the semi-metric $\boldsymbol{d}_q(.,.)$ and the kernel function $K(.)$. The routine `funopare.kernel` computes the quantities

[1] http://www.lsp.ups-tlse.fr/staph/npfda

$$R^{kernel}(\boldsymbol{z}_1),\ R^{kernel}(\boldsymbol{z}_2),\ldots,R^{kernel}(\boldsymbol{z}_{n'}),$$

where $\boldsymbol{z}_1,\ldots,\boldsymbol{z}_{n'}$ is either a new set of discretized curves or the original one $(\boldsymbol{x}_1,\ldots,\boldsymbol{x}_n)$.

- **Functional kernel estimator with automatic bandwidth selection**

The main goal is to compute the quantity:

$$R^{kernel}_{CV}(\boldsymbol{x}) \ = \ \frac{\sum_{i=1}^n y_i K\left(\boldsymbol{d}_q(\boldsymbol{x}_i,\boldsymbol{x})/h_{opt}\right)}{\sum_{i=1}^n K\left(\boldsymbol{d}_q(\boldsymbol{x}_i,\boldsymbol{x})/h_{opt}\right)},$$

where $(\boldsymbol{x}_i,y_i)_{i=1,\ldots,n}$ are the observed pairs and h_{opt} is the data-driven bandwidth obtained by a cross-validation procedure:

$$h_{opt} \ = \ \arg\min_h CV(h)$$

where

$$CV(h)=\sum_{i=1}^n\left(y_i-R^{kernel}_{(-i)}(\boldsymbol{x}_i)\right)^2,$$

with

$$R^{kernel}_{(-i)}(\boldsymbol{x}) \ = \ \frac{\displaystyle\sum_{j=1,j\neq i}^n y_j K\left(\boldsymbol{d}_q(\boldsymbol{x}_j,\boldsymbol{x})/h\right)}{\displaystyle\sum_{j=1,j\neq i}^n K\left(\boldsymbol{d}_q(\boldsymbol{x}_j,\boldsymbol{x})/h\right)}.$$

The user has to fix the semi-metric $\boldsymbol{d}_q(.,.)$ and the kernel function $K(.)$. The routine `funopare.kernel.cv` computes the quantities

$$R^{kernel}_{CV}(\boldsymbol{z}_1),\ R^{kernel}_{CV}(\boldsymbol{z}_2),\ldots,R^{kernel}_{CV}(\boldsymbol{z}_{n'}),$$

where $\boldsymbol{z}_1,\ldots,\boldsymbol{z}_{n'}$ is either a new set of discretized curves or the original one $(\boldsymbol{x}_1,\ldots,\boldsymbol{x}_n)$.

- **Functional kernel estimator with fixed number of neighbours**

The main goal is to compute the quantity:

$$R^{kNN}(\boldsymbol{x}) \ = \ \frac{\sum_{i=1}^n y_i K\left(\boldsymbol{d}_q(\boldsymbol{x}_i,\boldsymbol{x})/h_k(\boldsymbol{x})\right)}{\sum_{i=1}^n K\left(\boldsymbol{d}_q(\boldsymbol{x}_i,\boldsymbol{x})/h_k(\boldsymbol{x})\right)},$$

where $(\boldsymbol{x}_i,y_i)_{i=1,\ldots,n}$ are the observed pairs and where $h_k(\boldsymbol{x})$ is a bandwidth for which there are exactly k curves among the $\boldsymbol{x}_i$'s such that $d_q(\boldsymbol{x}_i,\boldsymbol{x})<h_k(\boldsymbol{x})$. The user has to fix the semi-metric $\boldsymbol{d}_q(.,.)$, the kernel function $K(.)$ and the number k. The routine `funopare.knn` computes the quantities

$$R^{kNN}(\boldsymbol{z}_1),\ R^{kNN}(\boldsymbol{z}_2),\ldots,R^{kNN}(\boldsymbol{z}_{n'}),$$

where $\boldsymbol{z}_1,\ldots,\boldsymbol{z}_{n'}$ is either a new set of discretized curves or the original one $(\boldsymbol{x}_1,\ldots,\boldsymbol{x}_n)$.

- **Kernel estimator with *global* choice of the number of neighbours**

  The main goal is to compute the quantity:

$$R^{kNN}_{GCV}(\boldsymbol{x}) \;=\; \frac{\sum_{i=1}^{n} y_i K\left(\boldsymbol{d}_q(\boldsymbol{x}_i,\boldsymbol{x})/h_{k_{opt}}(\boldsymbol{x})\right)}{\sum_{i=1}^{n} K\left(\boldsymbol{d}_q(\boldsymbol{x}_i,\boldsymbol{x})/h_{k_{opt}}(\boldsymbol{x})\right)},$$

  where $(\boldsymbol{x}_i, y_i)_{i=1,\ldots,n}$ are the observed pairs and $h_{k_{opt}}(\boldsymbol{x})$ is the bandwidth corresponding to the optimal number of neighbours obtained by a cross-validation procedure:

$$k_{opt} \;=\; \arg\min_{k} GCV(k)$$

  where

$$GCV(k) = \sum_{i=1}^{n} \left(y_i - R^{kNN}_{(-i)}(\boldsymbol{x}_i)\right)^2$$

  with

$$R^{kNN}_{(-i)}(\boldsymbol{x}) \;=\; \frac{\sum\limits_{j=1,j\neq i}^{n} y_j K\left(\boldsymbol{d}_q(\boldsymbol{x}_j,\boldsymbol{x})/h_k(\boldsymbol{x})\right)}{\sum\limits_{j=1,j\neq i}^{n} K\left(\boldsymbol{d}_q(\boldsymbol{x}_j,\boldsymbol{x})/h_k(\boldsymbol{x})\right)}.$$

  The term *global* selection means that we use the same number of neighbours at any curve: $h_{k_{opt}}(\boldsymbol{x})$ depends clearly on $\boldsymbol{x}$ (the bandwidth $h_{k_{opt}}(\boldsymbol{x})$ is such that only the k_{opt}-nearest neighbours of $\boldsymbol{x}$ are taken into account) but k_{opt} is the same for any curve $\boldsymbol{x}$. So, the user has to fix the semi-metric $\boldsymbol{d}_q(.,.)$ and the kernel function $K(.)$. The routine `funopare.knn.gcv` computes the quantities

$$R^{kNN}_{GCV}(\boldsymbol{z}_1),\; R^{kNN}_{GCV}(\boldsymbol{z}_2), \ldots, R^{kNN}_{GCV}(\boldsymbol{z}_{n'}),$$

  where $\boldsymbol{z}_1, \ldots, \boldsymbol{z}_{n'}$ is either a new set of discretized curves or the original one $(\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n)$.

- **Kernel estimator with *local* choice of the number of neighbours**

  The main goal is to compute the quantity:

$$R^{kNN}_{LCV}(\boldsymbol{x}) \;=\; \frac{\sum_{i=1}^{n} y_i K\left(\boldsymbol{d}_q(\boldsymbol{x}_i,\boldsymbol{x})/h_{k_{opt}(\boldsymbol{x}_{i_0})}\right)}{\sum_{i=1}^{n} K\left(\boldsymbol{d}_q(\boldsymbol{x}_i,\boldsymbol{x})/h_{k_{opt}(\boldsymbol{x}_{i_0})}\right)},$$

  where $(\boldsymbol{x}_i, y_i)_{i=1,\ldots,n}$ are the observed pairs, $i_0 = \arg\min_{i=1,\ldots,n} \boldsymbol{d}_q(\boldsymbol{x}, \boldsymbol{x}_i)$ and $h_{k_{opt}(\boldsymbol{x}_{i_0})}$ is the bandwidth corresponding to the optimal number of neighbours at $\boldsymbol{x}_{i_0}$ obtained by:

$$k_{opt}(\boldsymbol{x}_{i_0}) \;=\; \arg\min_{k} \left| y_{i_0} - \frac{\sum_{i=1, i\neq i_0}^{n} y_i K\left(\boldsymbol{d}_q(\boldsymbol{x}_i, \boldsymbol{x}_{i_0})/h_{k(\boldsymbol{x}_{i_0})}\right)}{\sum_{i=1, i\neq i_0}^{n} K\left(\boldsymbol{d}_q(\boldsymbol{x}_i, \boldsymbol{x}_{i_0})/h_{k(\boldsymbol{x}_{i_0})}\right)} \right|.$$

The main difference from the previous estimator appears in the local aspect of the bandwidth. More precisely, the optimal number of neighbours can change from one curve to another one. This is the reason why we use the term *local* selection. The user has to fix the semi-metric $\boldsymbol{d}_q(.,.)$ and the kernel function $K(.)$. The routine `funopare.knn.lcv` computes the quantities

$$R^{kNN}_{LCV}(\boldsymbol{z}_1),\; R^{kNN}_{LCV}(\boldsymbol{z}_2), \ldots, R^{kNN}_{LCV}(\boldsymbol{z}_{n'}),$$

where $\boldsymbol{z}_1, \ldots, \boldsymbol{z}_{n'}$ is either a new set of discretized curves or the original one $(\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n)$.

7.1.2 Prediction via Functional Conditional Quantiles

This section deals with the prediction method via the kernel estimation of the functional conditional quantile. Unlike the previous regression techniques, this kind of method introduces a smoothing parameter for the response in addition to the one needed for the curves. We have only developed the most sophisticated procedure, that is the one involving local automatic bandwidths choices. In order to reduce the computational cost, we prefer a data-driven procedure for selecting these parameters based on two learning subsamples (instead of a standard cross-validation method). Finally, the smoothing parameters are expressed in terms of k-Nearest Neighbours in a local way (i.e., the number k can differ from one unit to another).

The main goal is to compute the quantity:

$$\forall \alpha \in (0, 1/2),\; t^{kNN}_{\alpha}(\boldsymbol{x}) \;=\; \inf_{y \in S} \left\{ F^{kNN}_{k_{opt}, \kappa_{opt}}(\boldsymbol{x}, y) \geq 1 - \alpha \right\},$$

with

$$F^{kNN}_{k,\kappa}(\boldsymbol{x}, y) \;=\; \frac{\sum_{i \in I} K\left(\boldsymbol{d}_q(\boldsymbol{x}_i, \boldsymbol{x})/h_k\right)\; H\left((y - y_i)/g_\kappa\right)}{\sum_{i \in I} K\left(\boldsymbol{d}_q(\boldsymbol{x}_i, \boldsymbol{x})/h_k\right)}.$$

The $(\boldsymbol{x}_i, y_i)_{i \in I}$ are observed pairs, h_k is defined as before in regression and g_κ is the bandwidth for which there are exactly κ among the responses y_i's such that $|y_i - y| < g_\kappa$.

In order to obtain the optimal numbers of neighbours ($k_{opt}(\boldsymbol{x})$ and $\kappa_{opt}(y)$), we randomly split our learning sample into two learning subsamples:

$$(\boldsymbol{x}_{i_1}, y_{i_1})_{i_1 \in I_1},\; (\boldsymbol{x}_{i_2}, y_{i_2})_{i_2 \in I_2},\; I_1 \cap I_2 = \phi, I_1 \cup I_2 = I \text{ and } card(I_1) = [card(I)/2],$$

and we define for each $\boldsymbol{x}$, $i^* = \arg\min_{i_2 \in I_2} \boldsymbol{d}_q(\boldsymbol{x}, \boldsymbol{x}_{i_2})$. Then, we compute k_{opt} and κ_{opt} as follows:

$$(k_{opt}, \kappa_{opt}) \;=\; \arg\min_{(k,\kappa)} \left| y_{i^*} - \inf_{u \in S} \left\{ F_{k,\kappa}^{kNN}(\boldsymbol{x}_{i^*}, u) \geq 1 - \alpha \right\} \right|.$$

The optimal numbers of neighbours (k_{opt} and κ_{opt}) can change from one curve to another one. This is the reason why we use the term *local* selection. So, the user has to fix the semi-metric $\boldsymbol{d}_q(.,.)$, the kernel function $K(.)$ and α. The routine `funopare.quantile.lcv` computes the quantities

$$t_\alpha^{kNN}(\boldsymbol{z}_1),\; t_\alpha^{kNN}(\boldsymbol{z}_2), \ldots, t_\alpha^{kNN}(\boldsymbol{z}_{n'}),$$

where $\boldsymbol{z}_1, \ldots, \boldsymbol{z}_{n'}$ is either a new set of discretized curves or the original one $(\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n)$.

7.1.3 Prediction via Functional Conditional Mode

We focus now on the prediction method via the kernel estimation of the functional conditional mode. As previously, the computed estimator needs the selection of two smoothing parameters in terms of k-Nearest Neighbours in a local way (the number k can differ from one unit to another one). The data-driven procedure for choosing these parameters also involves two learning subsamples; one for building the kernel estimator, one for selecting the smoothing parameters.

The main goal is to compute the quantity:

$$\theta^{kNN}(\boldsymbol{x}) \;=\; \arg\sup_{y \in S} \widehat{f}_{k_{opt},\kappa_{opt}}^{kNN}(\boldsymbol{x}, y),$$

with

$$f_{k,\kappa}^{kNN}(\boldsymbol{x}, y) \;=\; \frac{\sum_{i \in I} K\left(\boldsymbol{d}_q(\boldsymbol{x}_i, \boldsymbol{x})/h_k\right)\, K_0\left((y - y_i)/g_\kappa\right)}{g_\kappa \sum_{i \in I} K\left(\boldsymbol{d}_q(\boldsymbol{x}_i, \boldsymbol{x})/h_k\right)}.$$

The $(\boldsymbol{x}_i, y_i)_{i \in I}$ are observed pairs, h_k and g_κ are defined as before in Section 7.1.2. Following the same steps as in Section 7.1.2, we split our sample into two learning subsamples I_1 and I_2 and we define for each $\boldsymbol{x}$, $i^* = \arg\min_{i_2 \in I_2} \boldsymbol{d}_q(\boldsymbol{x}, \boldsymbol{x}_{i_2})$. Then, we compute k_{opt} and κ_{opt} as follows:

$$(k_{opt}, \kappa_{opt}) \;=\; \arg\min_{(k,\kappa)} \left| y_{i^*} - \arg\sup_{u \in S} f_{k,\kappa}^{kNN}(\boldsymbol{x}_{i^*}, u) \right|.$$

The optimal numbers of neighbours (k_{opt} and κ_{opt}) can change form one curve to another. This is the reason why we use the term *local* selection. So, the user has to fix the semi-metric $\boldsymbol{d}_q(.,.)$ and the kernel function $K(.)$. The routine `funopare.mode.lcv` computes the quantities

$$\theta^{kNN}(\boldsymbol{z}_1),\; \theta^{kNN}(\boldsymbol{z}_2), \ldots, \theta^{kNN}(\boldsymbol{z}_{n'}),$$

where $\boldsymbol{z}_1, \ldots, \boldsymbol{z}_{n'}$ is either a new set of discretized curves or the original one $(\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n)$.

7.2 Predicting Fat Content From Spectrometric Curves

7.2.1 Chemometric Data and the Aim of the Problem

This section focuses on the spectrometric curves described in Section 2.1 and partially displayed in Figure 2.1. We recall that for each unit i (among 215 pieces of finely chopped meat), we observe one spectrometric discretized curve $(\boldsymbol{x}_i)$ which corresponds to the absorbance measured at a grid of 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 analytical chemical processing. The file "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. Given a new spectrometric curve $\boldsymbol{x}$, our main task is to predict the corresponding fat content $\widehat{y}$. In fact, obtaining a spectrometric curve is less expensive (in terms of time and cost) than the analytic chemistry needed for determining the percentage of fat. So, it is an important economic challenge to predict the fat content from the spectrometric curve.

In order to highlight the performance of our functional nonparametric prediction methods, we split our original sample into two subsamples. 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 us to build the functional kernel estimators 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 $\widehat{y}_{161}, \ldots, \widehat{y}_{215}$.

To measure the performance of each functional prediction method, we consider

i) the distribution of the *Square Errors*: $se_i = (y_i - \widehat{y}_i)^2$, $i = 161, \ldots, 215$, and

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

7.2.2 Functional Prediction in Action

We have 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 functional conditional mode one and the functional conditional median one. The $R/S+$ commandlines and their corresponding explanations enabling one to load the dataset, to run the subroutines and to display the results are available on the website[2]. We end this analysis by comparing these methods through the empirical Mean Square Errors (MSE) and by suggesting a substantial improvement.

The smoothness of the curves allow us to use the semi-metrics based on the derivatives. After trying some of them, it turns out that the best one is based on the second order derivatives. The results are summarized in Figure 7.1. Conditional mode and conditional median give very similar results whereas the conditional expectation seems sensitive to high values of the response. Nevertheless, the three methods give good predictions.

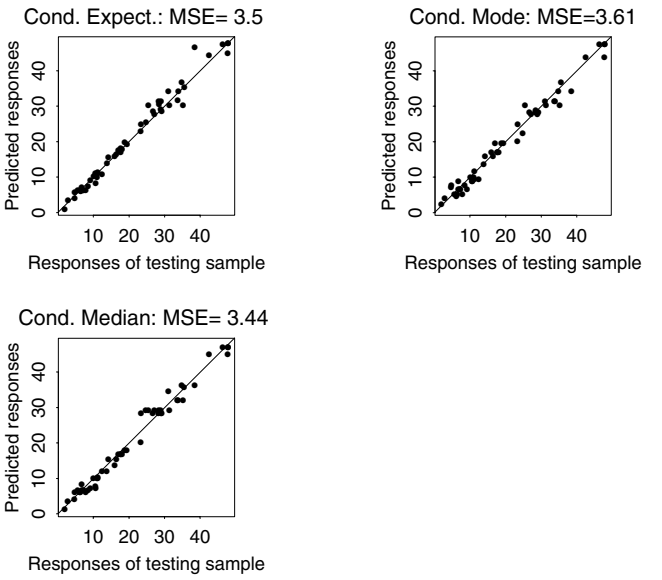

Fig. 7.1. Performance of the Three Functional Prediction Methods on Spectrometric Data

[2] http://www.lsp.ups-tlse.fr/staph/npfda

Because there are some differences from one method to another, one way to improve the results is to produce predictions by averaging those obtained with each method; the corresponding methodology is called *Multimethods*. As shown in Figure 7.2, the result is very interesting. There is a significant gain both in terms of mean square error and concerning the dispersion of the square error.

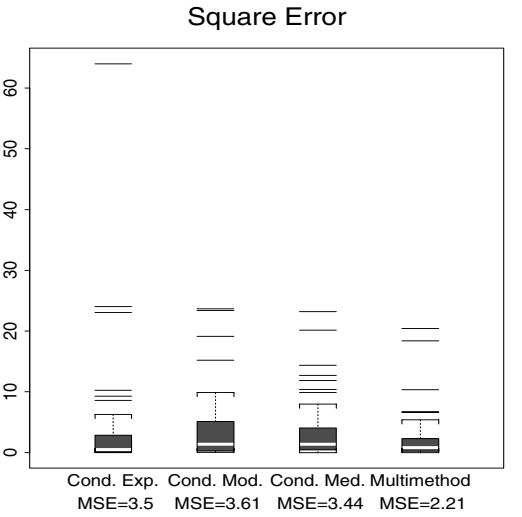

Fig. 7.2. Comparison Between the Three Functional Prediction Methods and the Multimethod One for Spectrometric Data

Note finally that the goal of this application was not a comparison study with other competitive methods but just to implement the three functional nonparametric prediction methods. Of course, the reader can download the spectrometric dataset on our website[3] and compare these functional statistical methods with alternative ones.

7.3 Conclusion

According to the previous results, one can say that our functional prediction methods are easy to implement and they work well for predicting fat content given spectrometric curves. In addition, subtantial improvements in terms of errors of prediction can be achieved by using the three predictive functional

[3] http://www.lsp.ups-tlse.fr/staph/npfda

techniques (for instance through the average of the three predictions). We will see later on that such functional methods still gives good results in the forecasting setting (i.e., time series, see Chapter 11) with another functional dataset (i.e., electricity consumption data, see Section 2.3). To conclude, let us emphasize with slight contributions, any user may easily incorporate his own semi-metrics, kernels, . . . still using the main bodies of our programs.