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. $\boldsymbol{x}_i = (\chi_i(\lambda_1), \dots, \chi_i(\lambda_{100}))$. 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 $(\boldsymbol{x}_i, y_i)_{i=1,\dots,215}$. 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 ($(\boldsymbol{x}_i, y_i)_{i=1,\dots,160}$). The second one, called *testing sample*, contains the last 55 units ($(\boldsymbol{x}_i, y_i)_{i=161,\dots,215}$). 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 $\boldsymbol{x}_{161}, \dots, \boldsymbol{x}_{215}$ ($y_{161}, \dots, y_{215}$ being ignored) which allows us to get the predicted responses $\widehat{y}_{161}, \dots, \widehat{y}_{215}$.

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

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

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

  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 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 2.

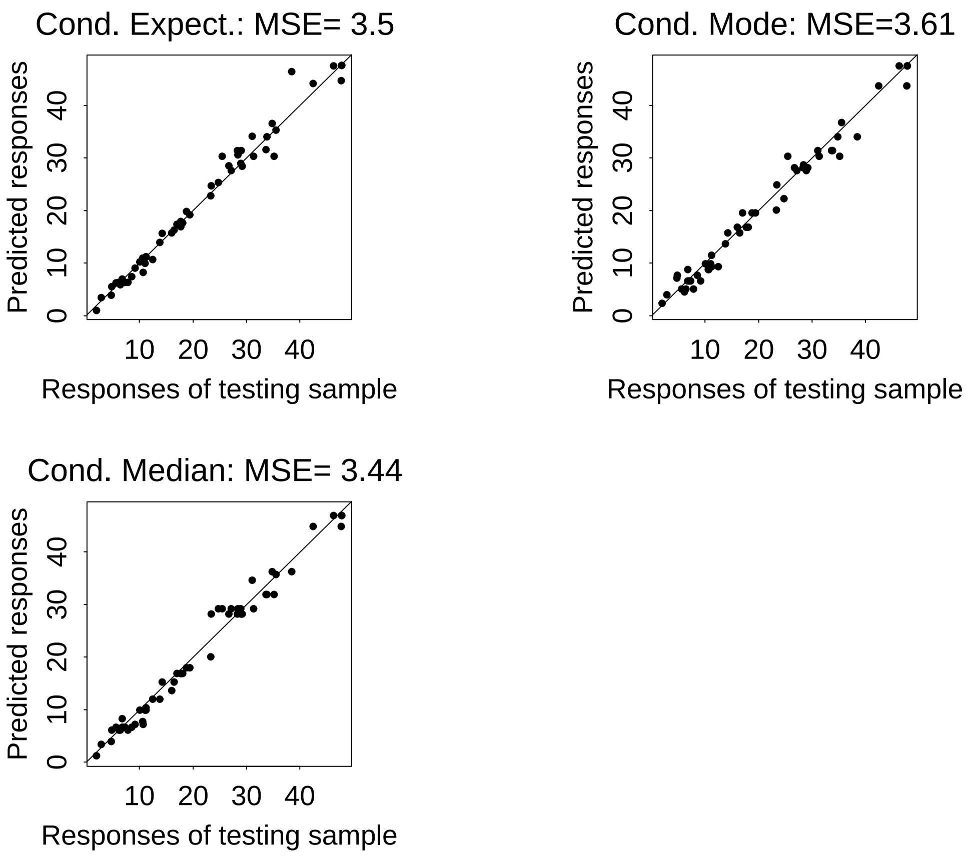

Figure 1: Performance of the three functional prediction methods

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

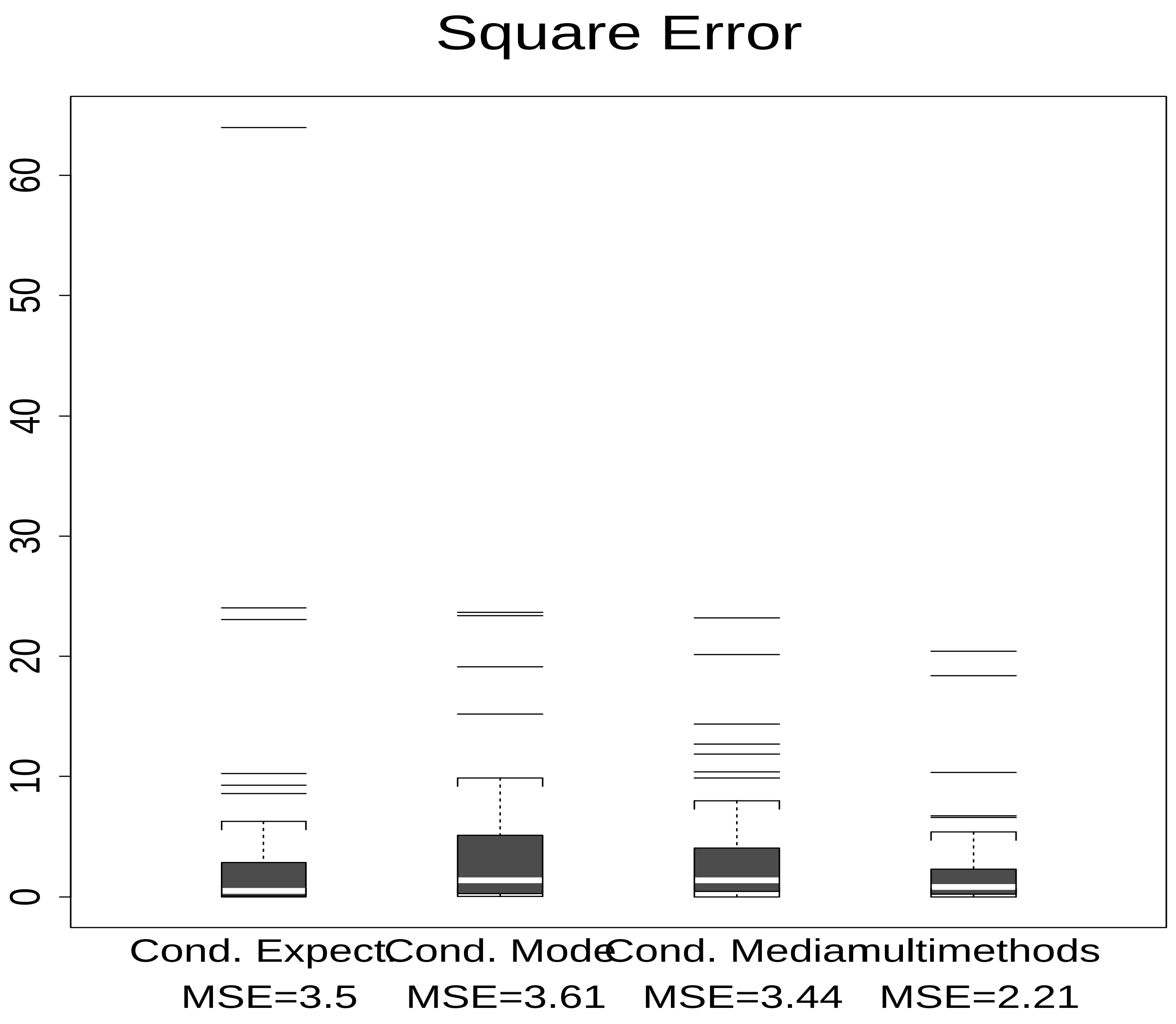

Figure 2: Comparison between the three functional prediction methods and the multimethod one