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_k's 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_k's 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")
```

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

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

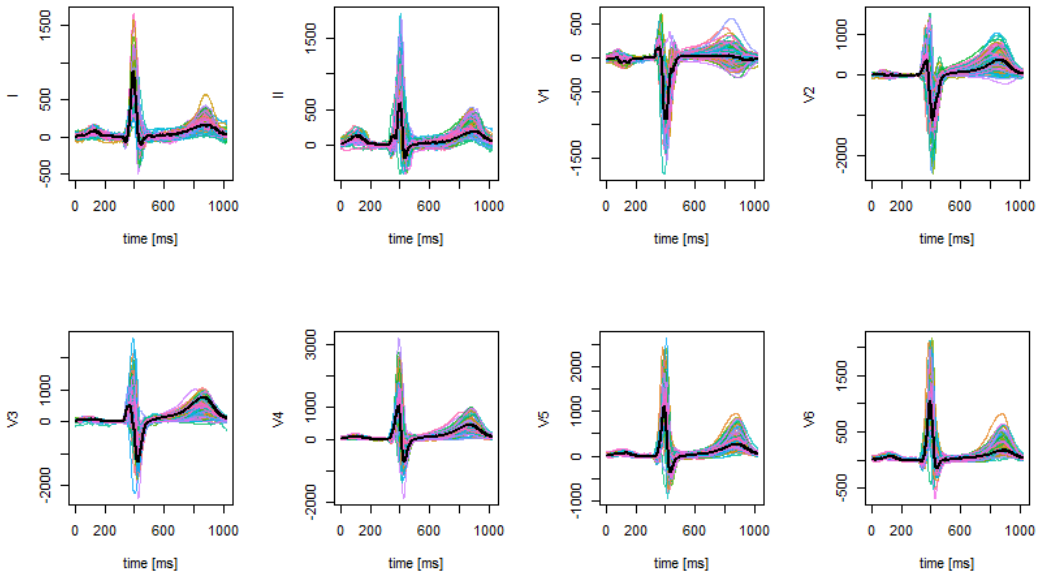

Figure 5: Plot of healthy data with superimposed the multivariate functional median computed maximizing the multivariate MBD (9) with uniform weights.

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:

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

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

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 α% 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 Robust Statistics. 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
```

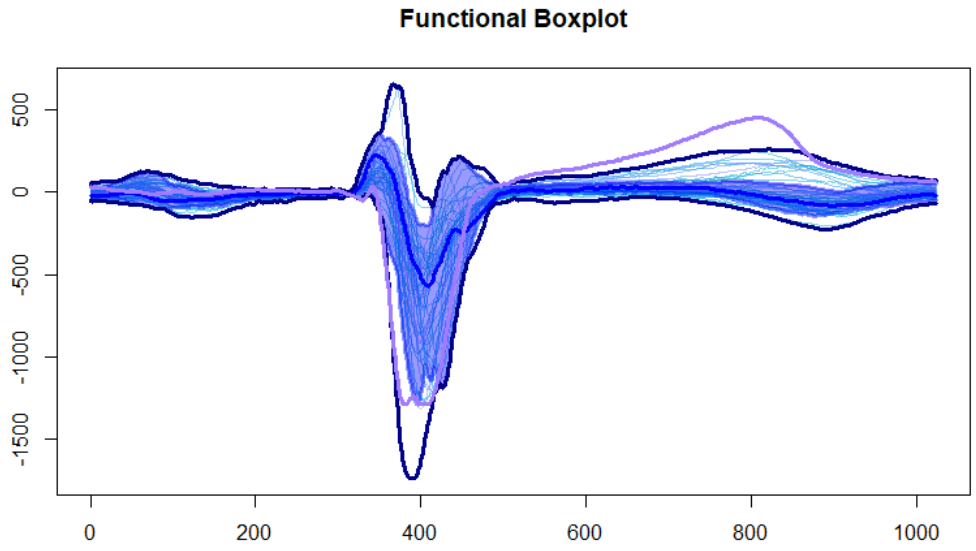

Figure 6: Functional Boxplot of the first lead, i.e., I of healthy data

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 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.

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

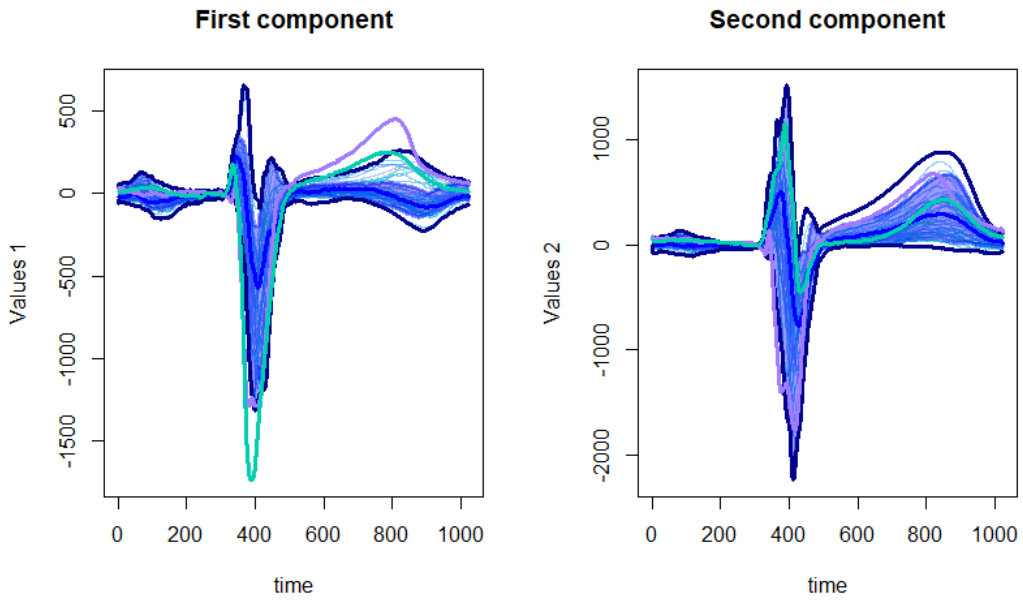

Figure 7: Functional Boxplot of the simulated data presented in Figure 4

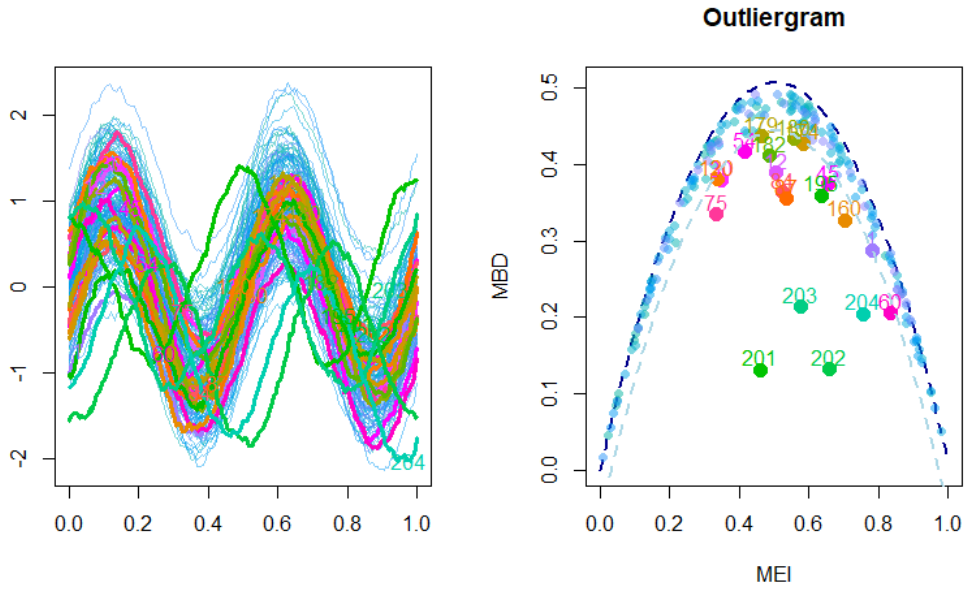

Figure 8: An example of outliergram on simulated univariate functional data

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 $\mathbf{f}_1, \ldots, \mathbf{f}_N$, $\mathbf{f}_j = (f_{j1}, ..., f_{jL})$. In Ieva and Paganoni (2017) the following inequality is proved:

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

where $a_0 = a_2 = -2/(n(n-1))$ and $a_1 = 2(n+1)/(n-1)$, and $MBD^J_{\{\mathbf{f_1},\ldots,\mathbf{f_n}\}}(\mathbf{f})$ and $MEI_{\{\mathbf{f_1},\ldots,\mathbf{f_n}\}}(\mathbf{f})$ 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
```

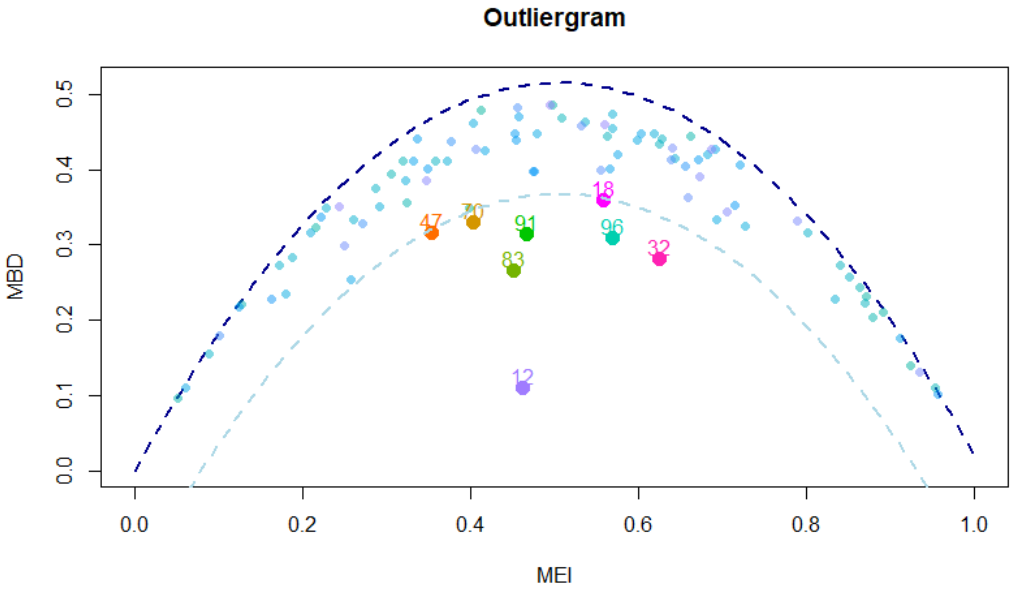

Figure 9: The outliergram of simulated data presented in Figure 4.

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. [p]

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

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. [p]