roahd Package: Robust Analysis of High Dimensional Data

Ieva, F; Paganoni, A.M.; Romo, J.; Tarabelloni, N.

roahd Package: Robust Analysis of High Dimensional Data

by Francesca Ieva, Anna Maria Paganoni, Juan Romo and Nicholas Tarabelloni

Abstract The focus of this paper is on the open-source **R** package **roahd** (**RO**bust **A**nalysis of **H**igh dimensional **D**ata), see Tarabelloni et al. (2017). **roahd** has been developed to gather recently proposed statistical methods that deal with the robust inferential analysis of univariate and multivariate functional data. In particular, efficient methods for outlier detection and related graphical tools, methods to represent and simulate functional data, as well as inferential tools for testing differences and dependency among families of curves will be discussed, and the associated functions of the package will be described in details.

Introduction

Functional Data Analysis (FDA) has seen an impressive growth in the statistical research due to the more and more frequent production of complex data in many different research contexts (i.e., healthcare, environmental, engineering, etc.). According to the FDA model, data can be seen as measurements of a certain quantity (or a set of quantities) along a given, independent and continuous indexing variable (such as time or space). Observations are then treated as random functions and can be viewed as trajectories of stochastic processes defined on a given infinite dimensional functional space. In this context 'high dimensional data' is meant in this sense: a high number of covariates/predictors (e.g., evaluations of a signal on a given grid) for a single sample unit (e.g., signal). We have to face the traditional 'large p, small n' problem: the number of features can exceed the number of observations. Many research areas deal with this kind of data where features exceed observations, for example, biomedical signals, high resolution imaging, website analysis of stream data.

Even if the research in FDA dates back to 1970s - 1980s, the first edition of Ramsay and Silverman (2005) and Ramsay and Silverman (2002) made the methods available to a larger audience with an enormous impact on the spread of this topic. The authors mainly cover explorative methods, parametric and semi-parametric approaches. Other important books on functional data analysis are Ferraty and Vieu (2006), Horvath and Kokoszka (2012) and Kokoszka and Reimherr (2017). In addition to these monographs there is a vast quantity of scientific papers ranging from theoretical to applied techniques aimed at modelling and analysing functions.

In the open-source **R** software development the number of packages focused on general functional data analysis is rapidly increasing. In particular, **fda** (Ramsay et al. (2014)) presents functions to implement many methods of functional data analysis, including smoothing, plotting and regression models (see Ramsay and Silverman (2005), Ramsay et al. (2009)). The package **fda.usc** (Febrero-Bande and Oviedo de la Fuente (2012)) carries out exploratory and descriptive analysis of functional data such as depth measurements or functional outliers detection, as well as functional regression models (univariate, nonparametric), basis representation and Functional Principal Component Analysis (FPCA). The package **fdasrvf** (Tucker (2017)) performs alignment, FPCA, and modeling of univariate and multivariate functions, allowing for elastic analysis of functional data through phase and amplitude separation. The core of the package **fdapace** (Dai et al. (2018)) is FPCA for sparsely or densely sampled random trajectories and time courses, via the Principal Analysis by Conditional Estimation (PACE) algorithm or numerical integration. The package **rainbow** (Shang and Hyndman (2019)) provides tools for functional data display, explorqatory analysis (plots, bagplots and boxplots) and outlier detection, while the package **fds** (cite(fds) contains 19 data sets with functional data. There are also a lot of other packages, focused on more specific methods for functional data analysis, like regression, classification and clustering, registering and aligning, studying time series of functional data (see Zeileis (2005))

The focus of this paper is on the open-source **R** package **roahd** (**RO**bust **A**nalysis of **H**igh dimensional **D**ata), see Tarabelloni et al. (2017).
roahd has been developed to gather recently proposed statistical methods that deal with the robust statistical analysis of univariate and multivariate functional data. The latter is the case where each observation in a dataset is a set of possibly correlated functions, measured at discrete points. Despite the usefulness of robust statistics methods in data analysis (e.g., median, quantiles, trimmed means), their generalisation to the functional framework is definitely not straightforward, due to the infinite-dimensional nature of the spaces embedding data. A possibility is to leverage on the concept of depth measures in order to create proper order statistics to be used in a suitable robust

inferential framework.

In the multivariate context there are many possible definitions of depth measures. Among others, see Tuckey (1975); Liu and Singh (1993); Liu et al. (1999); Zuo and Serfling (2000). For univariate and multivariate functional data, two main approaches to the generalisation of depth measures have been considered so far. A first approach is to average a multivariate depth, say D_L, defined on $\mathbb{R}^L$ and computed at each point over the domain interval $I \subset \mathbb{R}$ using suitable weights (see Claeskens et al. (2014); Lopez-Pintado et al. (2014) and references therein). Without loss of generality we will refer to I as a compact interval representing the time domain where the (multivariate) functional data are defined. A second approach, that is followed in **roahd**, (see Ieva and Paganoni (2013) and references therein) extends the notion of univariate (i.e., $L = 1$) functional band depth (introduced in López-Pintado and Romo (2007, 2009)) to the multivariate framework (i.e., $L > 1$). In particular, in Ieva and Paganoni (2013) the depth of each component of the multivariate functional data is defined as a measure of the time each curve spends within the envelope generated by any other family of curves. Then the global depth is computed weighing in a suitable way the contributions of each component of the multivariate signal. This approach is based on the intrinsic functional nature of the data since it looks at the position of the entire function with respect to envelopes generated by families of functions.

The main contributions of the package, described and detailed in the following sections, are

(a) the implementation of simple and handy **S3** classes representing functional data (`fData` and `mfData` for the univariate and multivariate case, respectively), as well as a set of algebraic operations and convenience functions to expressively operate on them;

(b) the implementation of useful generators for functional data, such as `generate_gauss_fdata` and `generate_gauss_mfdata`, that can be used to simulate artificial datasets of Gaussian functional data with a target mean and covariance (that must be specified by the researcher as arguments of the related functions), which could be very useful to test or illustrate existing and new methods;

(c) the implementation of efficient functions for computing depth measures and robust statistics, such as `MBD`, `MEI`, `median_mfData`, `cor_spearman`, for both univariate and multivariate functional data, that allow to rank observations from the center of the distribution-outward and down-upward/up-downard with respect to sample measurements;

(d) the implementation of useful graphical methods, like the functional boxplot (`fbplot`) and the outliergram (`outliergram`), that can be employed to carry out an explorative analysis of a functional dataset, and to robustify it by discarding shape, magnitude and covariance outliers.

Robust methods for functional data are generally rather computationally intensive, thus the package's functions have been implemented with an attention to computational efficiency, in order to allow the processing of realistic datasets.

The paper is structured as follows: in Section 2.2.1 we introduce the methods and the functions to represent and to generate functional data; in Section 2.3 we describe the robust statistics and indexes implemented in **roahd**; in Section 2.4 the main graphical tools for outlier detection are detailed, and finally Section 2.5 contains discussion, conclusion and further potential developments.

Representation of Functional Data

The **S3** classes `fData` and `mfData` implement a simple and compact representation of univariate and multivariate functional datasets. They can be used by specifying, for each observation in the functional dataset, a set of measurements over a discrete grid, representing the dependent variable indexing the functional data (e.g., time). If we denote by $I = [t_0, t_1, \ldots, t_{P-1}]$ an evenly spaced grid ($t_j - t_{j-1} = h > 0 \ \forall j = 1, \ldots, P-1$) and imagine to deal with a dataset $D_{i,j} = X_i(t_j)$, $\forall i = 1, \ldots, N$ and $\forall j = 0, \ldots, P-1$, the object `fData` requires the evenly spaced grid over which the functional observations are measured as parameter `grid`, and the values of the observations in the functional dataset, provided in form of a two dimensional data structure (e.g., matrix or array) having as rows the observations and as columns their measurements over the grid of length P `values`. When the constructor of the class is created, it checks that the grid is actually evenly spaced.

An example of the function's call is the following

```
grid <- seq(0, 1, length.out = 100)
values <- matrix(c( sin(2 * pi * grid),
cos(2 * pi * grid),
4 * grid * (1 - grid),
tan(grid),
log(grid)),
nrow = 5, ncol = 100, byrow = TRUE)
fD <- fData(grid, values)
plot(fD, main = 'Univariate FD', xlab = 'time [s]', ylab = 'values', lwd = 2)
```

In particular the number of rows is the sample size, i.e. the number of statistical units and each statistical unit is a function evaluated in the grid of points of length P. In the artificial example above we have 5 curves evaluated on a evenly spaced grid of 100 points. The resulting plot is shown in Figure 1.

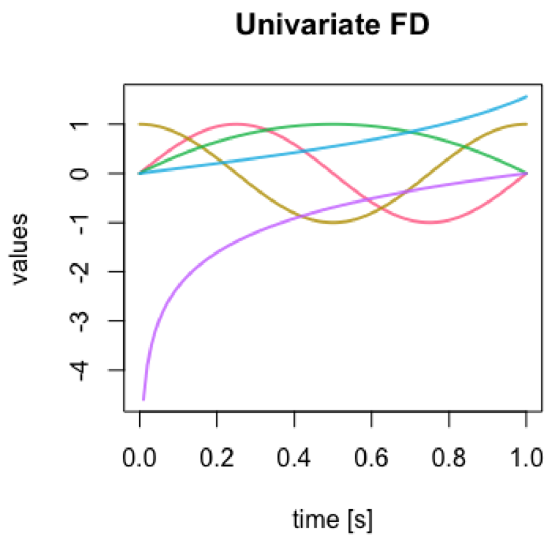

Figure 1: An example of a `fData` object, with 5 curves evaluated on a evenly spaced grid of 100 points.

An `mfData` object, instead, implements a multivariate functional dataset where each component is defined over the same indexing variable. In practice, we deal with a discrete grid $[t_0, t_1, \dots, t_{P-1}]$ and a dataset of N statistical units, each one having L components observed over the same discrete grid: $D_{i,j,k} = X_{i,k}(t_j)$, $\forall i = 1, \dots, N$, $\forall j = 0, \dots, P-1$ and $\forall k = 1, \dots, L$. The object `mfData` requires the evenly spaced grid of definition as parameter `grid` and a list containing the L components of the multivariate functional dataset, defined as 2D data structures (analogously to the constructor of `fData` class).

The **S3** implementation allows to enrich the package with expressive operations that enable an easy manipulation of datasets, keeping at the same time a light structure that allow users to easily access the inner state of objects. For instance, we added an overloaded operator `[.fData` (`[.mfData`) that allows to use standard slices of `matrix` and `array` classes also for `fData` (`mfData`). We provided an overloaded implementation of the four basic algebraic operations, `+.fData`, `-.fData`, `*.fData`, `/.fData`, that allow to write and evaluate simple expressions on `fData` objects without explicitly carry them out on the set of measurements. `+.fData`, `-.fData` support the sum or subtraction of compatible functional datasets (e.g, same definition grid I and same sample size N) or perform the element-wise (along the measurement direction, i.e., columns) sum or subtraction by a one-dimensional vector of compatible measurements (length P). The latter is useful, for instance, in order to translate or center data. The `*.fData`, `/.fData` operators perform an element-wise multiplication or division by a scalar quantity. We also added two convenience functions, `append_fData` and `append_mfData`, that can be used to concatenate two compatible (same definition grid I) functional datasets together.

Statistics functions for the computation of the mean (specification of the generic `mean` function of **R**), the median (through `median_fData` and `median_mfData`, see Section 2.3 for more details), or the covariance (`cov fun` and its specifications for `fData` and `mfData`), are also implemented for `fData` and `mfData`.

Finally, we implemented dedicated specialisations for the visualisation of functional data in `plot.fData` (`plot.mfData`). In case of `mfData` the graphical window is split into a rectangular lattice so that each component is plotted singularly. The rectangular frame has $\lfloor\sqrt{L}\rfloor$ rows and $\lceil\frac{L}{\lfloor\sqrt{L}\rfloor}\rceil$ columns. To give an example of this visualization method, consider the `mfD_healthy` dataset in the **roahd** package (see Figure 2).

```
data("mfD_healthy", package = "roahd")
```

The dataset `mfD healthy` collects preprocessed (denoised, smoothed and registered) 8-leads electrocardiographic (ECG) signals during a median heartbeat of a sample of 50 healthy subjects. An ECG signal records in a of voltage versus time the electrical activity of the heart using electrodes placed on the skin. These electrodes detect the small electrical changes that are a consequence of cardiac muscle depolarization followed by repolarization during each cardiac cycle (heartbeat). In a conventional 12-lead ECG, ten electrodes are placed on the patient's limbs and on the surface of the chest. The overall magnitude of the heart's electrical potential is then measured from twelve different angles ('leads'). Of these 12 leads, the first six are derived from the same three measurement points. Therefore, any two of these six leads include exactly the same information as the other four. So, the ECG traces gathered of 8 leads, called I, II, V1, V2, V3, V4, V5 and V6. Analogously the dataset `mfD LBBB` contains the ECGs of 50 patients affected by Left Bundle Branch Block, LBBB in the following. These data arise from the project PROMETEO (PROgetto sull'area Milanese Elettrocardiogrammi Teletrasferiti dall'Extra Ospedaliero) started with the aim of spreading the intensive use of ECG as pre-hospital diagnostic tool and of constructing a new database of ECGs with features never recorded before in any other data collection on heart diseases. Measures of electric potential are given in μV and the final sample grid after preprocessing consists of 1024 points at 1KHz. For more details on the original dataset, and on the preprocessing steps, see Ieva et al. (2013).

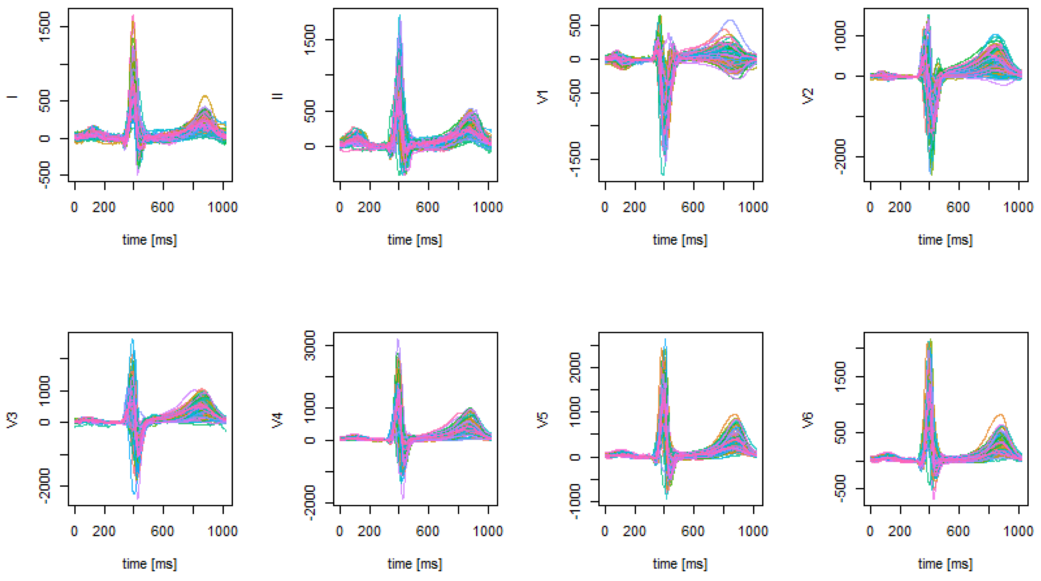

Figure 2: A visualization of the `mfD_healthy` dataset in the **roahd** package.

Simulation of functional data

roahd contains functions that can be used to simulate artificial data sets of functional data, both univariate and multivariate. The data are obtained as realisations of a Gaussian process over a discrete grid with a specific variance-covariance operator and mean, see Rasmussen and Williams (2006). In general, given a covariance function, $C(s,t) := \mathrm{Cov}(X(s), X(t))$, $s,t \in I$, and a mean function $\mu(t) := \mathbb{E}[X(t)]$, the model generating a data sample of size N is:

$$X_i(t) = \mu(t) + \varepsilon(t), \quad \mathrm{Cov}(\varepsilon(s), \varepsilon(t)) = C(s,t), \quad i = 1, \ldots, N, \qquad t \in I.$$

The finite-dimensional approximation on a discrete grid $I = [t_0, \ldots, t_{P-1}]$, with $t_{i+1} - t_i =$

h, $\forall i = 0, \dots, P-1$, instead, is generated according to:

$$X_{i,j} = X_i(t_j) = \mu(t_j) + \varepsilon(t_j), \quad \mathrm{Cov}(\varepsilon(t_j), \varepsilon(t_k)) = C_{j,k}, \quad i = 1, \dots, N, \qquad t_j, t_k \in I.$$

The function **generate_gauss_fdata** requires the sample size **N**, the mean vector $[\mu(t_j)]$ as parameter **centerline**, the matrix representation of the desired variance-covariance operator $[C_{j,k}]$ as parameter **Cov** or, as alternative to **Cov**, its Cholesky factor **CholCov**, which is what is actually used to impose the desired covariance structure of the errors of the generating formulas. A built-in function can be used to generate exponential Matérn covariance functions, namely **exp_cov_function(grid, alpha, beta)**, returning the discretised version of a covariance of the form $C(s,t) = \alpha e^{-\beta|s-t|}$. Here, the parameter α controls the overall level of variability in the signal, while the parameter β affects the autocorrelation length of the signal's noise, with lower values of β leading to wider correlation lengths and vice-versa. This parameter is particularly useful to produce random distortions in the shape of the synthetic functional data around the desired centerline. An example of the function is the following:

```
generate_gauss_fdata( N=50,
centerline = sin( 2 * pi * seq( 0, 1, length.out = 10^3 ) ),
Cov =exp_cov_function( seq( 0, 1, length.out = 10^3 ),
alpha = 0.2, beta  = 0.3 ) )
```

The simulated data are show in Figure 3.

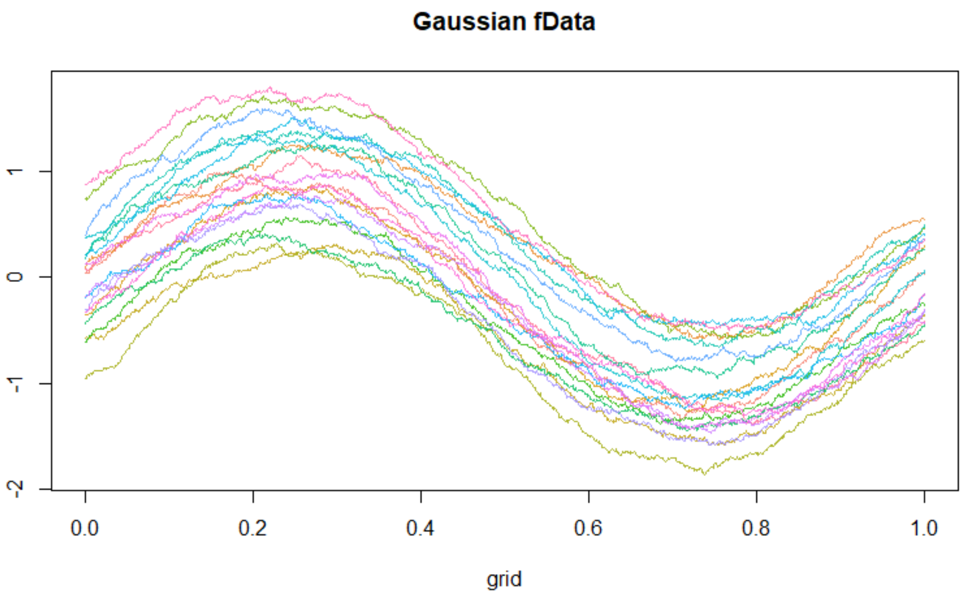

Figure 3: An example of simulated univariate functional data with 50 curves evaluated on a evenly spaced grid of 1000 points.

Similarly, we can use the function **generate_gauss_mfdata** to generate a sample of L-variate functional data according to the following model:

$$X_{i,k} = \mu_k(t) + \varepsilon_k(t), \quad \mathrm{Cov}(\varepsilon_k(s), \varepsilon_k(t)) = C(s,t), \quad \forall i = 1, \dots, N, \quad \forall k = 1, \dots, L,$$

where $Cor(\varepsilon_j(t), \varepsilon_l(t)) = \rho_{j,l} \in [-1, 1]$ specifies the synchronous, constant correlation structure among the components of the functional dataset.

The function requires: the sample size **N**; the number of components of the multivariate data **L**; a matrix containing by rows the means of each component **centerline**; a vector **correlations** of length $1/2 \cdot L \cdot (L-1)$ containing all the correlation coefficients $\rho_{j,l}$ among the components; either a list **listCov** containing the discretised covariance functions over the grid $I \times I$, or a list **listCholCov** of their Cholesky factors.

An example of the function is the following

```
generate_gauss_mfdata( N=100, 2,
centerline = matrix( c( sin( 2 * pi * seq( 0, 1, length.out = 10^3 )),
cos( 2 * pi * seq( 0, 1, length.out = 10^3 ) ) ), nrow = 2, byrow = TRUE ),
correlations = 0.5,
listCov = list(exp_cov_function( seq( 0, 1, length.out = 10^3 ),
alpha = 0.1, beta  = 0.5 ),
exp_cov_function( seq( 0, 1, length.out = 10^3 ),
alpha = 0.5, beta  = 0.1 )))
```

The simulated data are show in Figure 4.

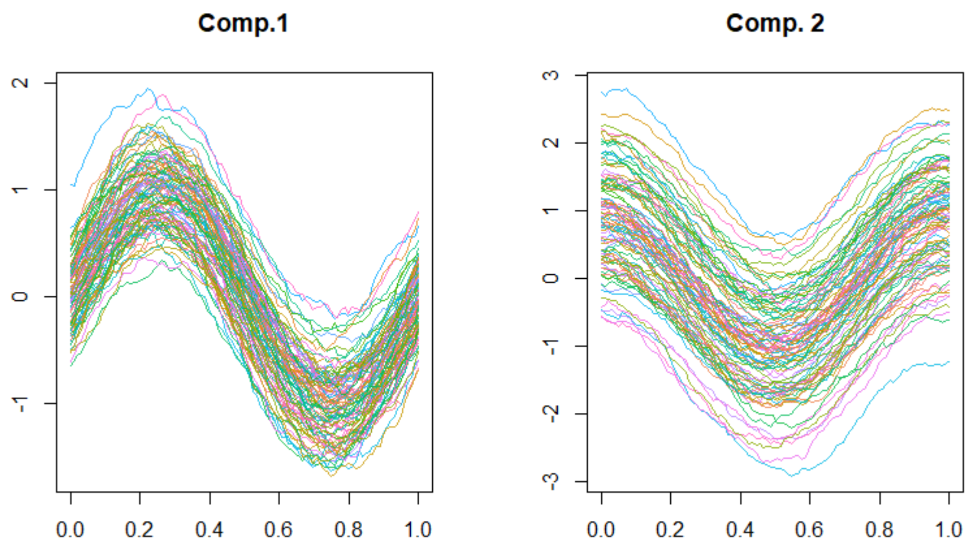

Figure 4: An example of simulated bivariate functional data with 100 bivariates curves evaluated on a evenly spaced grid of 1000 points.

Robust Statistics

In order to provide a center-outward and a down-upward/up-downward order of data, the Band Depth (BD) and Modified Band Depth (MBD) are implemented both for functional and multivariate functional data. Let us recall the empirical version of Band Depth for functional data, as introduced in López-Pintado and Romo (2009) and in López-Pintado and Romo (2011). Given a stochastic process X taking values on the space $\mathcal{C}(I)$ of real continuous functions on the compact interval I, the empirical version of the band depth of order $J \geq 2$ for a function $f \in \mathcal{C}(I)$ is

$$\mathrm{BD}_X^J(f) = \sum_{j=1}^{J} \binom{N}{j}^{-1} \sum_{i_1 < i_2 < \ldots, < i_j} \mathbb{I}\left\{G(f) \subset B(f_{i_1}, \ldots, f_{i_j})\right\}, \tag{1}$$

where the subset of the plane $G(f) = \{(t, f(t)) : t \in I\}$ is the graph of the function f. $B(f_1, f_2, \ldots, f_j)$ is the band in $\mathbb{R}^2$ delimited by $f_1, f_2, \ldots, f_j$, realizations of independent copies of the stochastic process X (i.e. the functional dataset), and it is defined as:

$$B(f_1, f_2, \ldots, f_j) = \{(t, y(t)) : t \in I, \min_{r=1,\ldots,j} f_r(t) \leq y(t) \leq \max_{r=1,\ldots,j} f_r(t)\}.$$

To overcome the problem of heavy ties due to the presence of the indicator function, López-Pintado and Romo (2009) proposed the Modified Band Depth, where the time interval that f spends in the random band is weighted over I. The empirical version of the MBD is:

$$\mathrm{MBD}_X^J(f) = \sum_{j=2}^{J} \binom{N}{j}^{-1} \sum_{1 \leq i_1 < i_2 < \cdots < i_j \leq n} \tilde{\lambda}\{E(f; f_{i_1}, \ldots, f_{i_j})\}, \tag{2}$$

where $E(f) := E(f; f_{i_1}, ..., f_{i_j})=\{t \in I, \min_{r=i_1,...,i_j} f_r(t) \leq f(t) \leq \max_{r=i_1,...,i_j} f_r(t)\}$ and $\tilde{\lambda}(g) = \lambda(E(g))/\lambda(I)$ with λ the Lebesgue measure on I. In López-Pintado and Romo (2009), the authors state that while the choice of J clearly increases the magnitude of depth, it does not affect the induced ordering and therefore the ranks. This was supported by a simulation study in Tarabelloni et al. (2015). To reduce the computational effort we set $J = 2$. Given a set of curves $(f, f_1, ...f_N)$, the MBD of f, that we will denote by $MBD_{\{f_1,...,f_N\}}(f)$, measures the proportion of time interval I where the graph of f belongs to the envelopes of all the possible couples (f_{i_1}, f_{i_2}), $i_1 < i_2 = 1, \ldots, N$.

Both the functions `BD` and `MBD` require either an object of class `fData` or a matrix-like dataset of functional data (e.g., `fData$values`), with observations as rows and measurements over grid points as columns. They return a vector containing the values of depth for the given dataset. Thanks to these values, the dataset $f_1, ..., f_N$ can be center-outward ranked. We will denote $f_{[i]}$ the sample curve associated with the i−th largest depth value, so $f_{[1]} = \text{argmax}_{f \in \{f_1,...,f_N\}} MBD(f)$ is the *median* (deepest and most central) curve, and $f_{[N]} = \text{argmin}_{f \in \{f_1,...,f_N\}} MBD(f)$ the most outlying one. Moreover, for $\alpha \in (0, 1)$ we can construct the α% central region determined by a sample of curves as:

$$\mathcal{C}_\alpha = \left\{(t, y(t)) : \min_{r=1,...,\lceil \alpha n \rceil} f_{[r]}(t) \leq y(t) \leq \max_{r=1,...,\lceil \alpha n \rceil} f_{[r]}(t)\right\}. \tag{3}$$

As an example, let us compute the depth measures and the median of the simulated dataset shown in Figure 3:

```
MBD(fD)
[1] 0.48503510 0.46228408 0.51505469 0.31594122 0.37595102
[6] 0.49060245 0.35306449 0.40934204 0.47676898 0.37799510
[11] 0.22585633 0.45465633 0.49204245 0.34556571 0.14763429
[16] 0.27375020 0.42478041 0.51758041 0.07788571 0.49579265
...
```

Another interesting down-upward/up-downward order of data can be built on top of Epigraph Index (EI) and Hypograph Index (HI) or of their corresponding Modified versions (MEI and MHI). We recall the definition of EI (HI) for univariate functional data as introduced in López-Pintado and Romo (2011). Given a stochastic process X taking values on the space $\mathcal{C}(I)$ the empirical version of the Epigraph (Hypograph) Index of a function $f \in \mathcal{C}(I)$ are:

$$EI_X(f) = \frac{1}{N}\sum_{i=1}^{N} \mathbb{I}\{f_i(t) \geq f(t), \quad \forall t \in I\}, \tag{4}$$

$$HI_X(f) = \frac{1}{N}\sum_{i=1}^{N} \mathbb{I}\{f_i(t) \leq f(t), \quad \forall t \in I\}, \tag{5}$$

where $f_1, f_2, ..., f_N$ are realizations of independent copies of the stochastic process X (i.e. the functional dataset). As before, to overcome the problem of heavy ties it is more suitable to use the modified version of these indexes, whose empirical version are:

$$MEI_X(f) = \frac{1}{N}\sum_{i=1}^{N} \tilde{\lambda}(\{t \in I, f_i(t) \geq f(t)\}). \tag{6}$$

$$MHI_X(f) = \frac{1}{N}\sum_{i=1}^{N} \tilde{\lambda}(\{t \in I, f_i(t) \leq f(t)\}). \tag{7}$$

Therefore, given a set of curves $(f, f_1, ...f_N)$ the MEI (MHI) of f, denoted by $MEI_{\{f_1,...,f_N\}}(f)$ ($MHI_{\{f_1,...,f_N\}}(f)$) accounts for the mean proportion of time interval I where f lies below (above) the curves of the sample. Like depths, all the functions `EI` (`HI`) and `MEI` (`MHI`) require either an object of class `fData` or a matrix-like dataset of functional data (e.g., `fData$values`), with observations as rows and measurements over grid points as columns. They return a vector containing the values of the corresponding indexes for the given dataset, that can provide the desired ordering of data. In López-Pintado and Romo (2011) the authors propose another well-posed definition of depth, the Modified Half Region Depth (MHRD) for functional data as:

$$MHRD_{\{f_1,...,f_N\}}(f) = \min(MEI_{\{f_1,...,f_N\}}(f), MHI_{\{f_1,...,f_N\}}(f)). \tag{8}$$

`MHRD` in **roahd** computes the MHRD of elements of a univariate functional dataset, and has the same usage as the previous functions.

In Ieva et al. (2013) and Ieva and Paganoni (2017) these statistics have been generalized to multivariate functional framework. Let $\mathbf{X}$ be a stochastic process taking values in the space $C(I; \mathbb{R}^L)$ of continuous functions $\mathbf{f} = (f_1, ..., f_L) : I \to \mathbb{R}^L$, with I as before. We have a dataset F_N constituted of $N \in \mathbb{N}$ sample observations of this process, which we indicate by $\mathbf{f}_1, \ldots, \mathbf{f}_N$, where $\mathbf{f}_j = (f_{j1}, ..., f_{jL})$. The MBD of $\mathbf{f}$ with respect to F_n becomes

$$MBD_{\{\mathbf{f_1},...,\mathbf{f_n}\}}(\mathbf{f}) = \sum_{k=1}^{L} p_k MBD^J_{\{f_{1k},...,f_{Nk}\}}(f_k), \tag{9}$$

with $p_k > 0$, $\forall\ k = 1, ..., L$, $\quad \sum_{k=1}^{L} p_k = 1$.
Analogously we define the MEI (MHI) of $\mathbf{f}$ with respect to F_N as

$$MEI_{\{\mathbf{f_1},...,\mathbf{f_N}\}}(\mathbf{f}) = \sum_{k=1}^{L} p_k MEI_{\{f_{1k},...,f_{Nk}\}}(f_k), \tag{10}$$

$$MHI_{\{\mathbf{f_1},...,\mathbf{f_N}\}}(\mathbf{f}) = \sum_{k=1}^{L} p_k MHI_{\{f_{1k},...,f_{Nk}\}}(f_k), \tag{11}$$

with $p_k > 0$, $k = 1, ..., L$, $\quad \sum_{k=1}^{L} p_k = 1$. In (9), (10) and (11) the curves that form the envelops are the components of the curves in F_N. In **roahd** the function `multiMBD` computes the MBD for a dataset of multivariate curves. In particular `multiMBD` requires either an object of class `mfData` or a list of 2-dimensional matrices (`Data`) having as rows the units of that component and as columns the measurements of the functional data over the grid, as well as either a set of weights `weights` or the string *uniform* specifying that a set of uniform weights (of value $1/L$, where L is the number of components of the functional dataset) must be employed. The function returns a vector containing the depth of each element of the multivariate functional dataset. As an example let us compute the depth measures of the simulated dataset (named `mfD`) shown in Figure 4, using uniform weights:

```
multiMBD(mfD, weights="uniform")
[1] 0.40842020 0.45438788 0.24038384 0.31500606 0.35914343
[6] 0.48603636 0.44544040 0.42960606 0.37119798 0.21466667
[11] 0.46331111 0.37947677 0.46344646 0.39914141 0.30079394
[16] 0.48643838 0.14745657 0.47115354 0.41804242 0.32814545
...
```

The choice for the weights p_ks averaging the contribution of each component of the multivariate functional data is usually problem-driven, and in general no gold standards are available. If there is no a priori knowledge about the dependence structure between components, they could be chosen uniformly. In Tarabelloni et al. (2015) a different choice has been proposed, taking into account the distance between the estimated variance-covariance operators of the two groups identified by the binary outcome which was the focus of the study. In Ieva and Paganoni (2017) the weights p_ks are chosen taking into account the variability of each component of the multivariate functional process that generates data, so the weight of each component is proportional to the inverse of spectral norm of its variance-covariance operator:

$$q_k = 1/\lambda_k^{(1)} \quad \text{and} \quad p_k = \frac{q_k}{\sum q_k}, \tag{12}$$

where $\lambda_k^{(1)}$ is the maximum eigenvalue of the variance-covariance operator of the $k-$th component.

The functions `median_fData` (`median_mfData`) of the package compute the sample median of a univariate (multivariate) functional dataset based on a definition of depth for univariate (multivariate) functional data. Their input is the dataset whose median is required, in form of `fData` or `mfData` object, and a string specifying the name of the depth definition to use, as parameter `type`. This name should bind to a function actually defined in the workspace, such as the build-in ones of **roahd** (e.g., MBD, MHRD, etc.).

Figure 5 shows the plot of healthy ECG data (see `mfD_healthy`) with superimposed the multivariate functional median computed maximizing the multivariate MBD (9) with uniform weights.

```
median_mfData(mfD_healthy, type = "multiMBD")
```

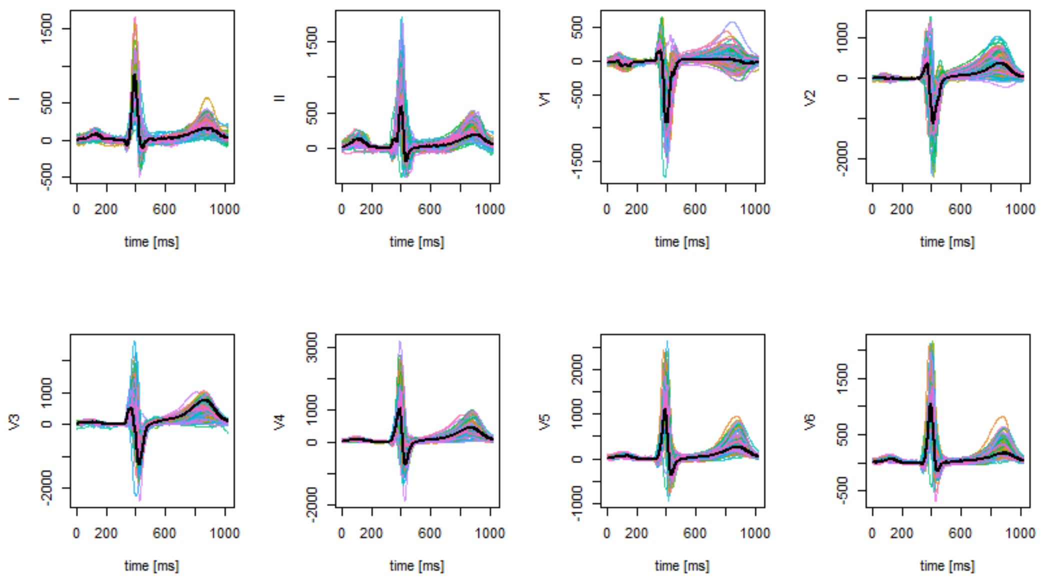

Figure 5: Plot of healthy data with superimposed the multivariate functional median computed maximizing the multivariate MBD (9) with uniform weights.

Correlation coefficient

When dealing with multivariate functional data, it is possible to compute correlation coefficients between observations' univariate components that generalise the Spearman's coefficient ρ_s, see Valencia et al. (2015b,a). In particular, let us consider a set of bivariate ($L = 2$) functional data $\{\mathbf{f_1}, ..., \mathbf{f_N}\}$. The Spearman coefficient related to the data set is defined by

$$\hat{\rho_s} = \hat{\rho_p}(MEI_{\{f_{11},...,f_{N1}\}}(\mathbf{f_1}), MEI_{\{f_{12},...,f_{N2}\}}(\mathbf{f_2})), \tag{13}$$

where $\hat{\rho_p}$ is the usual Pearson correlation coefficient between vectors and

$$MEI_{\{f_{11},...,f_{N1}\}}(\mathbf{f_1}) = (MEI_{\{f_{11},...,f_{N1}\}}(f_{11}), ..., MEI_{\{f_{11},...,f_{N1}\}}(f_{N1})),$$
$$MEI_{\{f_{12},...,f_{N2}\}}(\mathbf{f_2}) = (MEI_{\{f_{12},...,f_{N2}\}}(f_{12}), ..., MEI_{\{f_{12},...,f_{N2}\}}(f_{N2})).$$

Another definition can be obtained by replacing MEI by MHI. The properties of $\hat{\rho_s}$ are detailed in Valencia et al. (2015b), where also its consistency is proved.

The function `cor_spearman` can be used to compute the Spearman correlation coefficient (13) for a bivariate `mfData` object, using the ordering definition specified by `ordering` (the default is to use `MEI`) to rank univariate components and then compute the correlation coefficient. Besides `MEI`, also `MHI` can be used to determine ranks.

Another well known measure for concordance in bivariate data is the Kendall's τ index. In Valencia et al. (2015a) the authors propose two generalizations of τ based on two different preorders between curves $\preceq$. Consider two functions g and h in $\mathcal{C}(I)$. Two possible preorders are:

$$g(t) \preceq_m h(t) \equiv \max_{t\in I} g(t) \le \max_{t\in I} h(t), \tag{14}$$

$$g(t) \preceq_i h(t) \equiv \int_I (h(t) - g(t))dt \ge 0. \tag{15}$$

Let us consider a set of bivariate ($L = 2$) functional data $\{\mathbf{f_1}, ..., \mathbf{f_N}\}$, then the functional $\hat{\tau}$ is

$$\hat{\tau} = \binom{N}{2} \sum_{i<j}^{N} 2\mathbb{I}(f_{i1} \preceq f_{j1}, f_{i2} \preceq f_{j2}) + 2\mathbb{I}(f_{j1} \preceq f_{i1}, f_{j2} \preceq f_{i2}) - 1. \tag{16}$$

In fact it is possible to prove that $\hat{\tau}$ in (16) measures the difference between the number of concordant pairs of curves and the number of discordant pairs of curves, using a possible preorder. The function `cor_kendall` can be used to compute the Kendall correlation coefficient (16) for a bivariate `mfData` object, using the ordering definition specified by `ordering` (by default `max` is used, i.e., formula (14)) to rank univariate components, then to compute concordant and discordant pairs and then the correlation coefficient. Also `area` (i.e., formula (15)) can be used. As an example let us compute $\hat{\rho}$ and $\hat{\tau}$ on the simulated dataset (named `mfD`) shown in Figure 4:

```
cor_spearman(mfD, ordering ="MEI")
[1] 0.6098597

cor_kendall(mfD, ordering="area")
[1] 0.4222222
```

Inference on Spearman correlation

In Ieva et al. (2018) a boostrap-based inferential framework for the Spearman coefficient is introduced. In particular the authors suggest to compute a sample from the bootstrap distribution of the statistic $\hat{\rho}$, i.e. $(\hat{\rho_1}^*, ...\hat{\rho_B}^*)$, where $\hat{\rho_j}^*$ denotes the value of the statistics computed on a random sample of size N drawn with replacement from the population of N equally likely data, named boostrap sample, and B is the number of boostrap samples considered. Using the empirical quantiles of the boostrap distribution of the statistics, it's possible to compute a confidence interval of a fixed level (say $1-\alpha$, $\alpha \in (0,1)$):

$$(\hat{\theta_l}; \hat{\theta_u}) = (\hat{\rho}^*_{\alpha/2}; \hat{\rho}^*_{1-\alpha/2}); \tag{17}$$

where $\hat{\rho}^*_{\alpha}$ denotes the $\alpha\%$ percentile of the sample boostrap distribution $(\hat{\rho_1}^*, ...\hat{\rho_B}^*)$. To mitigate the bad coverage performances of (17), as detailed in Efron and Tibshirani (1993), it's better to consider an improved version of the percentile method called Bias-Corrected and Accelerated (BCA) interval. This improved version of the confidence interval is implemented by the function `BCIntervalSpearman`. This function requires: two univariate functional datasets in form of `fData` objects, `fD1`, `fD2`; the ordering relation to be used in the Spearman's coefficient computation as the parameter `ordering`; the number of bootstrap iterations to use in order to estimate the confidence interval, `bootstrap_iterations` and the coverage probability (1-α).
As an example let us compute a BCA interval of confidence 0.95 for the Spearman correlation coefficient of the simulated dataset (named `mfD`) shown in Figure 4:

```
BCIntervalSpearman(mfD$fDList[[1]], mfD$fDList[[2]], ordering = 'MEI',
alpha=0.05, bootstrap_iterations = 1000)
$lower
[1] 0.6520883

$upper
[1] 0.9819355
```

A verbosity parameter can be set in function `BCIntervalSpearman` in order to log information on the function's progress when the computational time is long. The simple or Bias-Corrected and Accelerated version of the confidence interval allows for testing the presence of dependency among two families od univariate curves, i.e. $H_0 : \rho_s = 0$ vs $H_1 : \rho_s \neq 0$. Consider now the case of a multivariate functional dataset, where the observations are realizations of the stochastic process $\mathbf{X}$ taking values in the space $C(I; \mathbb{R}^L)$, $L > 2$. In this case, the pattern of dependence between the components can be expressed in a symmetrix matrix S, named Spearman matrix. Its entry $S_{i,j}$ is the Spearman coefficient between the data of the i-th and j-th components. Analogously, for each of the $L(L-1)/2$ coefficients the corresponding BCA confidence interval can be computed. The function `cor_spearman` applied to a dataset of type `mfData` returns the pointwise estimate of the Spearman matrix S, while the function `BCIntervalSpearmanMultivariate` returns two matrices containing the lower and upper bounds of the corresponding confidence intervals. To clarify their use, we show the results corresponding to the first two leads, i.e. I and II of `mfD_healthy`.

```
mfD_healthy_subset = as.mfData(list(mfD_healthy$fDList[[1]],
mfD_healthy$fDList[[2]]))

cor_spearman(mfD_healthy_subset, ordering='MEI')
[1] 0.6840466

BCIntervalSpearmanMultivariate(mfD_healthy_subset,
ordering='MEI', alpha=0.05, bootstrap_iterations = 1000)
$lower
          [,1]      [,2]
```

```
[1,] 1.0000000 0.4805781
[2,] 0.4805781 1.0000000

$upper
         [,1]     [,2]
[1,] 1.000000 0.820072
[2,] 0.820072 1.000000
```

In order to perform a comparison between correlation patterns across different populations of multivariate functional data, in Ieva et al. (2018) the authors propose a bootstrap procedure to test the equality between the two corresponding Spearman matrices. Consider two multivariate functional datasets, where the observations are realizations of the stochastic processes $\mathbf{X}$ and $\mathbf{Y}$, respectively, both taking values in the space $C(I;\mathbb{R}^L)$. We want to perform a bootstrap test to check the hypothesis:

$$H_0 : S_X = S_Y \qquad \text{vs} \qquad H_1 : S_X \neq S_Y, \tag{18}$$

where S_X and S_Y denote the $L \times L$ matrices of Spearman correlation coefficients of the two populations. The proposed bootstrap test statistics is based on a norm of the differences between the two matrices. In particular the authors consider three different norms in the space of the $L \times L$ matrices:

(i) One norm: $\|W\|_1 = \max_{j=1,\cdots,L} \sum_{i=1}^{L} |w_{ij}|$;

(ii) Infinity norm: $\|W\|_\infty = \max_{i=1,\cdots,L} \sum_{j=1}^{L} |w_{ij}|$;

(iii) Frobenius norm: $\|W\|_F = \sqrt{\sum_{i=1}^{L} \sum_{j=1}^{L} |w_{ij}|^2}$.

The function `BTestSpearman` performs the test described above and requires: two univariate functional samples in form of `mfData` object, `mfD1`, `mfD2`; the ordering relation to be used in the Spearman's coefficient computation `ordering`; the number of bootstrap iterations to be performed `bootstrap_iterations`; the norm to measure the differences between the Spearman correlation matrices of the two functional datasets, `normtype` (the allowed values are the same as for parameter `type` in R's base function `norm`). The function returns the estimates of the test's p-value and statistics. As an example let us perform the test considering the first two components of `mfd_healthy` and `mfD_LBB` datasets provided by the **roahd** package.

```
mfD_healthy_subset = as.mfData(list(mfD_healthy$fDList[[1]],
mfD_healthy$fDList[[2]]))

mfD_LBBB_subset = as.mfData(list(mfD_LBBB$fDList[[1]],
mfD_LBBB$fDList[[2]]))

BTestSpearman(mfD_healthy_subset, mfD_LBBB_subset,
bootstrap_iterations = 1000,
ordering = "MEI", normtype = "f")

$pvalue
[1] 0.473

$phi
[1] 0.06562356
```

Graphical tools

The tools shown in this section (i.e., the functional boxplot and the outliergram) enable a complete inferential analysis of (multivariate) functional data based on robust statistics, like depth measures, described in Section 2.3. These tools are very useful also in the outlier detection framework which is of primary interest in FDA, since outliers may deeply affect the inference of high dimensional data, especially whenever the sample size is small.
The functional boxplot (see Sun and Genton (2011)) is obtained by ranking functions from the center of the distribution outwards thanks to a suitable depth definition, computing the region of 50% most central functions, see Eq. (3). The fences are obtained by inflating such region by a factor F. Given the envelope of the functions entirely contained inside the inflated region, the data crossing these fences even for one time instant are considered outliers. Once the outlying observations have been

identified, they can be isolated from the original dataset and either carefully examined or discarded as corrupted by undesired factors.

The function `fbplot` computes the depths of a dataset and marks outlying observations. If used with graphical option on (default behaviour), it also plots the functional boxplot of the dataset. `fbplot` requires: the univariate functional dataset whose functional boxplot must be determined in form of an `fData` object `Data`; either a vector containing the depths for each statistical unit of the dataset, or a string containing the name of the method you want to use to compute; the value of the inflation factor, `Fvalue` (the default value is 1.5). In Figure 6 we show the functional boxplot of the first lead, i.e. I of `mfD_healthy`.

```
fbplot(mfD_healthy$fDList[[1]], Depths="MBD", Fvalue=3,
main="Functional Boxplot")

$Depth
[1] 0.4399681 0.1534263 0.4097385 0.4116510 0.3872242
[6] 0.4123326 0.2387404 0.3568670 0.3669691 0.4601483
[11] 0.3006872 0.3744531 0.1360906 0.4319324 0.3206186
[16] 0.2400199 0.4340800 0.4462739 0.2805389 0.4638281
...

$Fvalue
[1] 3

$ID_outliers
2
```

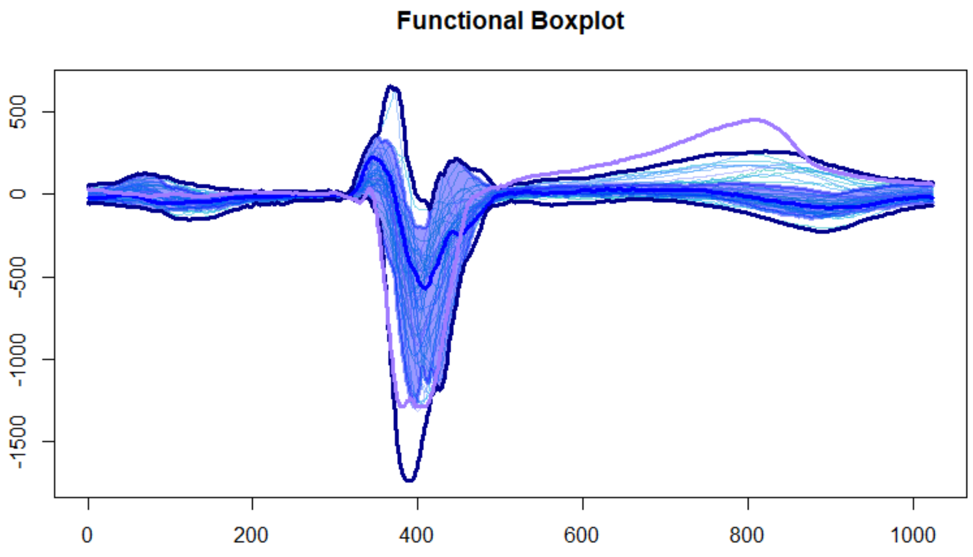

Figure 6: Functional Boxplot of the first lead, i.e., I of healthy data

The function `fbplot` also allows to automatically compute the best adjustment factor `F` that yields a desired proportion of outliers (True Positive Rate, `TPR`) of a Gaussian dataset with same center and covariance function as the `fData` object (see Sun and Genton (2012)). Such automatic tuning involves the simulation of a number `N_trials` of separate datasets of Gaussian functional data with same center and covariance as the original dataset (the covariance is robustly estimated with the function `covOGK` of the package **robustbase**, see Maronna and Zamar (2002)) of size `trial_size`, and the computation of `N_trials` values for `Fvalue` such that the desired proportion `TPR` of observations is flagged as outliers. The optimal value of `Fvalue` for the original population is then found as the average of the previously computed values `Fvalue`. The parameters to control the adjustment procedure can be passed through the argument `adjust`, whose default is `FALSE` and otherwise is a list with (some of) the fields:

- `N_trials`: the number of repetitions of the adjustment procedure based on the simulation of a Gaussisan dataset of functional data, each one producing an adjusted value of F, which will lead to the averaged adjusted value `Fvalue`. Default is 20;

- `trial_size`: the number of statistical units in the Gaussian population of functional data that will be simulated at each repetition of the adjustment procedure. Default is $8 \times N$;
- `TPR`: the True Positive Rate of outliers, i.e., the proportion of observations in a dataset without amplitude outliers that have to be considered outliers. Default is $2\phi(4z_{0.25})$ where ϕ and z_α denote, respectively, the cumulative distribution function and the quantile of order α of a standard Gaussian distribution.
- `F_min`: the minimum value of `Fvalue`, defining the left boundary for the optimisation problem aimed at finding the optimal value of `Fvalue`;
- `F_max`: the maximum value of `Fvalue`, defining the right boundary for the optimisation problem aimed at finding the optimal value of `Fvalue`;
- `tol`: the tolerance to be used in the optimisation problem aimed at finding the optimal value of `Fvalue`;
- `maxiter`: the maximum number of iterations to solve the optimisation problem aimed at finding the optimal value of `Fvalue`.

Due to the **S3** specialization, `fbplot` can construct also functional boxplots of a multivariate functional dataset (see Ieva and Paganoni (2013)). In Figure 7 we show the functional boxplot of the first two leads, i.e. I and II of `mfD_healthy`.

```
mfD_healthy_subset <- as.mfData(list(mfD_healthy$fDList[[1]],
mfD_healthy$fDList[[2]]))

fbplot( mfD_healthy_subset, Fvalue = 1.5, xlab = 'time',
ylab = list( 'Values 1', 'Values 2' ),
main = list( 'First component', 'Second component' ) )

$Depth
[1] 0.4061113 0.1695619 0.4086113 0.4204500 0.3005780
[6] 0.4233634 0.3170089 0.3522569 0.3821995 0.4625769
[11] 0.3055684 0.3535029 0.1630668 0.4213114 0.3553795
[16] 0.3079317 0.3853858 0.3711121 0.2640493 0.4489892
...

$Fvalue
[1] 1.5

$ID_outliers
[1]  2 16

$Depth
[1] 0.43298990 0.36373333 0.40771515 0.38667677 0.10381616
[6] 0.44186869 0.36482424 0.34480404 0.45246061 0.36904040
[11] 0.25263232 0.28706869 0.42344040 0.29333333 0.02921414
[16] 0.49872121 0.49638384 0.41457980 0.25275960 0.29689697
 ....

$Fvalue
[1] 2.5

$ID_outliers
[1]  5 15 27 31 32 33 34 35 44 50 52 55 57 59 63 67 71 73 75 85 23
```

Let us point out that the functional boxplot has been constructed mainly for detection of magnitude outliers, i.e., curves that lie far from the range of the majority bulk of data.

Outliergram

A method that can be used to detect shape outliers and covariance outliers is the outliergram (see Arribas-Gil and Romo (2014)), based on the computation of MBD and MEI (see (2) and (6)) of univariate functional data. Shape outliers are curves that present a different pattern with respect to the rest of the data in terms of their derivatives and covariance outliers are curves generated by

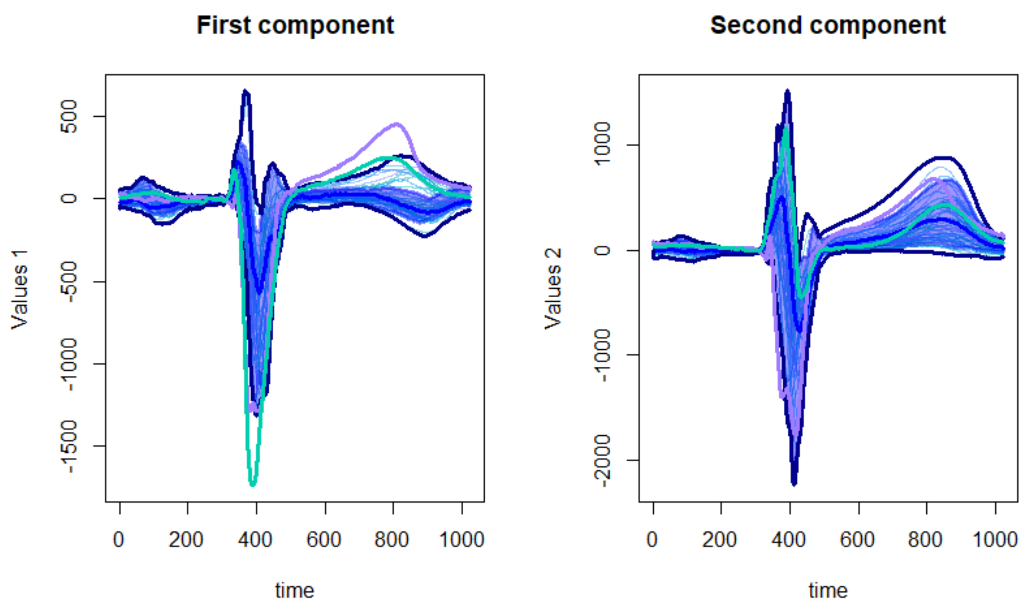

Figure 7: Functional Boxplot of the simulated data presented in Figure 4

a model that is different from the model of the majority of data just in terms of the variance and covariance operator that affects the second order moments of data. Given a set of data $f_1, \ldots, f_n$ in the space $\mathcal{C}(I;\mathbb{R})$ of the continuous functions the following inequality holds:

$$MBD_{\{f_1,\ldots,f_n\}}(f_j) \leq a_0 + a_1 MEI_{\{f_1,\ldots,f_n\}}(f_j) + a_2 n^2 (MEI_{\{f_1,\ldots,f_n\}}(f_j))^2, \quad j = 1, \cdots, n, \tag{19}$$

where $a_0 = a_2 = -2/(n(n-1))$ and $a_1 = 2(n+1)/(n-1)$. So considering a scatterplot of MBD against multivariate MEI of data, the points lying far from the quadratic boundary (19) correspond to shape outliers, and data with very low values of MBD are potential magnitude outliers (see Arribas-Gil and Romo (2014)). The function `outliergram` displays the outliergram of a univariate functional dataset of class `fData` (see Figure 8) and returns a vector of observation IDs indicating the outlying observations in the dataset.

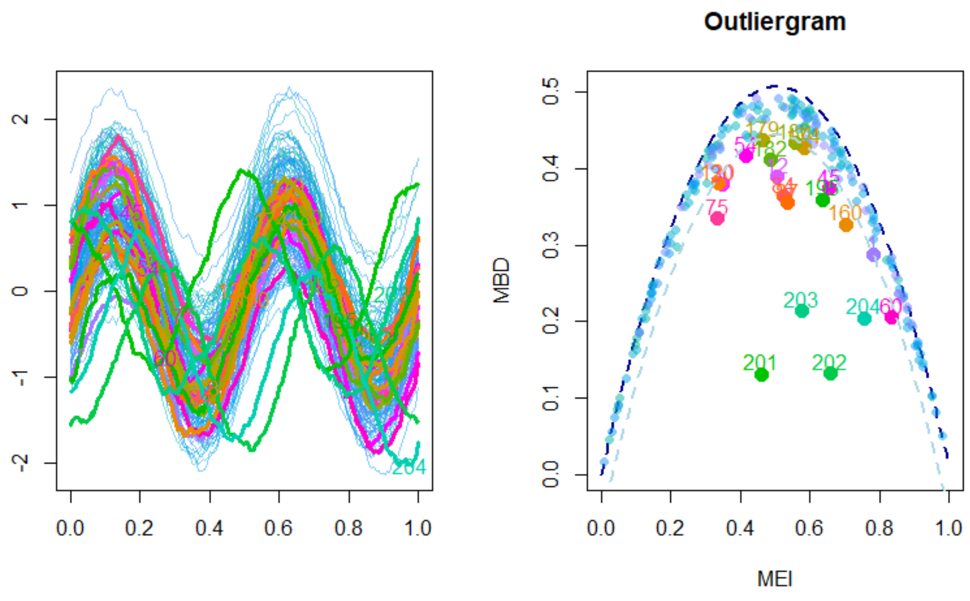

Figure 8: An example of outliergram on simulated univariate functional data

The function `multivariate_outliergram` implements the generalisation of the outliergram to multivariate functional data, following Ieva and Paganoni (2017). Let $\mathbf{X}$ be a stochastic process taking values in the space $C(I; \mathbb{R}^L)$ of continuous functions $\mathbf{f} = (f_1, ..., f_L) : I \to \mathbb{R}^L$, where I is a compact interval of $\mathbb{R}$. Let us consider a dataset F_N constituted of $N \in \mathbb{N}$ sample observations of this process, which we indicate by $\mathbf{f}_1, \dots, \mathbf{f}_N$, $\mathbf{f}_j = (f_{j1}, ..., f_{jL})$. In Ieva and Paganoni (2017) the following inequality is proved:

$$MBD^J_{\{\mathbf{f_1},\dots,\mathbf{f_n}\}}(\mathbf{f}) \leq a_0 + a_1 MEI_{\{\mathbf{f_1},\dots,\mathbf{f_n}\}}(\mathbf{f}) + a_2 n^2 (MEI_{\{\mathbf{f_1},\dots,\mathbf{f_n}\}}(\mathbf{f}))^2, \qquad (20)$$

where $a_0 = a_2 = -2/(n(n-1))$ and $a_1 = 2(n+1)/(n-1)$, and $MBD^J_{\{\mathbf{f_1},\dots,\mathbf{f_n}\}}(\mathbf{f})$ and $MEI_{\{\mathbf{f_1},\dots,\mathbf{f_n}\}}(\mathbf{f})$ are defined in (9) and (10), respectively.
So the outliergram for multivariate functional data is constructed in analogy with the univariate one and based on the quadratic boundary 20. The function `multivariate_outliergram` displays the outliergram of a multivariate functional dataset of class `mfData` (see Figure 9) and returns a vector of observation IDs indicating the outlying observations in the dataset.

```
multivariate_outliergram(mfD, Fvalue = 2, shift=TRUE)

$Fvalue
2

$Depth
[1] 0.0386524565 0.0267086844 0.1297124989 0.0474675556
[5] 0.0296031652 0.0446769503 0.0353957777 0.0294200492


$ID_outliers
[1]  12 18 32 47 70 83 91 96
```

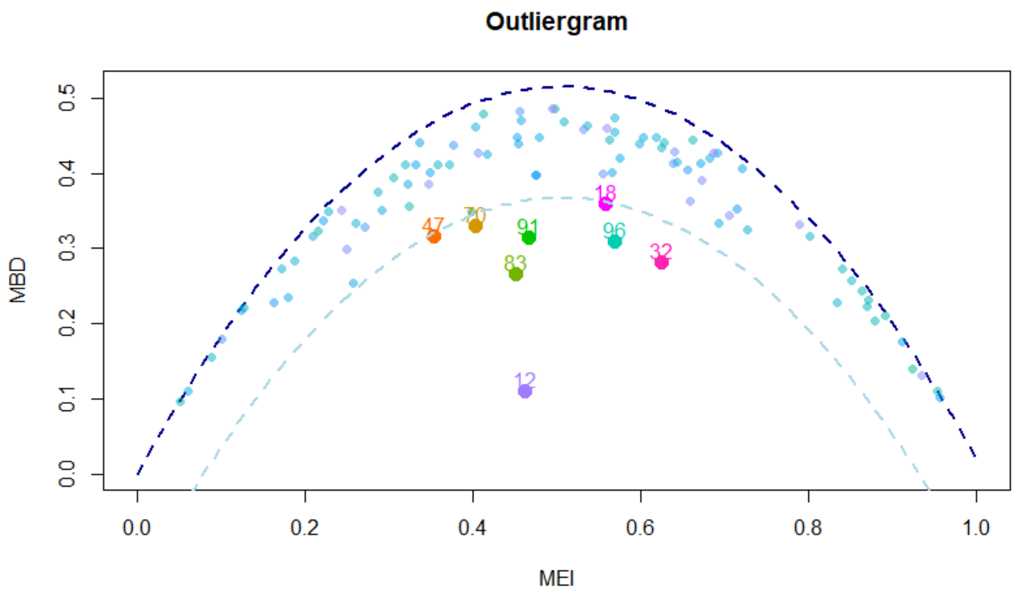

Figure 9: The outliergram of simulated data presented in Figure 4.

Conclusions

In this paper we have described the implementation in the **roahd** package of several statistical methods that deal with the robust statistical analysis of univariate and multivariate functional data, and some graphical tools mainly aimed at identifying and discarding outliers from a dataset of (potentially multivariate) functional data. The package should simplify the access and use of these strongly nonparametric methods to perform a suitable robust inferential analysis of high dimensional and complex data.

Bibliography

A. Arribas-Gil and J. Romo. Shape outlier detection and visualization for functional data: the outliergram. *Biostatistics*, 15(4):603–619, 2014. [p13, 14]

G. Claeskens, M. Hubert, L. Slaets, and K. Vakili. Multivariate functional halfspace depth. *Journal of the American Statistical Association*, 109(505):411–423, 2014. [p2]

X. Dai, P. Z. Hadjipantelis, K. Han, and H. Ji. *fdapace: Functional Data Analysis and Empirical Dynamics*, 2018. URL https://CRAN.R-project.org/package=fdapace. R package version 0.4.0. [p1]

B. Efron and R. Tibshirani. *An Introduction to the Bootstrap*. Chapman & Hall, 1993. [p10]

M. Febrero-Bande and M. Oviedo de la Fuente. Statistical computing in functional data analysis: The R package fda.usc. *Journal of Statistical Software*, 51(4):1–28, 2012. [p1]

F. Ferraty and P. Vieu. *Nonparametric Functional Data Analysis: Theory and Practice*. Springer, New York, 2006. [p1]

L. Horvath and P. Kokoszka. *Inference for Functional Data with Applications*. Springer, New York, 2012. [p1]

F. Ieva and A. Paganoni. Component-wise outlier detection methods for robustifying multivariate functional samples. *Statistical Papers*, 2017. URL https://doi.org/10.1007/s00362-017-0953-1. [p8, 15]

F. Ieva and A. M. Paganoni. Depth measures for multivariate functional data. *Communication in Statistics - Theory and Methods*, 42(7):1265 – 1276, 2013. [p2, 13]

F. Ieva, A. Paganoni, D. Pigoli, and V. Vitelli. Multivariate functional clustering for the morphological analysis of ecg curves. *Journal of the Royal Statistical Society: Series C (Applied Statistics)*, 62 (3):401 – 418, 2013. [p4, 8]

F. Ieva, F. Palma, and J. Romo. Bootstrap-based inference for dependence in multivariate functional data. MOX Report 30/2018, Politecnico di Milano, 2018. URL https://www.mate.polimi.it/biblioteca/add/qmox/30-2018.pdf. [p10, 11]

P. Kokoszka and M. Reimherr. *Introduction to Functional Data Analysis*. Chapman & Hall, 2017. [p1]

R. Liu and K. Singh. A quality index based on data depth and multivariate rank tests. *Journal of the American Statistical Association*, 88(421):252 – 260, 1993. [p2]

R. Liu, J. Parelius, and K. Singh. Multivariate analysis by data depth: Descriptive statistics, graphics and inference. *The Annals of Statistics*, 27(3):783–858, 1999. [p2]

S. López-Pintado and J. Romo. Depth-based inference for functional data. *Computational Statistics & Data Analysis*, 51(10):4957–4968, 2007. [p2]

S. López-Pintado and J. Romo. On the concept of depth for functional data. *Journal of the American Statistical Association*, 104(486):718 – 734, 2009. [p2, 6, 7]

S. López-Pintado and J. Romo. A half-region depth for functional data. *Computational Statistics & Data Analysis*, 55(4):1679–1695, 2011. [p6, 7]

S. Lopez-Pintado, Y. Sun, and M. Genton. Simplicial band depth for multivariate functional data. *Advances in Data Analysis and Classification*, 8:321–338, 2014. [p2]

R. A. Maronna and R. H. Zamar. Robust estimates of location and dispersion for high-dimensional datasets. *Technometrics*, 44(4):307–317, 2002. [p12]

J. Ramsay and B. Silverman. *Applied Functional Data Analysis: Methods and Case Studies*. Springer, New York, 2002. [p1]

J. Ramsay and B. Silverman. *Functional Data Analysis*. Springer, New York, second edition, 2005. [p1]

J. O. Ramsay, G. Hooker, and S. Graves. *Functional Data Analysis with R and MATLAB*. Springer Publishing Company, Incorporated, 1st edition, 2009. [p1]

J. O. Ramsay, H. Wickham, S. Graves, and G. Hooker. *fda: Functional Data Analysis*, 2014. URL https://CRAN.R-project.org/package=fda. R package version 2.4.4. [p1]

C. Rasmussen and C. Williams. *Gaussian Processes for Machine Learning*. MIT Press, 2006. [p4]

H. L. Shang and R. Hyndman. *rainbow: Bagplots, Boxplots and Rainbow Plots for Functional Data*, 2019. URL https://CRAN.R-project.org/package=rainbow. R package version 3.6. [p1]

Y. Sun and M. G. Genton. Functional boxplots. *Journal of Computational and Graphical Statistics*, 20(2):316–334, 2011. [p11]

Y. Sun and M. G. Genton. Adjusted functional boxplots for spatio-temporal data visualization and outlier detection. *Environmetrics*, 23(1):54–64, 2012. [p12]

N. Tarabelloni, F. Ieva, R. Biasi, and A. Paganoni. Use of depth measure for multivariate functional data in disease prediction: An application to electrocardiograph signals. *The international journal of biostatistics*, 11(2):189–201, 2015. [p7, 8]

N. Tarabelloni, A. Arribas-Gil, F. Ieva, A. M. Paganoni, and J. Romo. *roahd: Robust Analysis of High Dimensional Data*, 2017. URL https://CRAN.R-project.org/package=roahd. R package version 1.3. [p1]

J. D. Tucker. *fdasrvf: Elastic Functional Data Analysis*, 2017. URL https://CRAN.R-project.org/package=fdasrvf. R package version 1.8.3. [p1]

J. Tuckey. Mathematics and the picturing of data. In *Proceedings of the International Congress of Mathematicians, Vancouver*, volume 2, pages 523 – 531, 1975. [p2]

D. Valencia, J. Romo, and R. Lillo. A kendall correlation coefficient for functional dependence. Technical report, Universidad Carlos III de Madrid, http://EconPapers.repec.org/RePEc:cte:wsrepe:ws133228, 2015a. [p9]

D. Valencia, J. Romo, and R. Lillo. Spearman coefficient for functions. Technical report, Universidad Carlos III de Madrid, http://EconPapers.repec.org/RePEc:cte:wsrepe:ws133329, 2015b. [p9]

A. Zeileis. CRAN task views. *R News*, 5(1):39–40, 2005. URL https://CRAN.R-project.org/doc/Rnews/. [p1]

Y. Zuo and R. Serfling. General notions of statistical depth function. *The Annals of Statistics*, 28 (2):461–482, 2000. [p2]

Francesca Ieva
MOX - Department of Mathematics
Politecnico di Milano
Italy
francesca.ieva@polimi.it

Anna Maria Paganoni
MOX - Department of Mathematics
Politecnico di Milano
Italy
anna.paganoni@polimi.it

Juan Romo
Department of Statistics
Universidad Carlos III de Madrid
Spain
juan.romo@uc3m.es

Nicholas Tarabelloni
MOX - Department of Mathematics

Politecnico di Milano
Italy
nicholas.tarabelloni@polimi.it