UNIVERSIDADE DE SANTIAGO DE COMPOSTELA
DEPARTAMENTO DE ESTATÍSTICA E INVESTIGACIÓN OPERATIVA

Utilities for Statistical Computing in Functional Data Analysis: The R Package fda.usc

M. Oviedo de la Fuente, M. Febrero-Bande

Report 11-02

Reports in Statistics and Operations Research

Utilities for Statistical Computing in Functional Data Analysis: The R Package **fda.usc**

Manuel Oviedo de la Fuente Manuel Febrero-Bande
Department of Statistic and Operations Research
University of Santiago of Compostela
Spain, May 2011

Abstract

This paper is devoted the **R** package **fda.usc** which includes some utilities for functional data analysis. This package carries out exploratory and descriptive analysis of functional data analyzing its most important features such as depth measurements or functional outliers detection, among others. The **R** package **fda.usc** also includes functions to compute functional regression models, with a scalar response and a functional explanatory data via nonparametric functional regression, basis representation or functional principal components analysis. There are natural extensions such as functional linear models and semi-functional partial linear models, which allow non functional covariates and factors and make predictions. The functions of this package complement and incorporate the two main references of functional data analysis: The **R** package **fda** and the functions implemented by [Ferraty and Vieu (2006)].

Keywords: functional data regression, representation of functional data, nonparametric kernel estimation, depth measures, outlier

1 Introduction

The technological progress has led to the development of new, quick and accurate measurement procedures. As a consequence, further possibilities for obtainig experimental data are now available and some classical paradigms must be revised. For example, it is now possible (and even frequent) to find problems where the number of data is greater than the number of variables. In many areas it is common to work with large databases, which increasingly often these observations of a random variable taken over a continuous interval (or in increasingly larger discretizations of the continuous interval).
For example, in fields such as spectroscopy, the measurement result is a curve that, at least, has been evaluated in 100 points. This type of data, which we call functional data arise naturally in many disciplines. In economics, one could consider curves intra-day stock quotes. In environmental studies, one could find continuous measurements of atmospheric monitoring networks. It is also well-known the importance of functional data in image recognition or spatio temporal information.

Undoubtedly, package **fda** ([Ramsay *et al.* (2010)]) is a basic reference to work in **R** programming environment ([**R** Development Core Team(2011)]) with functional data. The book by is well-known referencein this field, [Ramsay and Silverman (2005)] has helped to popularize the statistical techniques for functional data. All the techniques included are restricted to the space of $\mathcal{L}_2$ functions (the Hilbert space of all square integrable functions over a certain interval). The book by [Ferraty and Vieu (2006)] is another important reference incorporating non-parametric approaches as well as the us of other theoretical tools such as seminorms and small ball probabilities. These authors are part of the French group STAPH maintaining the page http://www.lsp.ups-tlse.fr/staph/ where R software can be downloaded. Other functional data packages that can be useful for representing functional data are: The package **rainbow** ([Shang and Hyndman (2010)]) for functional data display and outlier detection, the package **fds** ([Hyndman and Shang (2010a)]) with functional data sets and the package **ftsa** ([Hyndman and Shang (2010b)]) for functional time series analysis.
The aim of the package **fda.usc** is to provide broader, flexible tool for the analysis of functional data. Therefore, we propose an integration of the work on nonparametric functional methods implemented by [Ferraty and Vieu (2006)] those from the group of the University of Santiago de Compostela (USC), thus complementing and extending some of the functions of package **fda**.
In Section 2.1 we introduce a new **R** class for functional data: *"fdata"*. Section 2.2 describes the functional representation of an object of class *"fdata"* by basis representation or kernel smoothing. Section 2.3 is focused in normed and semi-normed functional spaces and in how the metrics and semimetrics can be used as measures of proximity between functional data. Section 2.4 is concerned with the concept of statistical depth for functional data see [Cuevas *et al.* (2007)]. These depth measures are useful to define location and dispersion measures, for classification and for outlier detection [Febrero-Bande *et al.* (2008)] (Section 2.6). Section 3 is devoted to functional regression models with scalar response from different perspectives: Parametric, using basis representation (Section 3.1, 3.2 and 3.3), nonparametric, using smoothing kernel (Section 3.4) and semi-parametric combines both (Section 3.5). The Section 3.6 summarizes the functional regression models using a classical example to show the capabilities of the package **fda.usc**. Finally, Section 4 discuss the most important feature of the package.

2 Functional Data: Definition and descriptive analysis

After a brief introduction of the state of the art in functional data, we describe the most important features, procedures and utilities of the new package.

2.1 Functional Data Definition: The new R class fdata

A functional variable $\mathcal{X}$ is just a random variable taking values in a function space $\mathcal{E}$. Thus, a functional data set is just a sample $\{\mathcal{X}_1, \ldots, \mathcal{X}_n\}$ (also denoted $X_1(t), \ldots, X_n(t)$ when convenient) drawn from a functional variable $\mathcal{X}$. Usually $\mathcal{E}$ is assumed to be a normed or seminormed metric space...(completar definicion)
The first obstacle that we will always have when analyzing functional data is to find an adequate representation for the data. Typically, the functional data set $\{\mathcal{X}_1, \ldots, \mathcal{X}_n\}$ is evaluated on a number of discretization points $\{t_1, \ldots, t_m\}$ that can be observed in a non-equispaced way.

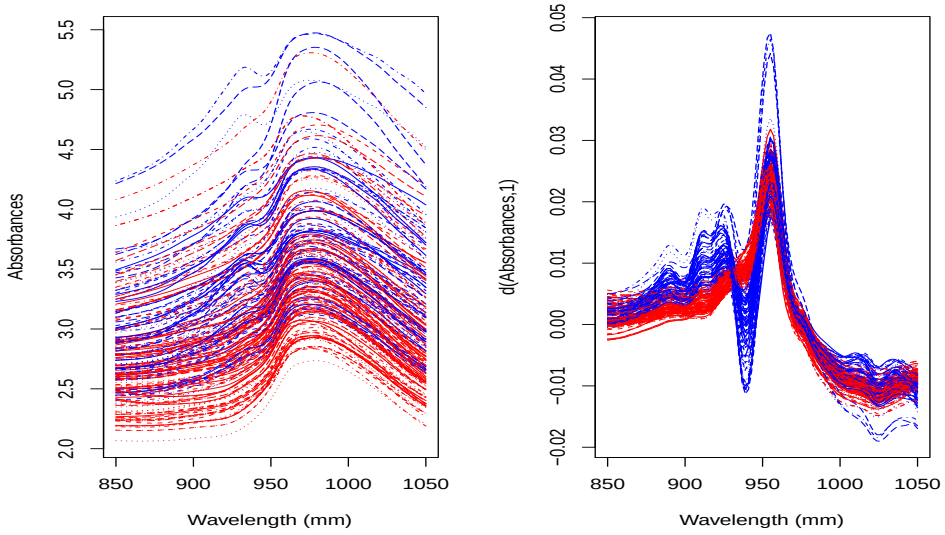

Figure 1: Spectrometric curves (left panel): curves with Fat¡20% (red lines) and curves with Fat≥20% (blue lines). Corresponding spectrometic curves after first order differencing (right panel).

The **fda.usc** package avoids the basis transformation performed by the **fda** package and define an object called **fdata** as a list of the following components:

- **data**: Typically a matrix of *(n,m)* dimension which contains a set of n curves discretized in m points or **argvals**.
- **argvals**: Locations of the discretization points, by default: $\{t_1 = 1, \ldots, t_m = m\}$.
- **rangeval**: Range of discretization points, by default: **range(argvals)**.
- **names**: Optional list with three components: **main**, an overall title, **xlab**, a title for the x axis and **ylab**, a title for the y axis.

All the methods in the package can work with the new class **fdata** that only uses the evaluations at the discretization points. There is no need to represent the functional data in a basis object. Also, some basic operations to handle this new class are introduced: "+", "-", "*", "/", "[]", "==", **is.fdata()**, **c()**, **dim()**, **ncol()**, **nrow()**.
The representation of a functional dataset should be consistent with the physical interpretation of the phenomenon being described. The **tecator** dataset (**data("tecator")**) has two components. The first component, **tecator$absorp.fdata**, includes the curves of absorbance for the analysis of pieces of meat stored in the format of class **fdata**. Each curve consists of a m=100-channel absorbance spectrum measured along the discretized points: **argvals**=$\{t_1 = 850, \ldots, t_m = 1050\}$. The second component, **tecator$y**, it is a data-frame including **Fat**, **Water** and **Protein** contents of each spectrometric curve obtained by an analytical chemical processing. The **tecator** dataset presents interesting features and it is a well known example in Functional Data Analysis (FDA), therefore we have incorporated into the package (available at http://lib.stat.cmu.edu/datasets/tecator).

In this package, many utility functions are incorporated to work with the new class **fdata**, as for example **plot()**. The following code displays the absorbance curves in function of the fat content, (see Figure 1).

```
R> library("fda.usc")
R> data("tecator")
R> names(tecator)
[1] "absorp.fdata" "y"
\end{CodeOutput}
\begin{CodeInput}
R> absorp <- tecator$absorp.fdata
R> Fat20 <- ifelse(tecator$y$Fat < 20, 0, 1) * 2 + 2
R> plot(tecator$absorp.fdata, col = Fat20)
```

When analyzing functional data it is important to choose the space where the data must be considered. For example, in left panel of Figure 1, the spectrometric curves are plotted showing with different colors the fat content. This representation which implicitly assumes a $\mathcal{L}_2$ space, is not related with the information of fat content. In other words, the vertical shift of these curves has no special relation with the fat content. So, if we are interested in the relationship between the spectromectric curve and the fat content, probably another choice of functional space would be more advisable as for example, that shown in the right panel of Figure 1 with the first derivative of the spectrometric curves.
In this case, the distances between curves are given by the semi-norm of the derivative $\left(d(f,g) = \sqrt{\int_T (f'(t) - g'(t))^2 dt}\right)$ instead of the norm as in the case of the $\mathcal{L}_2$ space . To calculate the derivative of a functional data the **fdata.deriv()** function has been implemented, which compute the derivative many argument by different methods. If **method** = **"bspline"**, **"exponential"**, **"fourier"**, or **"'monomial"** is selected, the command calculates the derivative of the fdata object using **deriv.fd{fda}**. If **method**=**"fmm"**, **"periodic"**, **"natural"** or **"monoH.FC"** is selected, the **splinefun{stats}** function is employed. Raw derivation could be applied with **method**=**"diff"**, but this is not recommended when the values are not equally spaced o with sparse data.

```
R> absorp.d1 = fdata.deriv(absorp, nderiv = 1 , method = "bspline" )
R> plot(absorp.d1, col = Fat20)
```

2.2 Functional data representation: Smoothing

The first step in a functional data analysis maybe the data representation. If we assume that our functional data $Y(t)$ is observed through the model: $Y(t_i) = X(t_i) + \epsilon(t_i)$ where the residuals $\epsilon(t)$ are independent from $X(t)$, we can get back the original signal $X(t)$ using a linear smoother,

$$\hat{x} = \sum_{i=1}^{n} s_{ij} y_i \text{ or } \hat{x} = Sy$$

where s_{ij} is the weight that the point t_j gives to the point t_i and $y_i = X(t_i), x_i = X(t_i)$. In the package two procedures have been implemented for this purpose. The first one is the representation in a $\mathcal{L}_2$ basis (or penalized basis) and the second one is based on the smoothing kernel methods.

a) Basis representation. A curve can be represented by a basis when the data are assumed to belong to $\mathcal{L}_2$ space. A basis is a set of known functions $\{\phi_k\}_{k\in\mathbb{N}}$ that any function could be arbitrarily approximated by a linear combination of a sufficiently large number K of these functions, (see page 43 and 44 of [Ramsay and Silverman (2005)]).
The procedure approximates a function $X(t)$ by using a fixed truncated basis expansion in terms of K known basis functions,

$$X(t) = \sum_{k\in\mathbb{N}} c_k\phi_k(t) \approx \sum_{k=1}^{K} c_k\phi_k(t) = \mathbf{c^T\Phi} \tag{1}$$

The projection (or smoothing) matrix is given by: $S = \Phi(\Phi^T W\Phi)^{-1}\Phi^T W$, with degrees of freedom of the fit $df(\nu) = trace(S(\nu)) = K$.
If smoothing penalization is required, also a parameter λ will be provided and in this case the projection matrix S is: $S = \Phi(\Phi^T W\Phi + \lambda R)^{-1}\Phi^T W$, where R is the penalization matrix.
This package also includes the function **create.*basis-name*.basis()**. Also, the function **fdata2fd()** converts an object of class **fdata** in to an object of class **fda** using the basis representation shown in 1. Inversely, the function **fdata()** converts data object of class: **fda**, **fds**, **fts**, **sfts** or another format (**vector**, **matrix**, **data.frame**) to an object of class **fdata**.

```
R> class( absorp.fd <- fdata2fd( absorp, type.basis= "fourier", nbasis= 15))
[1] "fd"
R> class(absorp.fdata <- fdata(absorp.fd))
[1] "fdata"
```

The choice of the parameter number of basis elements that of and the most appropriate basis for the observed data is also crucial and, in principle, there is no universal rule that would enable an optimal choice. The decision on what basis to choose should be based on the objective of the study and on the data. For example, is common to use the *"fourier"* basis for periodic data, and *"bspline"* basis for non-recurrent data. Among the different selection criteria to select the parameter $\nu = (K, \lambda)$, we have implemented the two: Cross Validation (CV) and Generalized Cross Validation (GCV). The purpose of the function **min.basis()** is to represent the functional data in terms of a (truncated) expansion with respect to a given basis of functions in the corresponding space. Such expansions depend on a parameter $\nu = (K, \lambda)$, where K is the number of basis functions used in the truncated expansion and λ is the penalization parameter. They are both chosen by cross-validation procedures, as follows:

$$\text{Cross-validation} : CV(\nu) = \frac{1}{n}\sum_{i=1}^{n}\left(y_i - \hat{r}^{\nu}_{-i}(x_i)\right)^2 w(x_i) \tag{2}$$

where $\hat{r}^{\nu}_{-i}(x_i)$ indicates the estimator based on leaving out the i pair (x_i, y_i) and $w(x_i)$ is the weight of data x at point t_i. This criterion is implemented by the function **CV.S()**.

$$\text{Generalized Cross-validation} : GCV(\nu) = \frac{1}{n}\sum_{i=1}^{n}\left(y_i - \hat{r}^{\nu}_{i}(x_i)\right)^2 w(x_i)\Xi(\nu) \tag{3}$$

where $\Xi(\nu)$ denotes the type of penalizing function.

Generalized Cross-Validation criteria is implemented in **GCV.S()** function with the following types of $\Xi(\nu)$ functions: Generalized Cross-validation (GCV), Akaike's Information Criterion (AIC), Finite Prediction Error (FPE), Shibata's model selector (Shibata) or Rice's bandwidth selector (Rice), see [Härdle (1990)].

b) The nonparametric methodology, and in particular, the kernel method, can also be used to represent functional data. Now, the nonparametric smoothing of functional data is given by the smoothing matrix S:

$$s_{ij} = \frac{1}{h} K\left(\frac{t_i - t_j}{h}\right)$$

Different types of kernels $K()$ are considered in the package: Gaussian, Epanechnikov, Triweigth, Uniform, Cosine or other user-defined (see **help(Kernel)**).
The **min.np()** function returns the "optimal" value **h.opt** of the smoothing parameter ν with respect to the cross-validation crietria 2 and 3). Among other features, the package **fda.usc** allows calculate the smoothing matrix **S** by: Nadaraya-Watson method (**S.NW**), K-nearest neighbors method (**S.KNN**) or local linear regression method (**S.LLR**), see [Wasserman (2006)].
The **min.np()** function returns the "optimal" value **h.opt** of the smoothing parameter ν that best represents the functional data $\hat{x} = \hat{r}_n^{\nu}(x) = \sum_{i=1}^{n} s_i(x) Y_i$ for a range of values of bandwidth ν using the validation criteria shown above, (Equation 2 and 3).

Let us know consider, for ilustration purposes, the **Phoneme)** Dataset **data("phoneme")**[1] see, e.g. [Ferraty and Vieu (2006)]. The **phoneme$classlearn** object contains 250 speech frames with class membership: "aa" (1), "ao" (2), "dcl" (3), "iy" (4) and "sh" (5). From each speech frame, a log-periodogram of length 150 have been stored in **phoneme$learn** which is used as learning sample. The goal is to predict the class membership **phoneme$classtest** using the test sample **phoneme$test**.
In the following example (see below **R** code), **phoneme$learn** is smoothed using **min.basis()** and **min.np()** functions. In the left panel of Figure 2, the GCV criterion is drawn in function of the number of *bspline* basis elements $\nu_1 = nbasis$ and penalizing parameter $\nu_2 = \lambda$ using **min.basis()** function. In the right panel of Figure 2, the GCV criterion is drawn in function of the bandwidth $\nu = h$ using the function **min.np()**.

```
R> data("phoneme")
R> learn <- phoneme$learn
R> l <- c(0, 2^seq(-2, 9, len = 30))
R> nb <- seq(7, 31, by = 2)
R> out0 <- min.basis(learn, lambda = l, numbasis = nb)
 The minimum GCV (GCV.OPT=611.4684) is achieved with
 the number of basis (numbasis.opt=27)
 and lambda value    (lambda.opt=81.27995)
R> out1 <- min.np(learn, Ker = Ker.epa)
 The minimum GCV (GCV.OPT=621.0195) is achieved with
 the h value (h.opt=6.3131)
R> out2 <- min.np(learn, type.S = S.LLR)
 The minimum GCV (GCV.OPT=663.8132) is achieved with
 the h value (h.opt=8.8826)
```

[1] This dataset includes the changes introduced in the file
http://www.math.univ-toulouse.fr/staph/npfda/npfda-phondiscRS.txt.

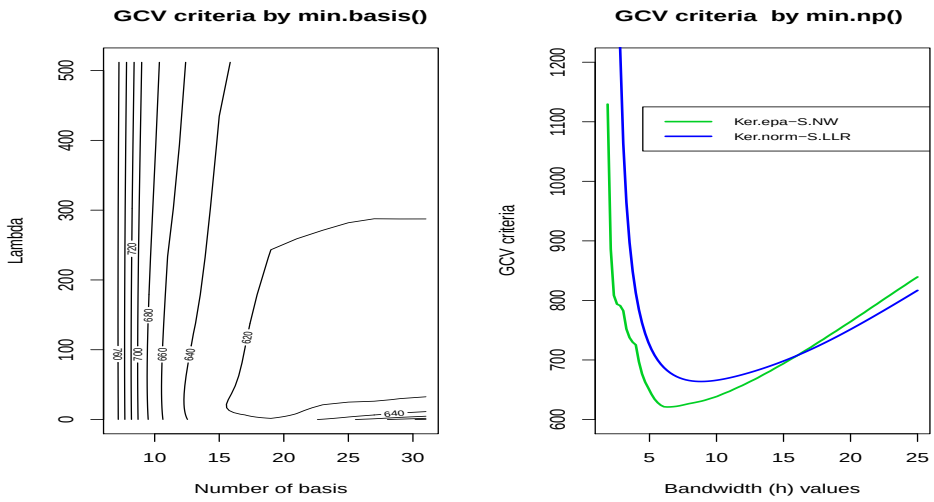

Figure 2: GCV criteria as a function of the number of bspline basis elements and the penalizing parameter λ (left panel). GCV criteria as a function of bandwidth parameter (right panel): Normal kernel and local linear smoothing matrix (blue line); Epanechnikov kernel and Nadaraya-Watson smoothing matrix (green line).

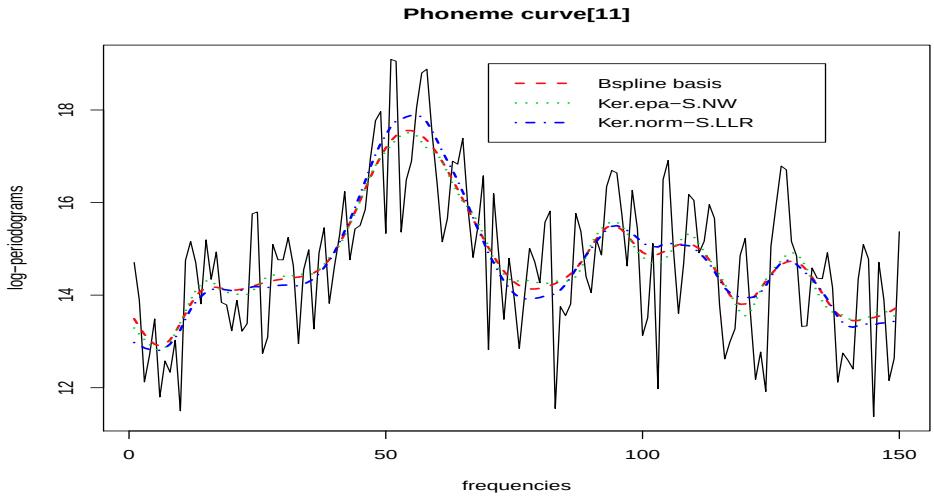

Figure 3: Phoneme curve[11]: Observed (black solid line), smoothed by 27 bspline basis and $\lambda = 81.28$ (red dashed line), smoothed by Epanechnikov kernel and Nadaraya-Watson smoothing matrix with bandwidth $h = 8.88$ (green dotted line) and smoothed by normal kernel and local linear smoothing matrix with bandwidth $h = 6.31$ (blue dashed line).

Figure 3 shows the 11^{th} curve of **phoneme$learn** and three smooth representations of the curve.

2.3 Measuring distances

It is difficult to find the best plot given a particular functional dataset because the shape of the graphics depends strongly on the chosen proximity measure. As shown in Figure 1, the plot of $X(t)$ against t is not necessarily the most informative and maybe another can allow us to extract much information from functional variables. This package collects several metric and semimetric functions which allow us to extract as much information possible as from the functional variable.
The spaces for functional data are the complete metric spaces where only the notion of distance between elements of the space is given. If the metric d is asociatted with a norm (so that $d(X(t), Y(t)) = \|X(t) - Y(t)\|$ we have a normed space (or a Banach space). In some important cases the norm $\|.\|$ is associated with an inner product $,.\langle\rangle$ in the sense that $\|X\| = \langle X, X\rangle^{1/2}$. A complete normeds pace (Banach space) whose norm derives from an inner product is called a Hilbert space. The best known example is the space $L^2[a, b]$ of real square-integrable functions de
ned on $[a, b]$ with $\langle f, g\rangle = \int_a^b fg$.
$\langle x, y\rangle = (1/4)(\|x + y\|^2 - \|x - y\|^2)$. A complete space with an inner product is called a Hilbert space which is a special kind of Banach spaces where $\|X(t)\| = \sqrt{\langle X(t), X(t)\rangle}$.
Utilities for computing distances, norms and inner products are included in the package. For example, a collection of semimetrics proposed by [Ferraty and Vieu (2006)]: **semimetric.deriv()**, **semimetric.fourier()**, **semimetric.hshift()**, **semmimetric.mplsr()** and **semmimetric.pca()** have been included, see **help(semimetric.NPFDA)**. If we focused on $\mathcal{L}_p$ spaces (the set of functions whose absolute value raised to the $_p$-th power has finite integral), **metric.lp()** uses Simpson's rule to compute distances between elements, **norm.fdata()** computes the norm and, specifically for $\mathcal{L}_2$, **inprod.fdata()** calculates the inner product between elements of the space.
The procedures of the package **fda.usc** including the argument **metric** allow us the use of metric or semimetrics functions implemented or other user defined with the only restriction that the first two arguments belong to the class **fdata**.
In the next example, the distances beetwen some training sample curves of phoneme data (**phoneme$learn**) are calculated by a metric function: **metric.lp()**, and several semimetrics functions: **semimmetric.basis()** based on their bspline expansion, **semmimetric.pca()** based on the functional principal components analysis method and **semmimetric.mplsr()** based on the partial least squares method. Figure 4 shows the dendograms for a selection of 11 curves of class **(3)** (corresponding to the indices from 110 to 120 curves) and 11 curves of class **(5)** (from 220 to 230). This example can be understood as a classification problem in which the goal is to classify the curves in 2 classes. Let us note that in this example the **semimmetric.mplsr()** function (which uses memberclass information) is the only one that properly classifies the 22 curves.

```
R> glearn = phoneme$classlearn
R> mdist1 <- metric.lp(learn)
R> mdist2 <- semimetric.basis(learn, type.basis1 = "fourier")
R> mdist4 <- semimetric.pca(learn, learn)
R> mdist5 <- semimetric.mplsr(learn, learn, q = 3, class1 = glearn)
```

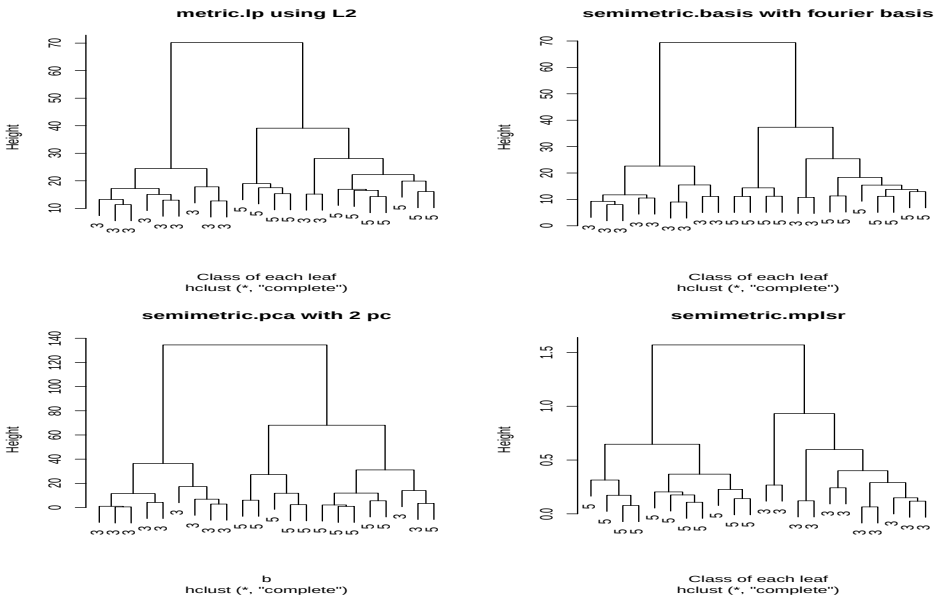

Figure 4: Dendograms for 22 phoneme curves: Dendogram using $\mathcal{L}_2$ metric **metric.lp()** (top left), dendogram using $\mathcal{L}_2$ metric with bspline basis representation **semimetric.basis()** (bottom left), dendogram using 2 principal components **semimetric.pca()** (top right) and dendogram using a semi-metric based on the partial least squares method **semimetric.mplsr()** (bottom right).

2.4 Exploring Functional Data

Any statistical study should begin with an exploratory stage. In **fda.usc**, the usual tools for summarize functional data are included: **func.mean()**, **func.var()** and **pc.svd.fdata()** for computing the mean, the marginal variance and principal eigenfunctions. The output of this tools is always an object of class **fdata**. Different depth notions have been proposed in the literature, with the aim of measuring how deep is a data point in the sample. In univariate data, the median would typically be the deepest point of clouds of points. Although there are more depth measures, this package includes those that are contained in the work of [Cuevas *et al.* (2007)]:

- **depth.FM()**: The depth measure is based on the median, [Fraiman and Muniz (2001)].
- **depth.mode()**: The depth measure is based on how densely surrounded the curves are respect to a metric or a semimetric distance, [Cuevas *et al.* (2007)].
- **depth.RP()**: The depth measure is calculated through random projections (RP) based on the Tukey depth, [Cuevas *et al.* (2007)].
- **depth.RPD()**: The depth measure is calculated through random projections of the curves and their derivatives, [Cuevas *et al.* (2007)].

All depth functions implemented return:

- **median**: Deepest curve.
- **lmed**: Index of the deepest curve.

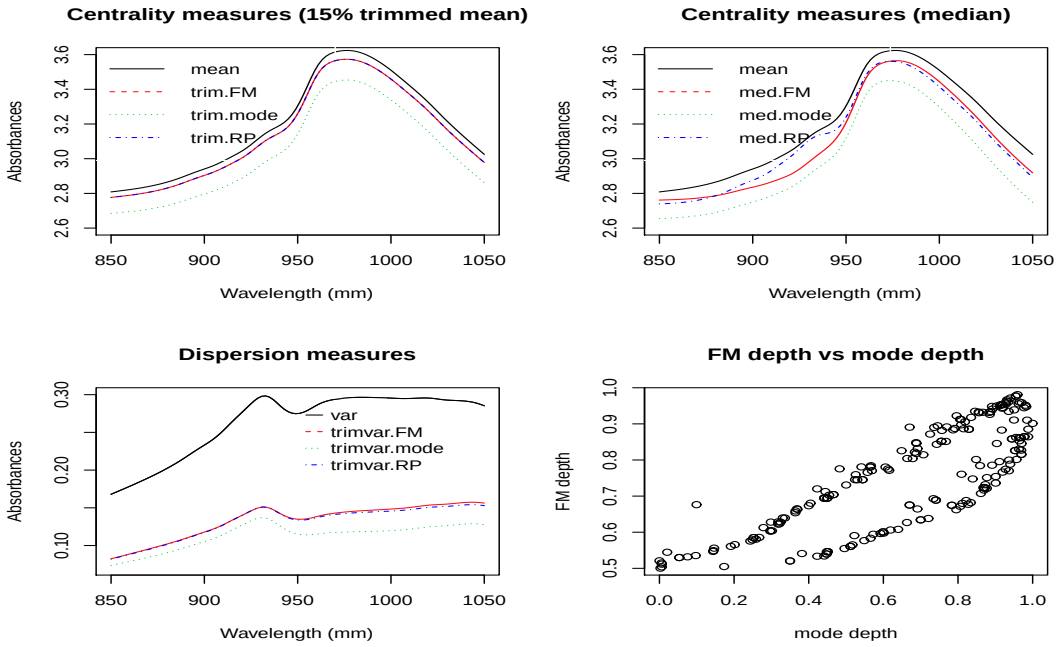

Figure 5: Descriptive statistics for Tecator dataset based on depth: 15%-trimmed mean (top left), medians (top right), dispersion measures (bottom left) and FM depth versus mode depth (bottom right).

- **mtrim**: Mean of $(1-\alpha)\%$ deepest curves.
- **ltrim**: Index of $(1-\alpha)\%$ deepest curves.
- **dep**: Depth of each curve.

An interesting application of the proposed depth measures is their use as measures of central tendency and/or dispersion. The top row of Figure 5 displays the α-trimmed means (top left) and the medians (top right). The bottom row of Figure 5 shows the marginal variance using trimmed subsets (bottom left), and the depths calculated by **depth.FM()** and **depth.mode()** (bottom right).

```
R> plot(func.mean(absorp), main = "Centrality measures (15% trimmed mean)",
+     ylim = c(2.6, 3.6))
R> lines(func.trim.FM(absorp, trim = 0.15), col = 2)
R> lines(func.trim.mode(absorp, trim = 0.15), col = 3, lty = 3)
R> lines(func.trim.RP(absorp, trim = 0.15), col = 4, lty = 4)
R> plot(func.mean(absorp), main = "Centrality measures (median)",
+ ylim = c(2.6,3.6))
R> lines(func.med.FM(absorp), col = 2)
R> lines(func.med.mode(absorp), col = 3, lty = 3)
R> lines(func.med.RP(absorp), col = 4, lty = 4)
R> plot(func.var(absorp), main = "Dispersion measures", ylim = c(.07,.31))
R> lines(func.trimvar.FM(absorp, trim = 0.15), col = 2)
R> lines(func.trimvar.mode(absorp, trim = 0.15), col = 3, lty = 3)
```

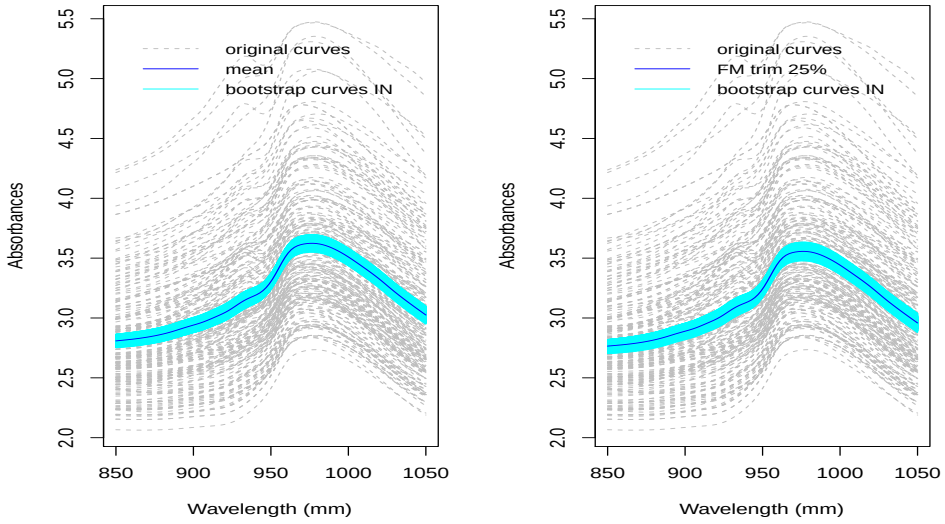

Figure 6: Bootstrap replications of spectrometric curves: Using the mean statistic (left) and using the 25%-trimmed mean statistic with FM depth (right).

```
R> lines(func.trimvar.RP(absorp, trim = 0.15), col = 4, lty = 4)
R> out.FM = depth.FM(absorp, trim = 0.1, draw = FALSE)
R> out.mode = depth.mode(absorp, trim = 0.1, draw = FALSE)
R> plot(out.mode$dep, out.FM$dep, main = "FM depth vs mode depth",
+     xlab = "mode depth", ylab = "FM depth")
```

In the previous code we have employed some shortcut functions as follows:

- For central trend: $func.\{centr\}.\{depth\}$,
- For dispersion measures (or marginal variability measures): $func.trimvar.\{depth\}$

where $centr = \{med, trim\}$ indicates whether it is used the median or trimmed mean and $depth = \{FM, mode, RP, RPD\}$ indicates the type of depth used.

2.5 Bootstrap replications as dispersion measures

The dispersion of a location statistic for functional data can be estimated by smoothed bootstrap, [Cuevas *et al.* (2006)]. The **fdata.bootstrap()** function allows us to define a statistic calculated on the **nb** resamples, control the degree of smoothing by **smo** argument and represent the confidence bands with level α. The **statistic** used by default is the mean **func.mean()** but also other depth-based functions can be used (see **help(Descriptive)**). The confidence bands are drawn as those resamples which are within a given distance from the estimator, see Figure 6.

```
R> out.boot1=fdata.bootstrap(absorp,statistic=func.mean,nb=1000,draw=TRUE)
R> out.boot2=fdata.bootstrap(absorp,statistic=func.trim.FM,nb=1000,draw=TRUE)
```

2.6 Functional Outlier Detection

In order to identify outliers in functional datasets, [Febrero-Bande *et al.* (2008)] make use on the factthat depth and outlyingness are inverse notions, so that if an outlier is in the dataset, the corresponding curve will have a significantly low depth. Therefore, a way to detect the presence of functional outliers is to look for curves with lower depths. Two procedures for detecting outliers are implemented: the first one is based on weighting **outliers.depth.pond()** and the second one isbased on trimming **outliers.depth.trim()**.

In the following example we use (**data("poblenou")**) that collects 127 curves of NO_x levels measured every hour $\{t_i\}_{0:23}$ by a control station in Poblenou in Barcelona (Spain). This dataset is used by [Febrero-Bande *et al.* (2008)] as an illustration of the outliers detection procedures. Figure 7 shows the result of applying the outliers detection method based on trimming **outliers.depth.trim** with mode depth for the case of working days and non-working days.

```
R> data("poblenou")
R> nox <- poblenou$nox
R> nb = 20
R> dd <- as.integer(poblenou$df$day.week)
R> working = poblenou$nox[poblenou$df$day.festive == 0 & dd < 6]
R> nonworking = poblenou$nox[poblenou$df$day.festive == 1 | dd > 5]
R> out = outliers.depth.trim(nonworking, dfunc = depth.RP, nb = nb,
+     smo = 0.1, trim = 0.06)
R> out2 = outliers.depth.trim(working, dfunc = depth.FM, nb = nb,
+     smo = 0.1, trim = 0.06)
```

3 Functional regression models

A regression model is said to be "functional" when at least one of the involved variables (either a regressor variable or the output variable) is functional. This section is devoted to all the functional regression models where the response variable is scalar and at least, there is one functional covariate. For illustration, we will use the Tecator dataset to predict the fat contents from the absorbance as a functional covariate $X(t)$=**X** and the water contents as a non functional covariate (**Z=Water**). We will use the first 129 curves to fit the model. The last 86 records will be used to check the predictions. The explanatory variables to introduce in the models are: The curves of absorbance **X** as functional data or one of its two first derivatives (**X.d1,X.d2**) and/or water content (**Water**) as non functional variable.

```
R> ind = 1:129
R> tt = absorp[["argvals"]]
R> y = tecator$y$Fat[ind]
R> X = absorp[ind, ]
R> X.d1 = fdata.deriv(X, nbasis = 19, nderiv = 1)
R> X.d2 = fdata.deriv(X, nbasis = 19, nderiv = 2)
```

In the following sections, regression methods implemented in the package are presented and illustrated with examples for estimating the **Fat** content of the Tecator dataset. Finally, we show in Section 3.6 a summary of the prediction methods.

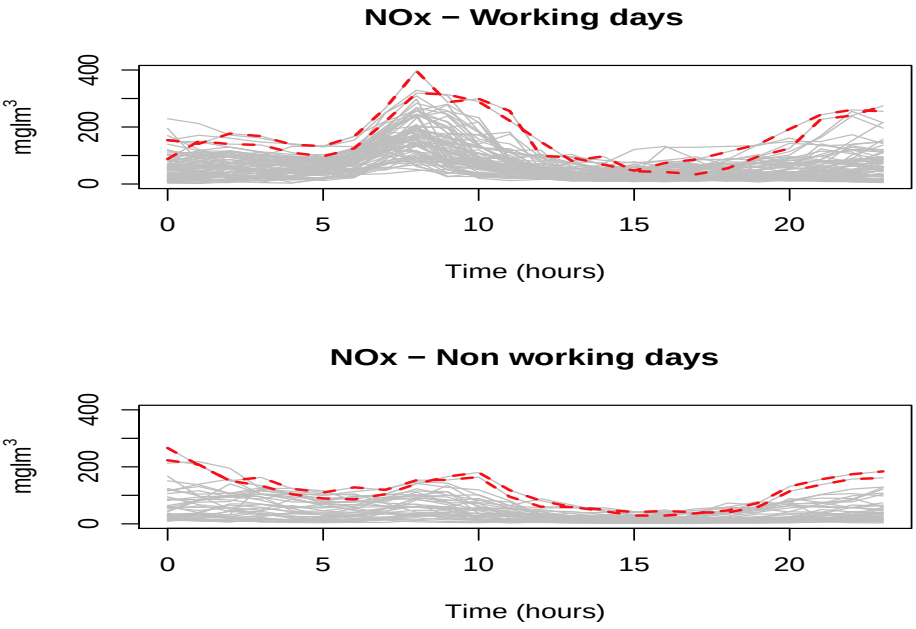

Figure 7: NO_x levels measured by a control station split into two groups (grey lines): Working days (top) and non-working days (bottom). The red lines correspond to days detected as otuliers. The considered periods are 03/18/2005 and 04/28/2005 for working days and 03/19/2005 and 04/30/2005 for non-working days.

3.1 Functional linear model with basis representation: fregre.basis()

In this section we will assume a functional linear model of type:

$$y_i = \langle X, \beta \rangle + \epsilon_i = \int_T X_i(t)\beta(t)dt + \epsilon_i \tag{4}$$

where $\langle \cdot, \cdot \rangle$ denotes the inner product on $\mathcal{L}_2$ and ϵ_i are random errors with mean zero and finite variance σ^2.
[Ramsay and Silverman (2005)] model the relationship between the scalar response and the functional covariate by basis representation of the observed functional data $X(t)$ and the unknown functional parameter $\beta(t)$. The functional linear model in Equation 4 is estimated by the expression:

$$\hat{y}_i = \int_T X_i(t)\beta(t)dt \approx \mathbf{C_i^T}\psi(\mathbf{t})\phi^{\mathbf{T}}(\mathbf{t})\hat{\mathbf{b}} = \tilde{\mathbf{X}}\hat{\mathbf{b}} \tag{5}$$

where $\tilde{\mathbf{X}}_\mathbf{i}(\mathbf{t}) = \mathbf{C_i^T}\psi(\mathbf{t})\phi^{\mathbf{T}}(\mathbf{t})$, and $\hat{\mathbf{b}} = (\tilde{\mathbf{X}}^{\mathbf{T}}\tilde{\mathbf{X}})^{-\mathbf{1}}\tilde{\mathbf{X}}^{\mathbf{T}}y$ and so, $\hat{y} = \tilde{\mathbf{X}}\hat{\mathbf{b}} = \tilde{\mathbf{X}}(\tilde{\mathbf{X}}^{\mathbf{T}}\tilde{\mathbf{X}})^{-\mathbf{1}}\tilde{\mathbf{X}}^{\mathbf{T}}y = \mathbf{H}y$ where $\mathbf{H}$ is the hat matrix with degrees of freedom: $df = trace(\mathbf{H})$.

If we want to incorporate a roughness penalty $\lambda > 0$ then the above expression is now:
$\hat{y} = \tilde{\mathbf{X}}\hat{\mathbf{b}} = \tilde{\mathbf{X}}(\tilde{\mathbf{X}}^{\mathbf{T}}\tilde{\mathbf{X}} + \lambda\mathbf{R_0})^{-\mathbf{1}}\tilde{\mathbf{X}}^{\mathbf{T}}y = \mathbf{H}y$ where $\mathbf{R}_0$ is the penalty matrix.
fregre.basis() function computes functional regression between functional explanatory variable and scalar response using basis representation. This function is presented as an alternative to the function **fRegress()** of **fda** package because it allows covariates of class **fdata**

and other class format as **matrix** or **data.frame**. The function also gives default values to arguments **basis.x** and **basis.b** for representation on the basis of functional data $X(t)$ and the functional parameter $\beta(t)$, respectively. In addition, the function **fregre.basis.cv()** uses validation criteria defined in Section 2.2 by argument **type.CV** to estimate the number of basis elements or the penalized parametrer (λ) that best predicts the response.
Going on with the example, the next code illustrates how to estimate the fat contents (**y=Fat**) using a training sample of absorbances curves **X**.

```
R> rangett <- absorp[ind, ][[‘‘rangeval’’]]
R> basis1 = create.bspline.basis(rangeval = rangett, nbasis = 17)
R> basis2 = create.bspline.basis(rangeval = rangett, nbasis = 7)
R> res.basis0=fregre.basis(X, y, basis.x = basis1, basis.b = basis2)

-Call: fregre.basis(fdataobj = X, y = y, basis.x = basis1, basis.b = basis2)

-Coefficients:
(Intercept)    X.bspl4.1    X.bspl4.2    X.bspl4.3    X.bspl4.4    X.bspl4.5
      18.24        47.89       -53.79        31.75       -15.48        16.21
  X.bspl4.6    X.bspl4.7
     -20.50        18.76

-R squared:  0.9367377
-Residual variance:  10.72561
```

The fitted object contains useful information as the smoothing parameter (ν), the degrees of freedom (df), the residual variance (S_R^2): $S_R^2 = \sum_{i=1}^{n} (y_i - \hat{y}_i)^2 / (n - df)$ (an unbiased estimator of σ^2), the coefficient of determination (R^2) or the hat matrix (**H**). This information is used in **summary.fregre.fd()** and **print.fregre.fd()** to show summaries of the functional regression fitted model, see Figure 8.

```
R> summary(res.basis0)
 *** Summary Functional Data Regression with representation in Basis ***

Call:
fregre.basis(fdataobj = X, y = y, basis.x = basis1, basis.b = basis2)

Residuals:
     Min       1Q   Median       3Q      Max
-10.5079  -1.8778   0.0192   2.5561   5.5249

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)  18.2357     0.2883  63.242  < 2e-16 ***
X.bspl4.1    47.8937     4.7485  10.086  < 2e-16 ***
X.bspl4.2   -53.7862     6.5834  -8.170 3.43e-13 ***
X.bspl4.3    31.7550     7.8395   4.051 9.06e-05 ***
X.bspl4.4   -15.4777     7.1935  -2.152   0.0334 *
X.bspl4.5    16.2149     9.5483   1.698   0.0920 .
```

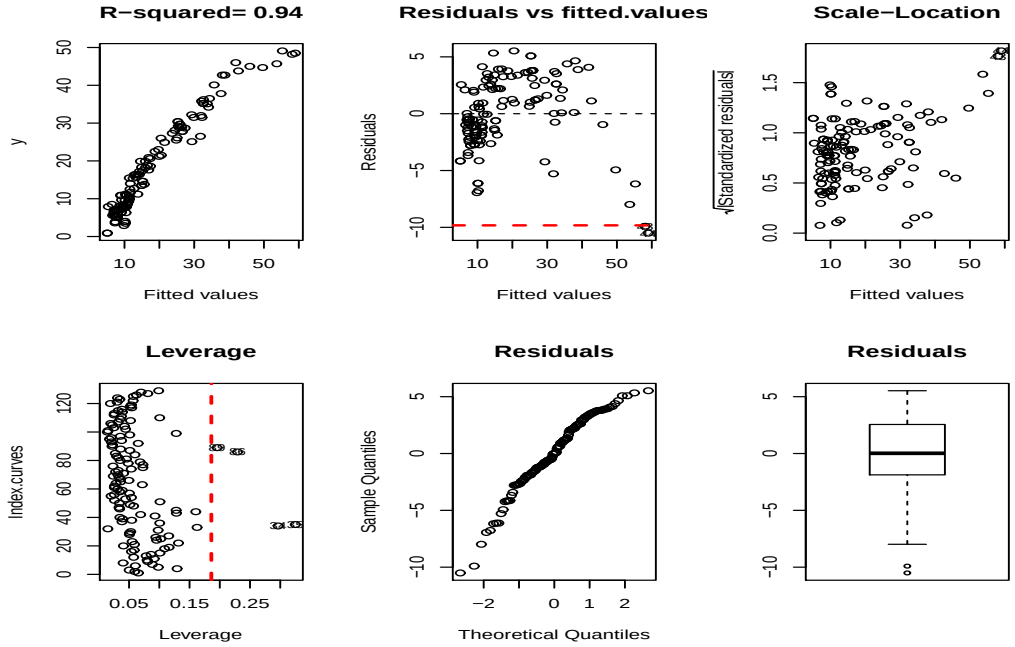

Figure 8: Summary plots for fitted object (**res.basis0**) from a functional linear model with basis representation (**fregre.basis()**).

```
X.bspl4.6   -20.5031    11.5959  -1.768   0.0796 .
X.bspl4.7    18.7593     9.2149   2.036   0.0440 *
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 3.275 on 121 degrees of freedom
Multiple R-squared: 0.9367,Adjusted R-squared: 0.9331
F-statistic:   256 on 7 and 121 DF,  p-value: < 2.2e-16

-Names of possible atypical curves: 43 44
-Names of possible influence curves: 34 35 86 89
```

3.2 Functional linear model with functional principal components basis: fregre.pc()

Similarly, [Cardot *et al.* (1999)] used a basis of functional principal components to represent the functional data $X(t)$ and the functional parameter $\beta(t)$ in the so-called functional principal components regression (FPCR).
Now, the estimation of β can be made by a few principal components (PC) of the functional data and the integral can be aproximated by:

$$\hat{y}_i = \int_T X_i(t)\beta(t)dt \approx \sum_{k=1}^{k_n} \gamma_{ik_n}\hat{\beta}_{k_n} \tag{6}$$

where, $\hat{\beta}_{(1:k_n)} = \left(\frac{\gamma_{.1}^T y}{n\lambda_1}, \ldots, \frac{\gamma_{.k_n}^T y}{n\lambda_{k_n}}\right)$ and $\gamma_{(1:k_n)}$ is the $(n \text{x} k_n)$ matrix with k_n principal components estimation of β scores and λ_i the eigenvalues of the PC.

The model of Equation 6 is expressed as: $\hat{y} = \mathbf{H}y$
where $\mathbf{H} = \left(\frac{\gamma_{.1}\gamma_{.1}^T y}{n\lambda_1}, \ldots, \frac{\gamma_{.k_n}\gamma_{.k_n}^T y}{n\lambda_{k_n}}\right)$ with degrees of freedom: $df = trace(\mathbf{H}) = k_n$.

We have implemented the functional principal regression in the function **fregre.pc()**. The call for Tecator example is shown below:

```
R> res.pc0=fregre.pc(X,y,l=1:6)
```

For the fitted object of **fregre.pc()** function, the function **summary.fregre.fd()** also shows:

- Variability of explicative variables explained by Principal Components.

- Variability for each principal components -PC-.

3.2.1 How to select k_n: fregre.pc.cv()

The novelty of the procedure used in **fregre.pc.cv()** is that the algorithm selects the principal components that best estimated the response. The selection is done by cross-validation (CV) or Model Selection Criteria (MSC). Finally, the regression model is fitted using the best selection of Functional Principal Components.

- Predictive Cross-Validation: $PCV(k_n) = \frac{1}{n}\sum_{i=1}^{n}\left(y_i - \left\langle X_i, \hat{\beta}_{(-i,k_n)}\right\rangle\right)^2$,
  criteria="CV"

- Model Selection Criteria: $MSC(k_n) = log\left[\frac{1}{n}\sum_{i=1}^{n}\left(y_i - \left\langle X_i, \hat{\beta}_{(i,k_n)}\right\rangle\right)^2\right] + p_n\frac{k}{n}$
  $p_n = 2$, **criteria**="AIC"
  $p_n = \frac{2n}{n-k_n-2}$, **criteria**="AICc"
  $p_n = \frac{log(n)}{n}$, **criteria**="SIC"
  $p_n = \frac{log(n)}{n-k_n-2}$, **criteria**="SICc"

where **criteria** is an argument of the function **fregre.pc.cv()** that controls the type of validation used in the selection of the smoothing parameter k_n.

```
R> res.pc2 = fregre.pc.cv(X.d2, y, kmax = 8)
R> res.pc2$pc.opt
Var1 Var2 Var3 Var4 Var5
   1    2    7    3    6
R> res.pc2$MSC
     [,1]     [,2]     [,3]     [,4]     [,5]     [,6]     [,7]     [,8]
[1,] 2.874226 2.395084 2.140198 2.138155 2.136051 2.147605 2.184047 2.221696
attr(,"names")
[1] "PC1" "PC2" "PC7" "PC3" "PC6" "PC8" "PC4" "PC5"
```

3.2.2 Functional influence measures: influence.fdata()

This section focuses on how to identify influential observations in the FLM discussed in the previous sections. [Febrero-Bande *et al.* (2010)] studied three statistics for measuring the influence: Cook Prediction Distance (CP_i), Cook Estimation Distance (CE_i) and Peña Distance (P_i), respectively.

1. The functional Cook's measure for prediction (CP_i) allows to detect observations whose deletion may entail important changes in the prediction of the rest of the data. It is defined as follows:

$$CP_i = \frac{(\hat{y} - \hat{y}_{-i})^T(\hat{y} - \hat{y}_{-i})}{S_R^2}$$

   where $\hat{y}_{-i}$ is the prediction of the response y excluding the i-th observation (X_i, y_i) in the estimation.

2. The functional Cook's measure for estimation (CE_i) allows to detect observations whose deletion may entail important changes in the estimation.

$$CE_i = \frac{\left\| \hat{\beta} - \hat{\beta}_{-i} \right\|^2}{\frac{S_R^2}{n} \sum_{k=1}^{k_x} \frac{1}{\lambda_k}}$$

   where $\hat{\beta}_{-i}$ is the estimator of the parameter β excluding the i-th observation (X_i, y_i) in the process.

3. The functional Peña's measure for prediction (P_i) allows to detect observations whose prediction is most affected by the deletion of other data.

$$P_i = \frac{(\hat{y}_i - \hat{y}_{(-1,i)}, ..., \hat{y}_i - \hat{y}_{(-n,i)})^T(\hat{y}_i - \hat{y}_{(-1,i)}, ..., \hat{y}_i - \hat{y}_{(-n,i)})}{S_R^2 H_{ii}}$$

   where $\hat{y}_{(-h,i)}$ is the i-th component of the prediction vector $\hat{\mathbf{y}}_{(-h)}$ for $h = 1, ..., n$.

Once estimated the functional regression model with scalar response (by **fregre.pc()** or **fregre.basis()**), the **influence.fdata()** function is used to obtain the influence measures, see Figure 9.
[Febrero-Bande *et al.* (2010)] propose to approximate quantiles of the above statistics by means of a procedure which uses smoothed bootstrap samples of the set of observations of the above statistics. This package includes the above procedure in **influence.quan()** function. When the goal is to make inferences about the functional parameter and not on influence measures, one advantag derived from this procedure is that the calculation takes the **mue.boot** curves of the functional parameter β in the functional Cook's measure for estimation. However, this procedure has a very high computation time. In order to reduce the computational time we have created the **fregre.bootstrap()** function.

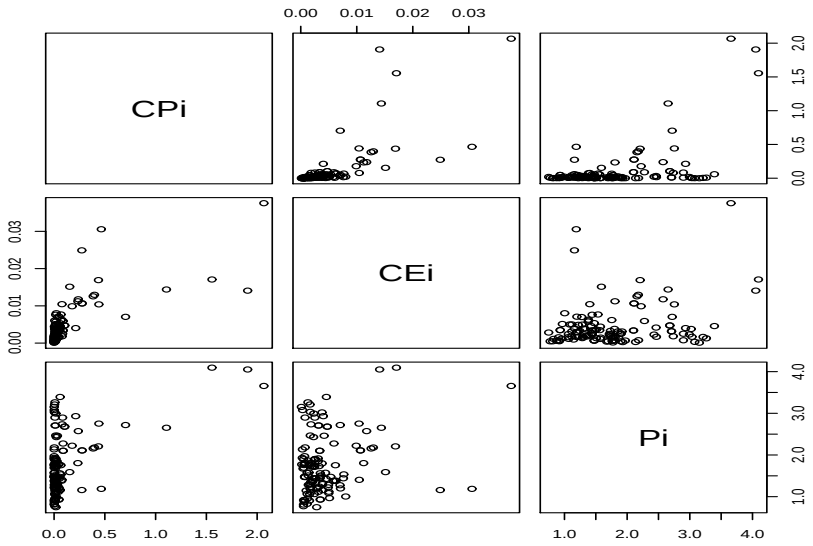

Figure 9: Influence measures for Tecator dataset calculated from fitted model (**res.pc0**): Matrix of scatterplots for Distance Cook Prediction (CP_i), Distance Cook Estimation (CE_i) and Peña Distance (P_i).

```
R>fregre.bootstrap(res.basis1,nb=100,kmax.fix=TRUE,alpha=.99)
R>fregre.bootstrap(res.pc0,nb=100,kmax.fix=TRUE,alpha=.99)
```

In Figure 10 are plotting the bootstrap confidence band for the $\hat{\beta}$ (blue line) with level of $(1-\alpha)\%$. For the two fitted models (basis and PC), with 100 $\hat{\beta}^*$ curves, 99 curves belonging to the confidence band are drawn in gray and the curve fell outside the band is drawn in red. The effect of β is significative (different from 0) for different wavelength, although these values are different depending on model.

```
R> res.infl=influence.fdata(res.pc0)
R> res.quan<-influence.quan(res.pc0,res.infl,mue.boot=500,kmax.fix=TRUE)
R> plot(res.quan$betas.boot,col="grey")
R> lines(res.pc0$beta.est,col=2,lwd=2)
```

3.3 Functional linear model with functional and non functional covariate: fregre.lm()

This section is presented as an extension of the previous linear regression models. Now, the scalar response Y is estimated by more than one functional covariate $X^j(t)$ and also more than one non functional covariate Z^j. The regression model is given by:

$$y_i = \alpha + \beta_1 Z_i^1 + \cdots + \beta_p Z_i^p + \int_{T_1} X_i^1(t)\beta_1(t)dt + \cdots + \int_{T_f} X_i^q(t)\beta_q(t)dt + \epsilon_i \qquad (7)$$

where $Z = \left[Z^1, \cdots, Z^p\right]$ are the non functional covariates and $X(t) = \left[X^1(t_1), \cdots, X^q(t_q)\right]$ are the functional covariates.

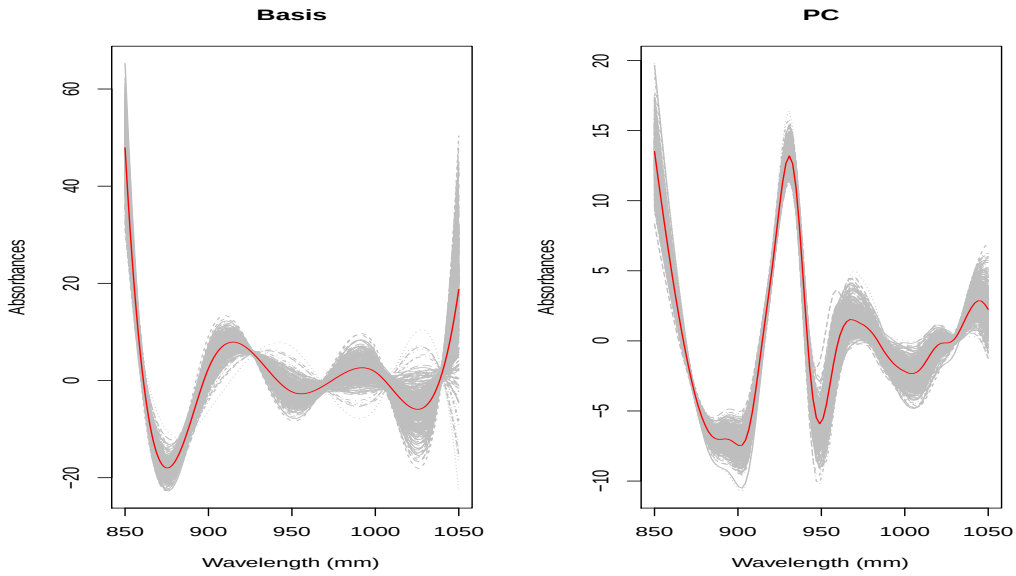

Figure 10: Estimated regression function $\hat{\beta}$ joint with a 99% bootstrap confidence band by: fregre.basis (left) and fregre.pc (rigth).

The functional linear model 7 is estimated by the expression:

$$\hat{y} = \tilde{\mathbf{X}}\mathbf{b} = \tilde{\mathbf{X}}(\tilde{\mathbf{X}}^{\mathbf{T}}\tilde{\mathbf{X}})^{-1}\tilde{\mathbf{X}}^{\mathbf{T}}y = \mathbf{H}y$$

where $\tilde{X} = \left[Z^1, \cdots, Z^p, (\mathbf{C^1})^{\mathbf{T}}\psi(\mathbf{t_1})\phi^{\mathbf{T}}(\mathbf{t_1}), \cdots, (\mathbf{C^q})^{\mathbf{T}}\psi(\mathbf{t_q})\phi^{\mathbf{T}}(\mathbf{t_q})\right]$
The arguments are as follows:

- **formula**: A symbolic description of the model to be fitted.
- **data**: List containing the variables in the model. The first item in the data list is a *"data.frame"* called **df** with the response and non functional explanatory covariates. Functional covariates (*"fdata"* or *"fd"* class) are introduced in the following items in the data list.
- **basis.x**: List with a basis object for every functional covariate.
- **basis.b**: List with a basis object for estimating the functional parameter β.

For the Tecator data example, the content of **Fat** is estimated from the second derivative of absorbances curves **X.d2** and the content of **Water** by **fregre.lm()** function.

```
R> ind <- 1:129
R> dataf = as.data.frame(tecator$y[ind, ])
R> newdataf = as.data.frame(tecator$y[-ind, ])
R> ldata = list(df = dataf, X = X, X.d1 = X.d1, X.d2 = X.d2)
R> f2 = Fat ~ Water + X.d2
R> basis.x1 = list(X.d2 = basis1)
R> basis.b1 = list(X.d2 = basis2)
R> res.lm2 = fregre.lm(f2, ldata, basis.x = basis.x1, basis.b = basis.b1)
[1] "Non functional covariate: Water"
[1] "Functional covariate: X.d2"
```

3.4 Nonparametric functional regression model: fregre.np()

An alternative to model of Equation 4 is the nonparametric functional regression studied by [Ferraty and Vieu (2006)]. In this case, the regression model is written as

$$y_i = r(X_i(t)) + \epsilon_i, \tag{8}$$

where the unknown smooth real function r is estimated using kernel estimation by means of

$$\hat{r}(X) = \frac{\sum_{i=1}^{n} K(h^{-1}d(X, X_i))y_i}{\sum_{i=1}^{n} K(h^{-1}d(X, X_i))}$$

where K is an asymmetric kernel function, h is the smoothing parameter and d is a metric or a semimetric.
This procedure is implemented in the **fregre.np()** function and some of its arguments are:

- **Ker**: Type of asymmetric kernel function, by default asymmetric normal kernel.
- **metric**: Type of metric or semimetric, by default $\mathcal{L}_2$ (**metric.lp(...,p=2)**).
- **type.S**: Type of smoothing matrix S, by default Nadaraya Watson (**S.NW()**).

Again, the function **fregre.np.cv()** is used to choose the smoothing parameter h by the validation criteria described in Section 2.2.

- **type.CV**: Type of validation criterion, by default GCV criterion (**GCV.S()**).

The code for the Tecator example is:

```
R> fregre.np(X, y, metric = semimetric.deriv, nderiv = 1)
-Call: fregre.np(fdataobj = X, y = y, metric = semimetric.deriv, nderiv = 1)

-Bandwidth (h):  0.07296407
-R squared:  0.985607
-Residual variance:  3.810363
```

3.5 Semi-functional partially linear model (SFPLM): fregre.plm()

An extension of the nonparametric functional regression models is the semi-functional partial linear model proposed in [Aneiros-Pérez and Vieu (2006)]. This model uses a non-parametric kernel procedure as that described in Section 3.4. The output y is scalar. A functional covariate $X(t)$ and a multivariate non functional covariate Z are considered.

$$y = r(X(t)) + \sum_{j=1}^{p} Z_j\beta_j + \epsilon \tag{9}$$

The unknown smooth real function r is estimated by means of

$$\hat{r}_h(t) = \sum_{i=1}^{n} w_{n,h}(t, X_i)(Y_i - Z_i^T\hat{\beta}_h)$$

where W_h is a weight function: $w_{n,h}(t, X_i) = \frac{K(d(t,X_i)/h)}{\sum_{j=1}^{n} K(d(t,X_j)/h)}$ with smoothing parameter h, an asymmetric kernel K and a metric or semimetric d. In **fregre.plm()** by default W_h is a functional version of the Nadaraya-Watson-type weights (**type.S=S.NW**) with asymmetric normal kernel (**Ker=AKer.norm**) in $\mathcal{L}_2$ (**metric=metric.lp** with **p=2**). The unknown parameters β_j for the multivariate non functional covariates are estimated by means of $\hat{\beta}_j = (\tilde{X}_h^T \tilde{X}_h)^{-1} \tilde{X}_h^T \tilde{X}_h$ where $\tilde{X}_h = (I - W_h)X$ with the smoothing parameter h and the identity matrix I. The errors ϵ are independent, with zero mean, finite variance σ^2 and $E[\epsilon|Z_1, \ldots, Z_p, X(t)] = 0$.
Coming back to the example of Section 3.3, the fitted model for the case of a real variable **Z=Water** and the second derivative of the absorbance curves (**X.d2**) as functional covariate can be obtained by:

```
R> fregre.plm(f2, ldata, Ker = AKer.epa, type.S = S.KNN)
-Call: fregre.plm(formula = f2, data = ldata, Ker = AKer.epa, type.S = S.KNN)

-Coefficients:
       Estimate    Std. Error  t value     Pr(>|t|)
Water  -9.634e-01   3.748e-02  -2.571e+01   3.018e-24

-Bandwidth (h):  9
-R squared:  0.994232
-Residual variance:  1.256837
```

3.6 Prediction methods for functional regression model fits: fregre.fd(), fregre.lm() and fregre.plm()

Once the model is estimated we can obtain predictions model object by means of:

- **predict.fregre.fd()** corresponds to the model fitted from the functions **fregre.pc()**, **fregre.np()** or **fregre.basis()**.
- **predict.fregre.lm()** corresponds to the model fitted from the function **fregre.lm()**.
- **predict.fregre.plm()** corresponds to the model fitted from the function **fregre.plm()**.

A sample test of the last 86 curves of absorbances (or one of the two first derivatives) can be used to predict the **Fat** content, as follows:

```
R> newy = matrix(tecator$y$Fat[-ind], ncol = 1)
R> newX = absorp[-ind, ]
R> newX.d1 = fdata.deriv(absorp[-ind, ], nbasis = 19, nderiv = 1)
R> newX.d2 = fdata.deriv(absorp[-ind, ], nbasis = 19, nderiv = 2)
R> res.basis2 = fregre.basis.cv(X.d2, y, type.basis = "fourier")
R> pred.basis2 = predict.fregre.fd(res.basis2, newX.d2)
R> res.pc1 = fregre.pc.cv(X.d1, y, 8)$fregre.pc
R> pred.pc1 = predict.fregre.fd(res.pc1, newX.d1)
R> res.np2 = fregre.np.cv(X.d2, y, metric = semimetric.fourier)
R> pred.np2 = predict(res.np2, newX.d2)
```

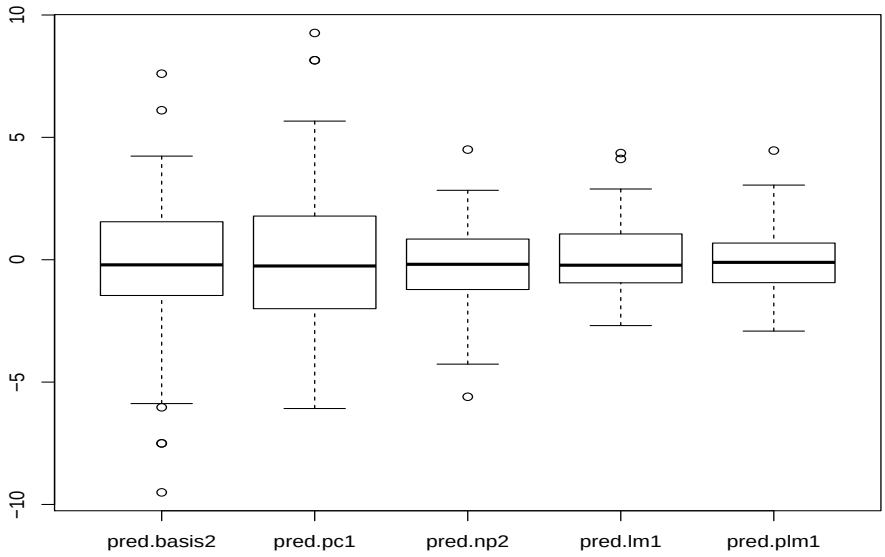

Figure 11: Boxplot of predicted residuals: **fregre.basis()**, **fregre.pc()**, **fregre.np()**, **fregre.lm()** and **fregre.plm()** (left to right).

We provide below the complete code for the best prediction of the fat content with the procedures **fregre.lm()** and **fregre.plm()** using the non functional covariate $Water$.

```
R> newldata = list(df = newdataf, X = newX, X.d1 = newX.d1, X.d2 = newX.d2)
R> f1 = Fat ~ Water + X.d1
R> basis.x1 = list(X.d1 = basis1)
R> basis.b1 = list(X.d1 = basis2)
R> res.lm1 = fregre.lm(f1, ldata, basis.x = basis.x1, basis.b = basis.b1)
[1] "Non functional covariate: Water"
[1] "Functional covariate: X.d1"
R> pred.lm1 = predict.fregre.lm(res.lm1, newldata)
R> res.plm1 = fregre.plm(f1, ldata, Ker = AKer.tri, type.S = S.KNN)
R> pred.plm1 = predict.fregre.plm(res.plm1, newldata)
```

The boxplots of predicted residuals for each procedure are shown in Figure 11.
We make predictions for the rest of models fitted in this paper (code not shown). Following the ideas by [Aneiros-Pérez and Vieu (2006)], we calculated the mean square error of prediction (MEP): $MEP = \left(\sum_{i=1}^{n} \left(y_i - \hat{y}_i\right)^2 / n\right) / \left(Var\left(y\right)\right)$, which is used for comparing the predictions of different fitted models. Table 1 resumes the statistics of the fitted models and their predictions.

Function	*df*	R^2	S^2_R	MEP
fregre.basis(X,Fat)	8	0.937	10.726	0.0544
fregre.basis.cv(X.d2,Fat)	12	0.962	6.613	0.0485
fregre.pc(X.d1,Fat)	7	0.947	8.928	0.0502
fregre.pc(X.d2,Fat)	7	0.943	9.626	0.0521
fregre.lm(Fat X.d1+Water)	9	0.987	2.149	0.0096
fregre.lm(Fat X.d2+Water)	7	0.986	2.412	0.0119
fregre.np(X.d1,Fat)	39.4	0.981	4.239	0.0287
fregre.np(X.d2,Fat)	39.9	0.990	2.361	0.0243
fregre.plm(Fat X+Water)	20.0	0.984	3.049	0.0178
fregre.plm(Fat X.d1+Water)	38.5	0.994	1.314	0.0093
fregre.plm(Fat X.d2+Water)	36.3	0.995	1.107	0.0114

Table 1: Results for functional regression models. *df* degrees of freedom, S^2_R residual variance, R^2 R-squared and MEP mean square error of prediction.

4 Conclusion

The package **fda.usc** presented in this paper is the result of the integration of our codes with other procedures from different authors, such as the package **fda** or the functions from **STAPH** group.
One major advantage of this software is to avoid the need of a basis representation of functional data. Using the new class **fdata**, the proposed methods can represent the functional data using discretized versions in a given grid of points.
This package includes most of the methods recently developed for Exploratory Functional Data Analysis and for Functional Regression with scalar response.
The **fda.usc** package also incorporates other utilities for statistical computing within the field of Functional Data Analysis (FDA). Some useful additions are:

- Functional Generalized Linear Models (FGLM): **fregre.glm().**
- Functional Generalized Additive Models (FGAM): **fregre.gsam()** and **fregre.kgam()**.
- Functional ANOVA: **anova.RPm().**
- Functional Supervised Classification : **classif.kernel.fd().**
- Functional Non-Supervised Classification : **kmeans.fd().**
- Other utilities and auxiliary functions, as the **cond.F()** function that calculates the conditional distribution function of a scalar response with functional data.

Finally, the **fda.usc** package is an attempt to get an integrated framework for FDA. It is under continuous development therefore updates will be available in CRAN (see the NEWS file). Further information is also available at the project website whose URL is given in the DESCRIPTION file.

Acknowledgments

This work was supported by grants MTM2008-03010 from the Ministerio de Ciencia e Innovación, 10MDS207015PR from the Xunta de Galicia and GI-1914 MODESTYA-Modelos de optimización, decisión, estadística y aplicaciones.

References

[Aneiros-Pérez and Vieu (2006)] Semi-Functional Partial Linear Regression. *Statist. Probab. Lett.*, **76**(11), 1102–1110. ISSN 0167-7152.

[Cardot *et al.* (1999)] Functional Linear Model. *Statist. Probab. Lett.*, **45**(1), 11–22.

[Cardot *et al.* (2003)] Spline Estimators for the Functional Linear Model. *Statistica Sinica,*, **13**, 571–591.

[Crainiceanu and Goldsmith (2010)] Bayesian functional data analysis using winbugs *Journal of Statistical Soft*, **32**(11).

[Cuesta *et al.* (2010)] A simple multiway ANOVA for functional data. *Test.*, **19**(3), 537–557.

[Cuevas *et al.* (2006)] On the Use of the Bootstrap for Estimating Functions with Functional Data. *Comput. Statist. Data Anal.*, **51**(2), 1063–1074.

[Cuevas *et al.* (2007)] Robust Estimation and Classification for Functional Data via Projection-Based Depth Notions. *Comput. Statist.*, **22**(3), 481–496.

[Escabias *et al.* (2005)] Modeling environmental data by functional principal component logistic regression. *CEnvironmetrics*, **16**(1), 95–107.

[Escabias *et al.* (2007)] Functional PLS logit regression. *Computational Statistics and Data Analysis*, **51**, 4891–4902.

[Febrero-Bande *et al.* (2010)] Measures of Influence for the Functional Linear Model with Scalar Response. *J. Multivariate Anal.*, **101**(2), 327–339.

[Febrero-Bande *et al.* (2008)] Outlier Detection in Functional Data by Depth Measures, with Application to Identify Abnormal NO_x Levels. *Environmetrics*, **19**(4), 331–345.

[Ferraty and Vieu (2006)] *Nonparametric Functional Data Analysis*. Springer Series in Statistics. Springer-Velag, New York. Theory and practice.

[Fraiman and Muniz (2001)] Trimmed Means for Functional Data. *Test*, **10**(2), 419–440.

[Goldsmith *et al.* (2011)] Penalized Functional Regression. *Journal of Computational and Graphical Statistics.*

[Härdle (1990)] Härdle W (1990). *Applied Nonparametric Regression*, volume 19 of *Econometric Society Monographs*. Cambridge University Press, Cambridge.

[Hyndman and Shang (2010a)] ***fds**: Functional Data Sets.* **R** package version 1.6., http://cran.r-project.org/package=fds.

[Hyndman and Shang (2010b)] ***ftsa**: Functional Time Series Analysis.* **R** package version 2.6., http://cran.r-project.org/package=ftsa.

[Martens and Naes (1989)] Multivariate calibration. New York: Wiley

[McCullagh and Nelder (1989)] Generalized Linear Models. Second ed. London: Chapman and Hall

[Morgan and Smith (1992)] P A note on Wadleys problem with overdispersion. *Applied Statistics*, **41**, 349–354.

[Müller and Stadtmüller (2005)] Generalized functional linear models. *Ann. Statist.*., **33**, 774–805.

[Mevik and Wehrens (2007)] *The **pls** Package: Principal Component and Partial Least Squares Regression in **R**.* **R** package version 2.1.0., http://cran.r-project.org/package=pls. *Journal of Statistical Software*, **18**(2), 1–24.

[Preda *et al.* (2007)] PLS classification of functional data. *Comput. Stat*, **22**(2), 223–235.

[**R** Development Core Team(2011)] **R** Development Core Team (2011). ***R**: A Language and Environment for Statistical Computing.* **R** Foundation for Statistical Computing, Vienna, Austria. ISBN 3-900051-00-3. http://www.R-project.org/.

[Ramsay and Silverman (2002)] *Applied Functional Data Analysis: Methods and case studies.* Springer Series in Statistics. Springer-Verlag, New York.

[Ramsay and Silverman (2005)] *Functional Data Analysis.* Springer Series in Statistics, second edition. Springer-Velag, New York.

[Ramsay *et al.* (2010)] ***fda**: Functional Data Analysis.* **R** package version 2.2.6., http://cran.r-project.org/package=fda.

[Reis *et al.* (2010)] ***refund**: Regression with Functional Data.* **R** package version 0.1.3., http://cran.r-project.org/package=refund.

[Shang and Hyndman (2010)] ***Rainbow**: Rainbow Plots, Bagplots and Boxplots for Functional Data.* **R** package version 2.3.4., http://cran.r-project.org/package=rainbow.

[Venables and Ripley (2002)] Modern applied statistics with S. New York: Springer-Verlag.

[Wasserman (2006)] *All of Nonparametric Statistics.* Springer Texts in Statistics. Springer-Velag, New York.

Reports in Statistics and Operations Research

2005

05-01 SiZer Map for Evaluating a Bootstrap Local Bandwidth Selector in Nonparametric Additive Models. M. D. Martínez-Miranda, R. Raya-Miranda, W. González-Manteiga and A. González-Carmona.

05-02 The Role of Commitment in Repeated Games. I. García Jurado, Julio González Díaz.

05-03 Project Games. A. Estévez Fernández, P. Borm, H. Hamers.

05-04 Semiparametric Inference in Generalized Mixed Effects Models. M. J. Lombardía, S. Sperlich.

2006

06-01 A unifying model for contests: effort-prize games. J. González Díaz.

06-02 The Harsanyi paradox and the "right to talk" in bargaining among coalitions. J. J. Vidal Puga.

06-03 A functional analysis of NOx levels: location and scale estimation and outlier detection. M. Febrero, P. Galeano, W. González-Manteiga.

06-04 Comparing spatial dependence structures. R. M. Crujeiras, R. Fernández-Casal, W. González-Manteiga.

06-05 On the spectral simulation of spatial dependence structures. R. M. Crujeiras, R. Fernández-Casal.

06-06 An L_2-test for comparing spatial spectral densities. R. M. Crujeiras, R. Fernández-Casal, W. González-Manteiga.

2007

07-01 Goodness-of-fit tests for the spatial spectral density. R. M. Crujeiras, R. Fernández-Casal, W. González-Manteiga.

07-02 Presmothed estimation with left truncated and right censores data. M. A. Jácome, M. C. Iglesias-Pérez.

07-03 Robust nonparametric estimation with missing data. G. Boente, W. González-Manteiga, A. Pérez-González.

07-04 k-Sample test based on the common area of kernel density estimators. P. Martínez-Camblor, J. de Uña Álvarez, N. Corral-Blanco.

07-05 A bootstrap based model checking for selection-biased data. J. L. Ojeda, W. González-Manteiga, J . A. Cristobal.

07-06 The Gaussian mixture dynamic conditional correlation model: Bayesian estimation, value at risk calculation and portfolio selection. P. Galeano, M. C. Ausín.

2008

08-01 ROC curves in nonparametric location-scale regression models. W. González-Manteiga, J. C. Pardo Fernández, I. Van Keilegom.

08-02 On the estimation of α-convex sets. B. Pateiro-López, A. Rodríguez-Casal.

2009

09-01 Lasso Logistic Regression, GSoft and the Cyclyc Coordinate Descent Algorithm. Application to Gene Expression Data. M. García-Magariños, A. Antoniadis, R. Cao, W. González-Manteiga.

2010

10-01 Asymptotic behaviour of robust estimators in partially linear models with missing responses: The effect of estimating the missing probability on simplified marginal estimators. A. Bianco, G. Boente, W. González-Manteiga, A. Pérez-González.

10-02 First-Price Winner-Takes-All Contents. J. González-Díaz.

10-03 Goodness of Fit Test for Interest Rate Models: an approach based on Empirical Process. A. E. Monsalve-Cobis, W. González-Manteiga, M. Febrero-Bande.

2011

11-01 Exploring wind direction and SO_2 concentration by circular–linear density estimation. E. García–Portugués, R.M. Crujeiras, W. González–Manteiga.

11-02 Utilities for Statistical Computing in Functional Data Analysis: The R Package fda.usc. M. Oviedo de la Fuente, M. Febrero-Bande

Previous issues (2001 – 2003):
http://eio.usc.es/reports.php