Advances in Data Analysis and Classification (2023) 17:463–484
https://doi.org/10.1007/s11634-022-00510-w

REGULAR ARTICLE

Band depth based initialization of K-means for functional data clustering

Javier Albert-Smet[1] · **Aurora Torrente**[2] · **Juan Romo**[1]

Received: 24 March 2022 / Revised: 21 July 2022 / Accepted: 25 July 2022 /
Published online: 3 September 2022

Abstract

The k-Means algorithm is one of the most popular choices for clustering data but is well-known to be sensitive to the initialization process. There is a substantial number of methods that aim at finding optimal initial seeds for k-Means, though none of them is universally valid. This paper presents an extension to longitudinal data of one of such methods, the BRIk algorithm, that relies on clustering a set of centroids derived from bootstrap replicates of the data and on the use of the versatile Modified Band Depth. In our approach we improve the BRIk method by adding a step where we fit appropriate B-splines to our observations and a resampling process that allows computational feasibility and handling issues such as noise or missing data. We have derived two techniques for providing suitable initial seeds, each of them stressing respectively the multivariate or the functional nature of the data. Our results with simulated and real data sets indicate that our *F*unctional Data *A*pproach to the BRIK method (FABRIk) and our *F*unctional *D*ata *E*xtension of the BRIK method (FDEBRIk) are more effective than previous proposals at providing seeds to initialize k-Means in terms of clustering recovery.

Keywords k-Means · Modified Band Depth · B-spline · functional data · bootstrap

Mathematics Subject Classification 62H30 · 62R10

✉ Aurora Torrente
etorrent@est-econ.uc3m.es

[1] Departamento de Estadística, Universidad Carlos III de Madrid, C/ Madrid 126, Getafe 28903, Madrid, Spain

[2] Departamento de Matemáticas, Instituto Gregorio Millán, Universidad Carlos III de Madrid, Av. Universidad 30, Leganés 28911, Madrid, Spain

1 Introduction

Amongst all non-hierarchical clustering algorithms, k-Means is the most widely used in every research field, from signal processing to molecular genetics. It is an iterative method that works by allocating each data point to the cluster with nearest gravity center until assignments no longer change or a maximum number of iterations is reached. Despite having a fast convergence to a minimum of the distortion –i.e., the sum of squared Euclidean distances between each data point and its nearest cluster center– (Selim and Ismail 1984), the method has well known disadvantages, including its dependence on the initialization process. If inappropriate initial points are chosen, the method can exhibit drawbacks such as getting stuck on a bad local minimum, converging more slowly or producing empty clusters (Celebi 2011). The existence of multiple local optima, which has proven to depend on the dataset size and on the overlap of clusters, greatly influences the performance of k-Means (Steinley 2006). Also, the method is known to be NP-hard (Garey and Johnson 1979); this has motivated numerous efforts to find techniques that provide sub-optimal solutions. Therefore, there are several methods for initializing k-Means with suitable seeds, though none of them is universally accepted; see He et al (2004); Steinley and Brusco (2007); Celebi et al (2013), for example.

Recently, the BRIk method (Bootstrap Random Initialization for k-Means) has been proposed in Torrente and Romo (2021) as a relevant alternative. This technique has two separate stages. In the first one, the input dataset is bootstrapped B times. k-Means is then run over these bootstrap replicates, with randomly chosen initial seeds, to obtain a set of cluster centers that form more compact groups than those in the original dataset. In the second stage these cluster centers are partitioned into groups and the deepest point of each of them is calculated. These prototypes are used as the initialization points for the k-Means clustering algorithm.

The BRIk method is flexible as it allows the user to make different choices regarding the number B of bootstrap replicates, the technique to cluster the bootstrap centers and the depth notion used to find the most representative seed for each cluster. There is a variety of data depth definitions, all of which allow generalizing the unidimensional median and rank to the multivariate or the functional data contexts. The main difference between them is that the latter models longitudinal data as continuous functions and thus incorporates much more information into the analysis. Furthermore, the study of functional data presenting "phase variation" –in addition to "amplitude variation"– can commonly benefit from the procedure known as curve registration or time warping. This aligns functions by identifying the timing of prominent features in the curves and transforms time so that these occur at the same instants of –transformed– time (Ramsay and Li 1998).

Over the last two decades there has been a substantial growth in the functional data analysis (FDA) literature, including techniques for cluster analysis (Ferreira and Hitchcock 2009; Jacques and Preda 2014), classification (López-Pintado and Romo 2003; Leng and Müller 2006), analysis of variance (Zhang 2013) and principal components analysis (Hall 2018), among others. In particular, the –computationally feasible– Modified Band Depth (MBD) (López-Pintado and Romo 2009) has been successfully applied to FDA for shape outlier detection (Arribas-Gil and Romo 2014), functional

boxplot construction (Sun and Genton 2011) or time warping (Arribas-Gil and Romo 2011); also, for BRIk, it is recommended to make use of the multivariate version of the MBD.

In this work we consider the FDA context and propose two alternative extensions of BRIk, that we have respectively called the Functional data Approach to BRIk (FABRIk) method and the Functional Data Extension of BRIK (FDEBRIk). The idea underlying both options is simple. We follow BRIk's pattern of bootstrapping and clustering a collection of tighter groups of centroids, but, at the initial stage, we additionally fit a continuous function to longitudinal data. Then, at a later phase, we return to a multivariate vector space by sampling functions at a number D of time points. This offers computational feasibility and several advantages over standard multivariate techniques, including the possibility of smoothing data to reduce noise or putting observations with missing features (time points) into the analysis. Note that the focus of this work is to present an innovative initialization method for K-Means and not a novel clustering algorithm. Within this framework, our approach can be classified as a filtering method, as designated by Jacques and Preda (2014).

The paper is organized as follows. Section 2 describes in detail our algorithms and the methods they will be compared to. Section 3 specifies the data, both simulated and real, and the quality measures used to assess FABRIk and FDEBRIk. Section 4 presents the overall results and in Sect. 5 we summarize our findings.

2 Methods

Consider a multivariate dataset $X = \{\mathbf{x}_1, \ldots, \mathbf{x}_N\}$ of N observations in d dimension, $\mathbf{x}_i = (x_{i1}, \ldots, x_{id})$, $i = 1, 2, ..., N$, that has to be partitioned into K clusters $G_1, \ldots, G_K$ by means of the k-Means algorithm (Forgy 1965). Once K initial seeds (centers) $\mathbf{c}_1, \ldots, \mathbf{c}_K$ have been obtained, k-Means works in the following way. Each data point $\mathbf{x}_i$ is assigned to exactly one of the clusters, G_{i_0}, where $i_0 = \arg\min_{1 \le j \le K} \mathrm{d}(\mathbf{x}_i, \mathbf{c}_j)$, with d the Euclidean distance. Next, for the j-th cluster, the center is updated by calculating the component-wise mean of elements in G_j. The assignment of elements to clusters and the computation of centers are repeated until there are no changes in the assignment or a maximum number of iterations is reached.

Since the k-Means output strongly depends on the initial seeds, there have been numerous efforts in the literature to obtain suitable initial centers for k-Means. In this work we focus on extending one of these methods, the BRIk algorithm, summarized as follows.

S1. FOR (b in $1 : B$) DO

- Obtain the b-th bootstrap sample X_b of the set X.
- Run k-Means, randomly initialized with K seeds, on X_b and store the final K centroids.

S2. Group the dataset of $K \times B$ centroids into K clusters, using some non-hierarchical algorithm.

S3. Find the deepest point of each cluster from step S2, using some data depth notion; these are used as initial seeds of k-Means.

Steps S2 and S3 grant flexibility to BRIk. Specifically, the method is designed to use any clustering technique in step S2. Here we use the Partitioning Around the Medoids (PAM) method (Kaufman and Rousseeuw 1990), though the Ward algorithm (Ward Jr 1963) also exhibited a good performance in the experiments (Torrente and Romo 2021). Also, in step S3 it is recommended to use the MBD depth notion (López-Pintado and Romo 2009), whose multivariate version for bands formed by pairs of distinct elements of X is given by the expression

$$MBD(\mathbf{x}_i) = \frac{1}{d \times \binom{N}{2}} \sum_{1 \le i_1 < i_2 \le N} \sum_{n=1}^{d} I_{\{\min\{x_{i_1 n}, x_{i_2 n}\} \le x_{in} \le \max\{x_{i_1 n}, x_{i_2 n}\}\}},$$

where I_A is the indicator function of set A. Thus, the BRIk algorithm's third step relies on finding the MBD-deepest point of each cluster (center grouping) from S2.

In this work we adapt BRIk to the case where data come from function observations to provide better initial seeds for k-Means by taking advantage of the nature of the data. In particular, we first take B-splines to approximate the original data in the least squares sense. This process sets a basis of (continuous) piecewise polynomial functions of a given degree and constructs the linear combination of these that best fits the data, to provide an approximation to the original function that is continuous and differentiable to a certain order, with all its advantages.

At this stage, we propose two variants, which respectively enhance the multivariate or functional properties of the original data. The differences between both versions can be seen through the left and right paths in Fig. 1.

The most straightforward technique, FABRIk (left path in Fig. 1), takes a plain computational perspective and evaluates the functions obtained on new time points (on a grid as dense as desired) to get a new dataset in D dimension. Then, these resampled (multivariate) data are input to the BRIk algorithm to find the initial seeds of k-Means.

Alternatively, relevance can be given to the functional nature of the data by accounting for the analytical expression of the fitted curves to compute distances between them (or their derivatives). In that case, to develop the FDEBRIk method (right path in Fig. 1), the step S1 in BRIk can be taken with the only modification of replacing k-Means by any technique suited for clustering functional data. In particular, we suggest to use the k-mean alignment (kma) method for curve clustering (Sangalli et al 2010). This technique generalizes k-Means by allowing curve alignment to a given template in an iterative process of template identification –corresponding to cluster centroids–; assignment and (possible) alignment of curves to such templates; and normalization to avoid cluster drifting. In the assignment and alignment step, if necessary, phase variability is removed by means of appropriate warping functions, while amplitude variability is accounted for with some similarity measure. In particular, in kma similarity between two (continuous, differentiable) functions x_i and x_j can be measured

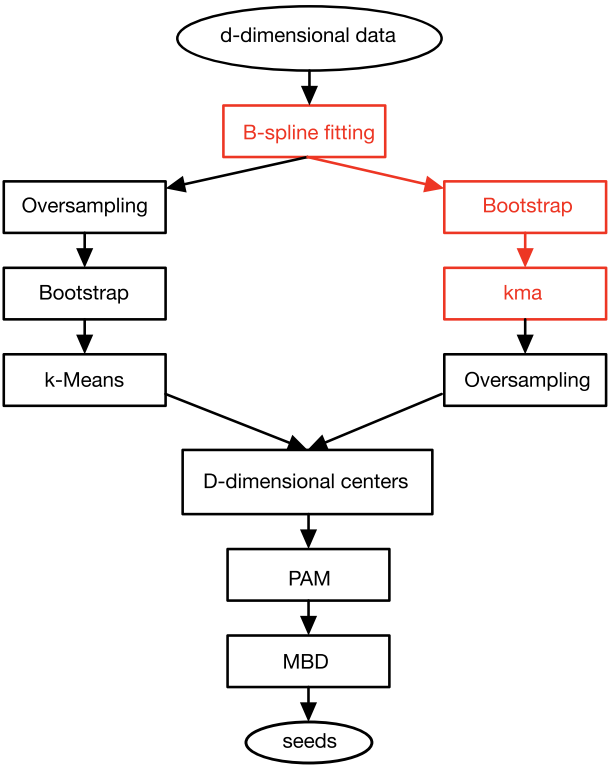

Fig. 1 Workflow of the FABRIk (left path) and FDEBRIk (right path) methods. Black (red) boxes indicate procedures whose output is multivariate (functional) data. The three first steps of FABRIk are the ones used in FKM and FKMPP (color figure online)

by the cosines of the angles between the functions

$$\rho_0(x_i, x_j) = \frac{\int_{\mathbb{R}} x_i(s)x_j(s)ds}{\sqrt{\int_{\mathbb{R}} x_i^2(s)ds}\sqrt{\int_{\mathbb{R}} x_j^2(s)ds}}, \tag{1}$$

or between their derivatives

$$\rho_1(x_i, x_j) = \frac{\int_{\mathbb{R}} x_i'(s)x_j'(s)ds}{\sqrt{\int_{\mathbb{R}} (x_i'(s))^2ds}\sqrt{\int_{\mathbb{R}} (x_j'(s))^2ds}}, \tag{2}$$

which are efficiently evaluated for B-splines. The kma process stops when the increments in the similarities are below a given threshold, while the initial templates (seeds) are chosen at random among the original curves.

Since this work focuses on finding suitable multivariate seeds to be used straightaway in k-Means, we have not considered the use of time-warping, as this would require to modify the original observations at each iteration too, let alone the increase

of computational cost. Thus we leave the integration of our methodology with time-warping to address phase variation for a future study.

The FDEBRIk method, then, retrieves from the B runs of kma the non-aligned, functional centroids corresponding to each bootstrap sample and projects them into a D-dimensional (multivariate) vector space before proceeding with steps S2 and S3, which will provide the final collection of initial seeds for k-Means. In the cases where we need to specify which of the similarity measures (1) or (2) we are using, we will write FDEBRIk$_0$ or FDEBRIk$_1$, respectively.

To check the performance of our methods we have selected several initialization techniques. First, as benchmark, the classical Forgy approach (Forgy 1965), where the initial seeds are selected at random; we refer to this as the KM initialization.

Next, we have considered a widely-used algorithm, k-Means++ (KMPP) (Arthur and Vassilvitskii 2007), which aims at improving the random selection of the initial seeds in the following way. A random data point $\mathbf{c}_1$ is firstly picked from the data set. This conditions the selection of the remaining initial seeds $\mathbf{c}_j$, $j = 2, \ldots, K$, which are sequentially chosen among the remaining observations $\mathbf{x}$ according to a probability proportional to $\mathbb{d}^2(\mathbf{x}, \mathbf{c}_{j_x})$, the squared Euclidean distance between the point $\mathbf{x}$ and its closest seed $\mathbf{c}_{j_x}$, where $j_x = \arg\min_{1 \leq i < j-1} \mathbb{d}(\mathbf{x}, \mathbf{c}_i)$. Following this procedure, the initial centers are typically separated from each other and yield more accurate groupings.

Additionally, in order to assess the potential improvement of our method, the functional approximation stage of the FABRIk and FDEBRIk methods is added before the KM and the KMPP methods are run. That is, we take the B-spline fitting and the oversampling steps and use KM and KMPP on a new D-dimensional data set. This way we can make a complete and fair comparison of how the FDA approach affects the BRIk method against how it improves KM or KMPP.

Hence, on the one hand, we will compare eight different k-Means initialization techniques: KM and its FDA version –with B-spline fitting and oversampling– (designated as FKM); KMPP and KMPP with the functional approximation (denoted by FKMPP); and BRIk and its functional variants, FABRIk and FDEBRIk with ρ_0 and ρ_1. On the other hand, we want to compare our strategy of replacing the d-dimensional data by those estimated in D dimension against the popular proposal of clustering the B-splines coefficients (Abraham et al 2003). This is because two functions in the same cluster are expected to have similar vectors of coefficients, and also because in most situations these vectors are considerably smaller in size than the D-dimensional ones. Therefore, we ran k-Means on the set of computed coefficients, initialized with KM, KMPP and BRIk. We will refer to these approaches as C-KM, C-KMPP and C-BRIk.

In order to explore the advantages of our method, we have also carried out experiments where the data points had missing observations, what translates into "sparse data" in the functional data context. For each simulated dataset we randomly removed a proportion $p \in \{0.1, 0.25, 0.5\}$ of the coordinates of each d-dimensional vector. For FKM, FKMPP, FABRIk, FDEBRIk and the methods based on clustering the B-spline coefficients, we estimated each missing value in a given vector by means of the corresponding B-spline. For KM, BRIk and KMPP, we imputed the missing data by using linear interpolation; note that, for simplicity, the removal of coordinates was hence

not applied to the first and last values. We then performed the analysis of the resulting data with each of the eleven methods mentioned.

3 Experimental setup

Our experiments were carried out in the R statistical language (R Core Team 2017), using the implementation of k-Means included in the `stats` package; the fitting of B-splines in the `splines2` package; the kma algorithm in the `fdakma` package; and the MBD coding provided in the `depthTools` package (Torrente et al 2013). For each dataset, the number of clusters in the k-Means algorithm was set to be equal to the number of groups and the maximum number of iterations was set to default (10). For BRIk, FABRIk and FDEBRIk, we used bootstrap sizes of $B = 25$. Using a larger bootstrap size decreases the speed of these algorithms, while slightly improving the distortion. Cubic B-splines with no intercept and a varying number of equally spaced knots, depending on the model to be analyzed, were chosen to approximate our data; then new evenly spaced observations were obtained by using an oversampling process with different oversampling factors. An oversampling factor of m means that the number D of time points observed in the approximated function is m times the number of original input samples: $D = m \times d$. The knots are defined through the degrees of freedom (DF) parameter. A DF value of n with cubic B-splines implementation indicates that $n - 3$ internal knots are placed uniformly in the horizontal axis. The resemblance of the approximated function to the real one in each of the models is determined by the DF parameter of the B-splines. Finally, as mentioned before, kma was used with no alignment.

3.1 Datasets

We conducted experiments involving simulated and real datasets. For the simulated ones we chose the four models described in Table 1; the functions giving origin to each of the clusters are shown in Fig. 2. Models 1 and 2 consider polynomial and sinusoidal functions; the former is designed to assess the effect of rapidly changing signals on the clustering quality whereas the latter could be used, for instance, to mimic monthly average temperatures in different climates. Model 3 consists of (raw and transformed) Gaussian functions and is used to test the impact of sudden peaks on signal clustering. Finally, Model 4, taken from Leroy et al (2018), attempts to model swimmers' progression curves. The time vector (x coordinate) varies from model to model, while the number of simulated functions per cluster is 25 for all of them. To construct the clusters, additive white Gaussian noise, whose intensity is described by the standard deviation σ, is incorporated to each model to mimic the randomness in the data collection process.

The DF parameter requires a careful choice for each model. Higher values of this parameter can account for larger variations of a function, and therefore Models 2 and 3 would require higher DFs. In our experiments, the specific value for each situation

Table 1 Description of the simulated models. The second column provides the time vector where the functions are observed, whereas the third column describes the signal defining each cluster, the values computed in a coordinate-wise manner. The fourth column includes the DFs used in each model, according to an elbow-like rule. The last column displays the SNR of each cluster corresponding to a level of noise $\sigma = 1$

Model	X-coordinates	Y-coordinates (signal)	DF	SNR (dB)
Model 1	$\mathbf{x} = (0, 0.01, 0.02, ..., 1)$	$\mathbf{y}_1 = \mathbf{x} - 0.5$	15	−10.79
		$\mathbf{y}_2 = (\mathbf{x} - 0.5)^2 - 0.8$		−2.84
		$\mathbf{y}_3 = -(\mathbf{x} - 0.5)^2 + 0.7$		−4.14
		$\mathbf{y}_4 = 0.75 \cdot \sin(8\pi \cdot \mathbf{x})$		−5.51
Model 2	$\mathbf{x} = (0, 0.01, 0.02, ..., 1)$	$\mathbf{y}_1 = \mathbf{x}$	4	−4.84
		$\mathbf{y}_2 = 2 \cdot (\mathbf{x} - 0.5)^2 - 0.25$		−15.35
		$\mathbf{y}_3 = -2 \cdot (\mathbf{x} - 0.5)^2 + 0.3$		−13.98
		$\mathbf{y}_4 = 0.6 \cdot \sin(2\pi \cdot \mathbf{x} - 0.5)$		−7.45
Model 3	$\mathbf{x} = (-10, -9.9, -9.8, ..., 10)$	$\mathbf{y}_1 = \frac{1}{2\sqrt{2\pi}} \cdot e^{-\frac{\mathbf{x}^2}{2 \cdot 2^2}}$	13	−21.52
		$\mathbf{y}_2 = \frac{1}{\sqrt{2\pi}} \cdot e^{-\frac{(\mathbf{x}+2)^2}{2 \cdot 1^2}}$		−18.51
		$\mathbf{y}_3 = \frac{1}{\sqrt{2\pi}} \cdot e^{-\frac{(\mathbf{x}-2)^2}{2 \cdot 1^2}}$		−18.51
		$\mathbf{y}_4 = \frac{-1}{\sqrt{2\pi}} \cdot e^{-\frac{\mathbf{x}^2}{2 \cdot 1^2}} + 0.4$		−8.73
		$\mathbf{y}_5 = \frac{-2}{3\sqrt{2\pi}} \cdot e^{-\frac{\mathbf{x}^2}{2 \cdot 3^2}} + 0.4$		−10.05
Model 4	$\mathbf{x} = (0, 0.05, 0.1, ..., 1)$	$\mathbf{y}_1 = \mathbf{x} - 1$	4	−4.45
		$\mathbf{y}_2 = \mathbf{x}^2$		−7.55
		$\mathbf{y}_3 = \mathbf{x}^3$		−9.24
		$\mathbf{y}_4 = \sqrt{\mathbf{x}}$		−3.23

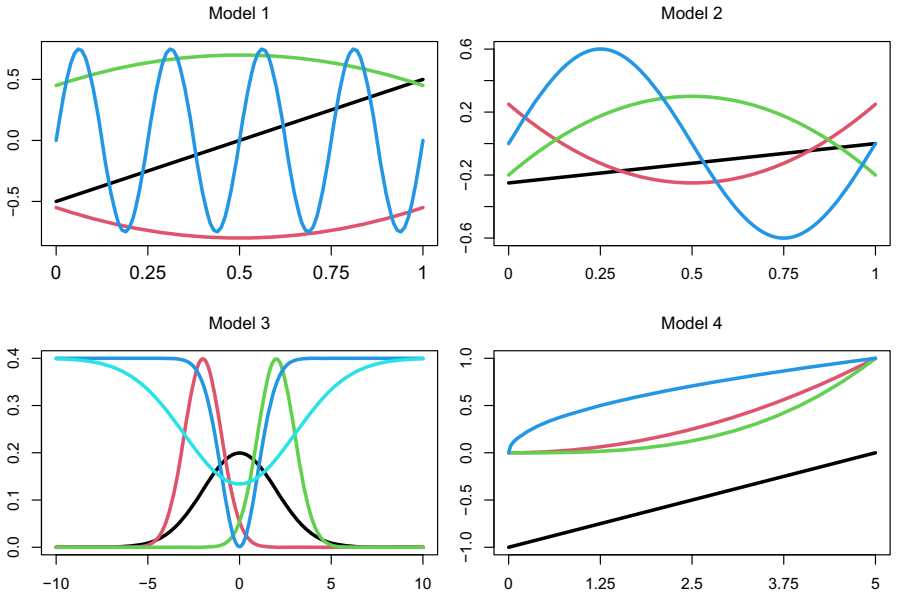

Fig. 2 Functions originating each of the clusters for the simulated models by adding Gaussian noise to each sampled component independently (color figure online)

was selected in the set {4,…, 50} according to an elbow-like rule for the plot of the (average) distortion against the DFs; these are provided in the fourth column of Table 1.

For each model, and using $\sigma = 1$, we generated 1000 independent datasets that were clustered with the eleven methods. To understand how this level of noise affects each particular model, we provide in the last column of Table 1 the signal-to-noise ratio (SNR), calculated from the classical signal processing definition as $10 \times \log_{10}(P_{signal}/P_{noise})$, where P_{signal} is the mean of the squared signal and $P_{noise} = \sigma^2$ (i.e., the noise variance). Nevertheless, in Fig. 2 it is apparent that, for each model, some clusters are more prone to confusion than others –see, for instance, clusters 2 (red) and 3 (green) of Model 4–, regardless of the particular SNR value.

Other values of σ ($\in \{0.5, 1.5, 2\}$) produced similar relative outputs; note that increasing the standard deviation to a value greater than $\sigma = 2$ renders a very poor cluster accuracy for every method tested; however, FABRIk, FDEBRIk$_0$ and FKMPP present slightly higher accuracy measures than the alternatives as the functional stage is noise-smoothing.

To complete the study of our algorithms we used real data to assess whether they are of practical use.

First, we have considered a dataset containing 200 electrocardiogram (ECG) signals, by a single electrode, 133 labeled as normal and 67 as abnormal (myocardial infarction), as formatted in Olszewski (2001) (units not provided). Each observation reflects a heartbeat and consists of 96 measurements. The dataset is available at the UCR Time Series Classification Archive (Dau et al 2018). See Fig. 3, left panel.

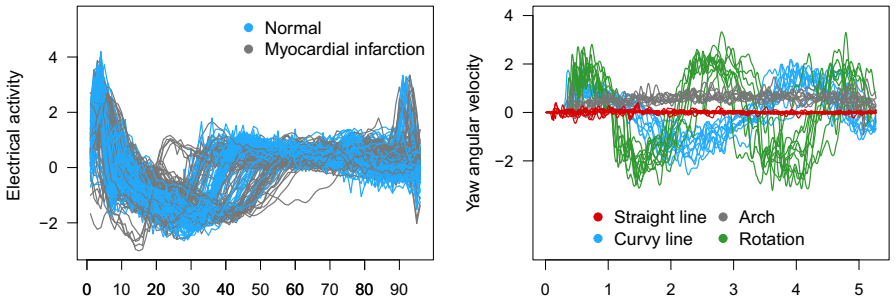

Fig. 3 Real data. Left panel: heart electrical activity recorded during a cardiac cycle for patients with a normal heart or with a cardiac condition; units not provided. Right panel: gyroscope yaw velocity readings (in rad/s) for four different patterns, registered at steps of $0.01s$ (color figure online)

Secondly, the Gyroscope dataset was recorded using a Xiaomi Pocophone F1. The mobile phone was laid on a table and moved to follow four patterns: a straight line, a curvy line, an arch and a rotation from side to side on the spot. The yaw angular velocity in rad/s was recorded using the Sensor Record app available in Google Play Store. This dataset was used to test the method's applicability to sensor data, specifically targeting a potential use case in robotic applications.

Each recording for each pattern was truncated to 527 registered time points, spaced by 10ms, in order for all data points to have the same length. Thus, their duration is approximately 5 seconds. See Fig. 3, right panel. The dataset consists of 11 recordings for each pattern and is available as Supplementary material.

Since all the methods tested in this study are random, in the sense that different runs produce in general distinct centroids, we ran each of them 1000 times for each dataset.

3.2 Performance evaluation

The overall performance of these methods has been evaluated according to five different measures that fall into four categories:

- Accuracy: We measure how similar the clusters are to the true groups by means of the Adjusted Rand Index (ARI) (Hubert and Arabie 1985) and the clustering correctness, which is computed as the percentage of label agreement (i.e. correctly assigned elements), according to the label permutation that yields the maximum set similarity.
- Dispersion: The obvious choice to determine how compact the clusters $G_1, \ldots, G_K$ are is the distortion $\sum_{j=1}^{K} \sum_{\mathbf{x}_i \in G_j} \mathrm{d}^2(\mathbf{x}_i, \mathbf{c}_j)$, where $\mathbf{c}_j$ is the gravity center of cluster G_j. Its evaluation is done by identifying each cluster from the partitioning labels and calculating the corresponding centroid, in the original (multivariate) data space.
- Convergence: We assess the convergence speed with the number of iterations required by the k-Means algorithm to converge after being initialized.

- Computational cost: Finally, we consider the execution time, in seconds, used by each algorithm from start to finish. Calculations are carried out with an Intel Core i7-6700HQ CPU with 2.60GHz and 8 GB RAM.

The performance of the methods we consider is assessed in terms of the median $\tilde{x}$, the mean $\bar{x}$ and the standard deviation s for all five measures.

4 Results

4.1 Simulated data

We used the four models to evaluate the performance of FABRIk and FDEBRIk in different situations. For Model 1, the FABRIk and FDEBRIk$_0$ methods –followed by BRIk and FDEBRIk$_1$– outperform the alternatives with respect to all the evaluation measures except for the execution time, as shown in Table 2, where statistics for the distortion have been rounded to four significant figures. In particular, it is clear that FDEBRIk has a much larger computational burden, specially when using the similarity ρ_1.

All the techniques based on clustering the vectors of coefficients are drastically worse than the other ones. The same situation is observed for all the scenarios we have considered (different models and levels of noise and presence or absence of missing data) and thus we do not include them in the subsequent tables for compactness.

Notably, in this model the variability of the first four measures is remarkably smaller for FABRIk and FDEBRIk$_0$. Here, the synthetic groups 2 and 3 are easily confounded and inappropriate initial seeds lead k-Means to merge these two groups into a single cluster and, consequently, to split one of the other groups into two clusters. This situation considerably reduces the ARI of the corresponding algorithms, whose distributions become bimodal. As an example, we considered a single dataset following Model 1, and compared the output of FABRIk and FDEBRIk, with $B = 25$, versus 25 runs of k-Means with random initialization. Our methods correctly allocate all the elements ($ARI = 1$), whereas none of 25 runs of standard k-Means is capable of retrieving the correct grouping (average $ARI = 0.9177$), with the confusion of clusters 2 and 3 in four of the runs.

Figure 4, upper panel, depicts violin plots of the ARI distributions; the ones corresponding to the correctness (not shown) display a similar pattern. The effect of wrong allocations is also reflected in the distribution of the distortion: all the methods except the ones based on bootstrap and MBD have bimodal densities or heavy upper tails, as shown in Fig. 4, bottom panel. This behavior is observed in a significant number of runs of the methods, but FABRIK and FDEBRIk$_0$ –followed by BRIk and FDEBRIk$_1$– seem to find the right partitioning more often.

Despite a median test does not reject the equality of medians for the accuracy measures (correctness and ARI), we conducted pairwise t-tests for testing equality of means. As expected from their asymmetric distributions, we obtained p-values lower than 10^{-43} for the comparison of FABRIk and FDEBRIk with the other methods, except for BRIk, with p-values in the order of 10^{-3} and 10^{-6}. In addition, when

Table 2 Summary statistics for Model 1. The median, mean and variance of 1000 independent datasets for the five performance evaluation measures are provided. Best and second-best medians and means are in boldface and italics, respectively

		Correctness	ARI	Distortion	Iterations	Exec. Time (s)
KM	$\tilde{x}$	**1.0000**	**1.0000**	9744	*2.000*	**0.0020**
	$\bar{x}$	0.9236	0.9137	9799	2.518	**0.0026**
	s	0.1428	0.1575	265.2	0.557	0.0027
KMPP	$\tilde{x}$	**1.0000**	**1.0000**	9755	*2.000*	*0.0060*
	$\bar{x}$	0.9100	0.8986	9821	2.455	*0.0066*
	s	0.1523	0.1681	276.2	0.542	0.0063
BRIk	$\tilde{x}$	**1.0000**	**1.0000**	**9683**	**1.000**	0.1021
	$\bar{x}$	0.9983	*0.9961*	**9687**	1.301	0.1048
	s	0.0120	0.0156	145.1	0.465	0.0198
FKM	$\tilde{x}$	**1.0000**	**1.0000**	9763	*2.000*	0.1533
	$\bar{x}$	0.8996	0.8886	9852	2.247	0.1603
	s	0.1619	0.1787	308.4	0.550	0.0328
FKMPP	$\tilde{x}$	**1.0000**	**1.0000**	9733	*2.000*	0.1956
	$\bar{x}$	0.9340	0.9269	9786	2.031	0.1967
	s	0.1355	0.1470	254.5	0.530	0.0204
FABRIk	$\tilde{x}$	**1.0000**	**1.0000**	*9684*	**1.000**	0.2908
	$\bar{x}$	*0.9992*	**0.9984**	**9687**	**1.034**	0.2916
	s	0.0029	0.0065	144.1	0.181	0.0226
FDEBRIk$_0$	$\tilde{x}$	**1.0000**	**1.0000**	*9684*	**1.000**	15.640
	$\bar{x}$	**0.9994**	**0.9984**	**9687**	*1.182*	15.530
	s	0.0024	0.0066	144.2	0.386	0.7075
FDEBRIk$_1$	$\tilde{x}$	**1.0000**	**1.0000**	9691	*2.000*	27.220
	$\bar{x}$	0.9918	0.9900	*9700*	2.166	27.490
	s	0.0506	0.0561	168.4	0.378	1.1690
C-KM	$\tilde{x}$	0.7700	0.6268	10290	3.000	0.1312
	$\bar{x}$	0.7741	0.6299	10300	3.155	0.1411
	s	0.0791	0.0924	252.4	0.676	0.0370
C-KMPP	$\tilde{x}$	0.7700	0.6272	10290	3.000	0.1368
	$\bar{x}$	0.7750	0.6316	10290	3.116	0.1429
	s	0.0754	0.0856	240.2	0.691	0.0374
C-BRIk	$\tilde{x}$	*0.7800*	*0.6352*	10260	*2.000*	0.1521
	$\bar{x}$	0.7912	0.6432	10260	2.139	0.1600
	s	0.0664	0.0804	224.9	0.422	0.0396

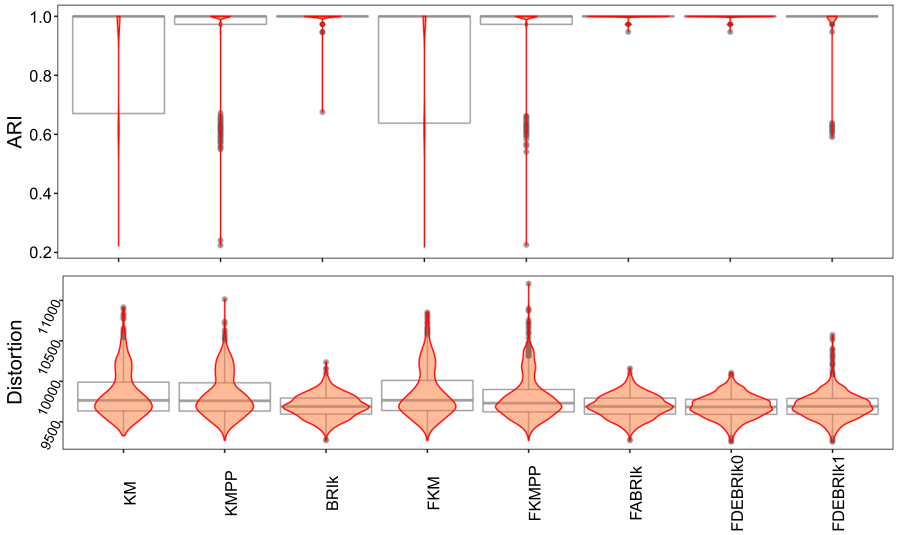

Fig. 4 Violin plots, in red, for the distribution of the ARI index (top) and the distortion (bottom) in Model 1, with $\sigma = 1$. The corresponding wider boxplots are superimposed in gray. BRIk, FABRIk and FDEBRIk have a consistent behaviour whereas the other methods have more spread or bimodal distributions, with heavy lower and upper tails, respectively (color figure online)

FDEBRIk_1 is compared to FABRIk and FDEBRIk_0, the p-values are also in the order of 10^{-6}, but the test leads to clearly not rejecting the null hypothesis for FABRIk versus FDEBRIk_0. As a consequence, the slight improvement in the correctness achieved by FDEBRIk_0 over FABRIk in this particular model does not compensate the increase in the computational cost.

With respect to the missing data case (sparse data), we report in Table 3 the results for a percentage $p = 25\%$ of missing data. Similar relative outputs can be observed in the other cases. FDEBRIk and FABRIk are again the best methods in terms of correctness and ARI, whereas FABRIk and BRIk need a lower number of iterations to reach convergence. With respect to the distortion, FABRIk is only slightly surpassed by FDEBRIk and BRIk. Regarding the execution time, KM, BRIk and KMPP required longer times than before due to the interpolation step, whilst no additional computation is needed for the functional approaches; hence the methods based on FDA are, globally, a more suitable option (with the same caveat as before regarding FDEBRIk's execution time).

In contrast to the previous case, in Models 2–4 FABRIk has in general a distortion slightly higher than that of KM, BRIk and KMPP but smaller than that of the other initialization methods with the functional approximation, as shown in Tables 4, 5 and Tables S1–S4 in the Suplementary material.

This difference in the distortion of multivariate and functional methods can be explained by accounting for the two different data spaces we are considering and our process of computing the distortion. In particular, from the point of view of the functional data space, FABRIk is simply the initialization of k-Means with appropriate seeds. However, from the point of view of the initial multivariate data space, the

Table 3 Summary statistics for Model 1 with 25% missing values. The median, mean and variance of 1000 independent datasets for the five performance evaluation measures are provided. Best and second-best medians and means are in boldface and italics, respectively

		Correctness	ARI	Distortion	Iterations	Exec. Time (s)
KM	$\tilde{x}$	*0.9900*	*0.9731*	8607	3.000	0.2620
	$\bar{x}$	0.9330	0.9152	8660	2.635	0.2693
	s	0.1297	0.1459	257.9	0.608	0.0334
KMPP	$\tilde{x}$	*0.9900*	*0.9731*	8608	3.000	0.2653
	$\bar{x}$	0.9383	0.9216	8650	2.564	0.2728
	s	0.1254	0.1408	244.3	0.557	0.0335
BRIk	$\tilde{x}$	**1.0000**	**1.0000**	**8572**	**1.000**	0.3569
	$\bar{x}$	0.9938	0.9835	**8571**	*1.469*	0.3662
	s	0.2495	0.0210	152.1	0.509	0.0398
FKM	$\tilde{x}$	**1.0000**	**1.0000**	8615	*2.000*	**0.1446**
	$\bar{x}$	0.9328	0.9193	8672	2.374	**0.1526**
	s	0.1344	0.1489	258.9	0.528	0.0301
FKMPP	$\tilde{x}$	**1.0000**	**1.0000**	8611	*2.000*	*0.1845*
	$\bar{x}$	0.9428	0.9307	8653	2.224	*0.1882*
	s	0.1238	0.1358	240.5	0.498	0.0244
FABRIk	$\tilde{x}$	**1.0000**	**1.0000**	*8574*	**1.000**	0.2757
	$\bar{x}$	*0.9957*	0.9886	*8573*	**1.133**	0.2807
	s	0.2203	0.0185	152.3	0.340	0.0314
FDEBRIk_0	$\tilde{x}$	**1.0000**	**1.0000**	**8572**	*2.000*	16.250
	$\bar{x}$	**0.9958**	**0.9961**	**8571**	1.536	16.210
	s	0.0064	0.0170	152.1	0.503	0.5668
FDEBRIk_1	$\tilde{x}$	**1.0000**	**1.0000**	**8572**	*2.000*	28.160
	$\bar{x}$	0.9824	*0.9924*	8580	2.374	28.220
	s	0.0474	0.0543	174.5	0.494	0.8180

clustering obtained with FABRIk does not necessarily (and not even frequently) correspond to a local minimum of the distortion in this space, therefore yielding higher values of the objective function. A similar explanation applies to the other functional techniques. On the contrary, FABRIk, FDEBRIk_0 and FDEBRIk_1 consistently provide remarkably higher accuracy measures, with FABRIk and FDEBRIk_0 getting a faster convergence in terms of the number of iterations. They also have longer execution times, as expected. However, FABRIk's ranking improves again if missing data are considered.

To assess the relevance of this increment in the computational cost we compared these results with the strategy of initializing k-Means several times and choosing the set of seeds providing the lowest distortion. For instance, in Model 2, KM with 200 random starts increases the average ARI from 0.4200 to 0.4549, which is far from the 0.6396, 0.6530 and 0.6315 obtained with our methods, and requires 0.1263 seconds on average, which is not far from FABRIK's execution time. This highlights

Table 4 Summary statistics for **Model 2**. The median, mean and variance of 1000 independent datasets for the five performance evaluation measures are provided. Best and second-best medians and means are in boldface and italics, respectively

		Correctness	ARI	Distortion	Iterations	Exec. Time (s)
KM	$\tilde{x}$	0.6600	0.4294	**9568**	4.000	$\sim$ **0**
	$\bar{x}$	0.6524	0.4200	**9568**	3.721	**0.0010**
	s	0.0709	0.0856	141.9	0.771	0.0026
KMPP	$\tilde{x}$	0.6600	0.4265	9684	4.000	*0.0030*
	$\bar{x}$	0.6527	0.4193	9687	3.701	*0.0040*
	s	0.0711	0.0864	144.1	0.737	0.0044
BRIk	$\tilde{x}$	0.6700	0.4369	*9575*	**2.000**	0.0828
	$\bar{x}$	0.6596	0.4315	*9573*	3.484	0.0885
	s	0.0668	0.0853	142.4	0.818	0.0221
FKM	$\tilde{x}$	0.8100	0.6109	9659	*3.000*	0.1046
	$\bar{x}$	0.7762	0.6113	9659	2.795	0.1068
	s	0.0976	0.0882	142.6	0.594	0.0124
FKMPP	$\tilde{x}$	0.8000	0.6103	9656	*3.000*	0.1089
	$\bar{x}$	0.7730	0.6099	9659	2.632	0.1140
	s	0.0992	0.0903	142.8	0.599	0.0261
FABRIk	$\tilde{x}$	*0.8300*	*0.6416*	9650	**2.000**	0.1862
	$\bar{x}$	*0.8198*	*0.6396*	9651	**1.949**	0.1955
	s	0.0606	0.0754	142.4	0.335	0.0385
FDEBRIk_0	$\tilde{x}$	**0.8400**	**0.6549**	9651	**2.000**	15.640
	$\bar{x}$	**0.8328**	**0.6530**	9651	*2.194*	15.530
	s	0.0515	0.0706	142.1	0.434	0.7075
FDEBRIk_1	$\tilde{x}$	*0.8300*	0.6343	9651	*3.000*	26.210
	$\bar{x}$	0.8063	0.6315	9653	2.551	26.150
	s	0.0767	0.0815	142.3	0.544	0.8799

the suitability, most of all, of FABRIk for the common situation in which optimality in terms of an external quality measure (i.e., high values of correctness or ARI) is not necessarily in agreement with optimality in terms of an internal quality measure (low values of distortion). Nevertheless, if computational cost is not a critical issue, FDEBRIk surpasses FABRIk in many situations.

In these models, all the pairwise tests for equality of means and medians for correctness, ARI and distortion, to compare FABRIk and FDEBRIk with the other methods and between them, yielded low p-values with an order of magnitude smaller than 10^{-9}. In summary, we can report a significant improvement over the alternatives.

Table 5 Summary statistics for Model 2 with 25% missing values. The median, mean and variance of 1000 independent datasets for the five performance evaluation measures are provided. Best and second-best medians and means are in boldface and italics, respectively

		Correctness	ARI	Distortion	Iterations	Exec. Time (s)
KM	$\tilde{x}$	0.6400	0.3825	*8427*	4.000	0.2170
	$\bar{x}$	0.6380	0.3810	**8429**	3.653	0.2248
	s	0.0674	0.0787	149.9	0.716	0.0312
KMPP	$\tilde{x}$	0.6400	0.3817	**8426**	4.000	0.2200
	$\bar{x}$	0.6333	0.3782	**8429**	3.651	0.2288
	s	0.0655	0.0749	149.1	0.740	0.0327
BRIk	$\tilde{x}$	0.6500	0.3973	8430	*3.000*	0.2974
	$\bar{x}$	0.6434	0.3948	*8432*	3.336	0.3088
	s	0.0594	0.0736	149.9	0.829	0.0381
FKM	$\tilde{x}$	0.7500	0.5245	8517	*3.000*	**0.1101**
	$\bar{x}$	0.7344	0.5270	8514	2.937	**0.1110**
	s	0.0829	0.0776	153.1	0.661	0.0122
FKMPP	$\tilde{x}$	0.7400	0.5182	8518	*3.000*	*0.1108*
	$\bar{x}$	0.7260	0.5219	8516	2.776	*0.1164*
	s	0.0864	0.0785	152.5	0.615	0.0218
FABRIk	$\tilde{x}$	*0.7800*	*0.5537*	8513	**2.000**	0.1896
	$\bar{x}$	*0.7690*	*0.5516*	8508	**2.001**	0.1967
	s	0.0669	0.0753	151.3	0.324	0.0306
FDEBRIk_0	$\tilde{x}$	**0.7900**	**0.5696**	8513	**2.000**	16.250
	$\bar{x}$	**0.7867**	**0.5673**	8509	*2.312*	16.210
	s	0.0680	0.0708	151.2	0.523	0.5668
FDEBRIk_1	$\tilde{x}$	*0.7800*	0.5505	8514	*3.000*	25.685
	$\bar{x}$	0.7644	0.5494	8511	2.696	25.982
	s	0.0726	0.0763	151.8	0.613	1.3870

4.2 Real data

We next applied all the initialization methods to the real data.

For the ECG dataset, the DFs were set to 15 according to the elbow rule and we chose an oversampling factor of 1 for speed, as using a denser time grid produces a similar output. Table 6 summarizes our results.

Note that the quality of the clustering recovery in terms of ARI is low. Except for FDEBRIk_1, whose performance values are clearly the worst, we do not find prominent differences across methods. In particular, all of them require a single iteration to converge and all except FDEBRIk_1 have the same median for correctness, ARI and distortion. Yet, FDEBRIk_0 leads to the best average correctness, ARI and distortion, followed by BRIk and FABRIk, and has the smallest standard deviations. This corresponds to a single-mode distribution: a scenario similar to that depicted in Fig. 4 for

Table 6 Summary statistics for the ECG dataset. The median, mean and variance of 1000 runs of each initialization method for the five performance evaluation measures are provided. Best and second-best medians and means are in boldface and italics, respectively

		Correctness	ARI	Distortion	Iterations	Exec. Time (s)
KM	$\tilde{x}$	**0.7450**	**0.2194**	**5117**	**1.000**	**0.0010**
	$\bar{x}$	0.7307	0.1877	5370	**1.000**	**0.0008**
	s	0.0209	0.0465	371.6	0.000	0.0009
KMPP	$\tilde{x}$	**0.7450**	**0.2194**	**5117**	**1.000**	*0.0020*
	$\bar{x}$	0.7278	0.1812	5422	**1.000**	*0.0024*
	s	0.0219	0.0485	388.0	0.000	0.0022
BRIk	$\tilde{x}$	**0.7450**	**0.2194**	**5117**	**1.000**	0.0509
	$\bar{x}$	*0.7374*	*0.2025*	*5252*	**1.000**	0.0554
	s	0.0168	0.0374	299.3	0.000	0.0167
FKM	$\tilde{x}$	**0.7450**	**0.2194**	**5117**	**1.000**	0.3850
	$\bar{x}$	0.7326	0.1908	5364	**1.000**	0.3860
	s	0.0184	0.0428	370.0	0.000	0.0014
FKMPP	$\tilde{x}$	**0.7450**	**0.2194**	**5117**	**1.000**	0.3871
	$\bar{x}$	0.7304	0.1855	5410	**1.000**	0.3875
	s	0.0193	0.0446	385.7	0.000	0.0040
FABRIk	$\tilde{x}$	**0.7450**	**0.2194**	**5117**	**1.000**	0.4459
	$\bar{x}$	*0.7374*	0.2018	5269	**1.000**	0.4484
	s	0.0157	0.0363	314.1	0.000	0.0129
FDEBRIk_0	$\tilde{x}$	**0.7450**	**0.2194**	**5117**	**1.000**	31.706
	$\bar{x}$	**0.7400**	**0.2078**	**5217**	**1.000**	31.753
	s	0.0132	0.0306	264.8	0.000	1.3815
FDEBRIk_1	$\tilde{x}$	*0.7050*	*0.1268*	*5917*	**1.000**	60.097
	$\bar{x}$	0.7090	0.1361	5837	**1.000**	60.268
	s	0.0140	0.0378	270.2	0.000	3.0197

simulated data. As usual, FABRIk and FDEBRIk are largely outperformed in terms of execution time by those methods that do not rely on the B-spline approximation.

For the Gyroscope dataset, the DFs were set to 15, once more, according to the elbow rule. The oversampling factor was set to 1. Again, a similar performance is observed for higher values of this parameter, which does not influence the final results.

In contrast to the previous case, the values of correctness and ARI are much higher, as shown in Table 7. However, the FABRIk method finds more accurate groups, obtaining ARI values larger than 0.9 in roughly 60% of the iterations, whereas for instance, this percentage is around 35% and 20% for KM and FKMPP, respectively. Also, for distortion, it achieves low values (after FDEBRIk and BRIk) and shows, after FDEBRIk_1, the least variability, followed by BRIk and FDEBRIk_0. In fact, for BRIk, FABRIk and FDEBRIk_0, the accuracy and dispersion measures have bi-modal distributions, while for FDEBRIk_1 these values are distributed around a single mode, which explains the lowest value of the variance. However, this mode is clearly not the best local minimum.

Table 7 Summary statistics for the Gyroscope dataset. The median, mean and variance of 1000 runs of each initialization method for the five performance evaluation measures are provided. Best and second-best medians and means are in boldface and italics, respectively

		Correctness	ARI	Distortion	Iterations	Exec. Time (s)
KM	$\tilde{x}$	*0.6591*	*0.6154*	*4645*	**2.000**	**0.0010**
	$\bar{x}$	0.7565	0.7126	4717	2.037	**0.0014**
	s	0.1726	0.1830	636.8	0.334	0.0007
KMPP	$\tilde{x}$	*0.6591*	*0.6154*	**3929**	**2.000**	*0.0050*
	$\bar{x}$	0.6887	0.6387	4483	1.899	*0.0055*
	s	0.1535	0.1729	698.2	0.373	0.0019
BRIk	$\tilde{x}$	*0.6591*	*0.6154*	**3929**	**2.000**	0.4644
	$\bar{x}$	*0.8067*	*0.7650*	4261	**1.570**	0.4724
	s	0.1588	0.1609	356.9	0.495	0.0582
FKM	$\tilde{x}$	*0.6591*	*0.6154*	*4645*	**2.000**	0.2509
	$\bar{x}$	0.7727	0.7291	4753	2.003	0.2514
	s	0.1810	0.1920	610.1	0.345	0.0008
FKMPP	$\tilde{x}$	*0.6591*	*0.6154*	**3929**	**2.000**	0.2549
	$\bar{x}$	0.7090	0.6615	4429	1.820	0.2554
	s	0.1541	0.1688	636.9	0.426	0.0016
FABRIk	$\tilde{x}$	**0.9773**	**0.9379**	*4645*	**2.000**	0.7123
	$\bar{x}$	**0.8471**	**0.8060**	4352	*1.613*	0.7190
	s	0.1565	0.1586	351.9	0.487	0.0556
FDEBRIk$_0$	$\tilde{x}$	*0.6591*	*0.6154*	**3929**	**2.000**	25.654
	$\bar{x}$	0.7980	0.7559	*4258*	1.775	25.666
	s	0.1595	0.1626	373.5	0.420	0.8900
FDEBRIk$_1$	$\tilde{x}$	*0.6591*	*0.6154*	**3929**	**2.000**	33.512
	$\bar{x}$	0.6794	0.6356	**4021**	1.876	33.563
	s	0.0822	0.0852	286.5	0.332	0.9857

On the contrary, the distributions corresponding to the other algorithms present three or more peaks, what highlights the more consistent performance of the techniques based on bootstrap and MBD. On the other hand, the computational cost of BRIk, FABRIk and FDEBRIk are the largest ones. With respect to the number of iterations, all methods have similar values, with BRIk and FABRIk slightly better on average.

4.3 Implementation

We have implemented an R package, `briKmeans`, to provide the basic tools to run BRIk, FABRIk and FDEBRIk, with ρ_0 and ρ_1. Users can tune the different parameters of the methods through the functions parameters and retrieve the corresponding initial seeds and the resulting k-Means output, which includes the partitioning of the data set. For instance, the following simple call

```
> fabrik(exampleM1, k=4, degFr=10)
```

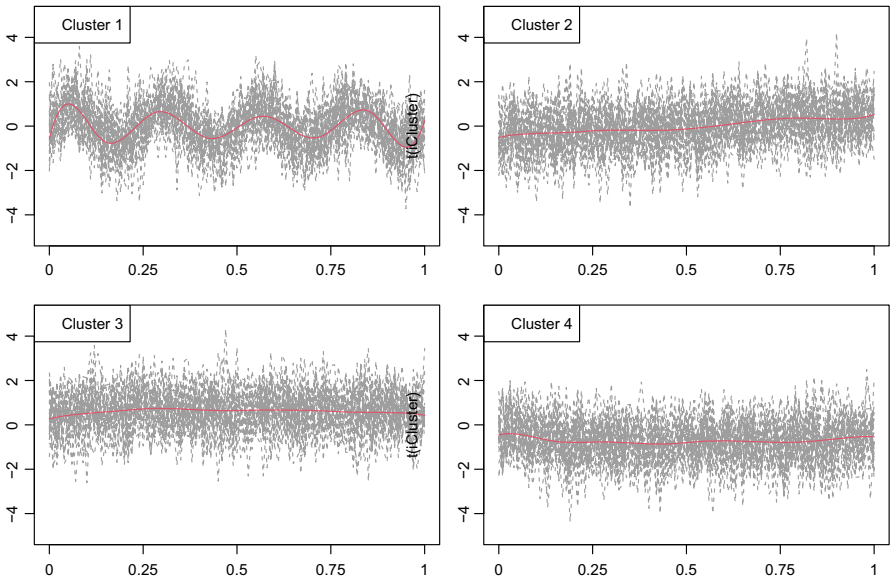

Fig. 5 Representation of the four clusters retrieved by FABRIk for a dataset following Model 1 with $\sigma = 1$, along with the final centroid (solid line). Our method correctly allocates all the elements ($ARI = 1$) (color figure online)

will run FABRIk with DFs set to 10 and the rest of parameters set to default, and return `k=4` clusters for the dataset `exampleM1`. The clusters can be visualized individually in parallel coordinates (Inselberg 1985) by means of the `plotKmeansClustering` function, including the final centroids. In Fig. 5 we illustrate this representation for a dataset following Model 1, with $\sigma = 1$. Note that users can also turn to the `elbowRule` function to plot the distortion associated to FABRIk or FDEBRIk against the DFs in order to optimize this parameter.

5 Conclusion

In this work we have developed FABRIk and FDEBRIk, two initialization methods for k-Means that extend the BRIk algorithm to the functional data case at two different levels. Both take d-dimensional longitudinal observations from continuous functions as an input dataset and return the D-dimensional initial seeds for k-Means after a functional approximation process via B-splines and a re-sampling stage. The difference between them is the step at which the re-sampling is carried out and, thus, the extent of use of the functional nature of the initial data.

Similarly to their precursor BRIk, our methods are flexible in several ways. The number of bootstrap replicates B can be tuned by the user; in general, low values of B are enough to produce a relevant improvement over the alternatives. Additionally, the oversampling factor m and the DFs can be chosen to best adapt to the data. An oversampling factor of 1 has proven to yield similar results to higher values of this

parameter, while remaining less computationally expensive. The DFs are selected according to the elbow rule. Nevertheless, our experiments show that a wide range of values for these parameters are also suitable. The clustering algorithm used to partition the cluster centers is an extra feature that can be determined by the user. In particular, for the FDEBRIk method we have chosen the kma algorithm, which offers an additional degree of freedom through the selection of the similarity measure between functions. We have considered the measures ρ_0 and ρ_1, which account for the cosines of the angles between the functions or their derivatives, respectively. Our study demonstrates that the performance of the method is consistently better when the similarity is based on the cosine of the angle between the functions, as opposed to the cosine of the angle between their derivatives. Finally, one could potentially use any feasible data depth definition, but our recommendation is to choose MBD for its fast computation and because it has proven to score high in the accuracy measures.

We have compared our functional initialization strategies to their multivariate version and to two more techniques, with and without the FDA approach. Furthermore, we have assessed the behavior of the methods based on clustering the B-splines coefficients obtained for each data point, which have proven to be poor competitors.

Generally speaking, FABRIk works well with both synthetic and real data, though FDEBRIk with ρ_0 commonly measures up to or surpasses it, with respect to the correctness, ARI and distortion. Only for the Gyroscope dataset FABRIk gets substantially better values of the distortion than FDEBRIk and each of the competitors. Nevertheless, FABRIK has a computational cost that is two orders of magnitude smaller than that of FDEBRIk$_0$, and competes with the other ones in the case of sparse data, which is a valuable aspect of the method. Thus, the major reason for choosing one against the other should be the computational cost. On the other hand, FDEBRIk with ρ_1 only ranks as the best option in one of the datasets that we have considered; in addition, its computational cost is markedly larger, making it the less suitable option of the methods that we have proposed. In summary, FABRIk and FDEBRIk$_0$ provide an advantageous solution that offers higher quality (in terms of clustering recovery, i.e., ARI and correctness) than other techniques at the cost of a longer computational time and, commonly, a slightly larger distortion. However, these aspects can be alleviated, respectively, by using a lower or higher value of B.

Additionally, we have shown that in some situations, and particularly with the real data we have considered, FABRIk and FDEBRIk$_0$ rise as more reliable ways of initializing k-Means, which consistently provide better accuracy results with lower variances. This reinforces the practical application of the methods for data analysis. Moreover, as any technique based on a functional approximation of the observations, they allow denoising and imputation of missing data.

Supplementary Information The online version contains supplementary material available at https://doi.org/10.1007/s11634-022-00510-w.

Author Contributions Javier Albert-Smet collected the real datasets, wrote the code for FABRIk and FDEBRIk, performed the computations and analysis of the results and contributed to the final manuscript. Aurora Torrente conceived the experiments, participated in the code writing process, supervised the findings of this work, interpreted the results and wrote the manuscript. Juan Romo helped supervise the project and provided valuable state-of-the-art insights.

Funding Open Access funding provided thanks to the CRUE-CSIC agreement with Springer Nature. This work was partially supported by the Spanish Ministry of Education [collaboration grant in university departments, Archive ID 18C01/003730] and the Spanish Ministry of Science, Innovation and Universities [grants numbers PID2020-116567GB-C22 and PID2020-112796RB-C22].

Availability of data and material The gyroscope dataset is available within the article as supplementary material. The electrocardiogram dataset is openly available at https://www.cs.ucr.edu/~eamonn/time series data 2018/.

Code availability We have made both methods publicly available through the R package `briKmeans` (on the CRAN repository), which also allows following the elbow-like rule for selecting suitable values of the DF parameter and representing the clusters, along with the final centroids, in parallel coordinates.

Declarations

Competing interests Authors of this manuscript have no financial or non-financial interests that are directly or indirectly related to the work submitted for publication.

Ethics approval Not applicable.

Consent to participate Not applicable.

Consent for publication The three authors consent to the publication of this manuscript.

References

Abraham C, Cornillon PA, Matzner-Løber E et al (2003) Unsupervised curve clustering using B-Splines. Scandinavian J Stat 30:581–595. https://doi.org/10.1111/1467-9469.00350

Arribas-Gil A, Romo J (2011) Robust depth-based estimation in the time warping model. Biostatistics 13(3):398–414. https://doi.org/10.1093/biostatistics/kxr037

Arribas-Gil A, Romo J (2014) Shape outlier detection and visualization for functional data: the outliergram. Biostatistics 15(4):603–619. https://doi.org/10.1093/biostatistics/kxu006

Arthur D, Vassilvitskii S (2007) k-means++: The advantages of careful seeding. Proceedings of the 18th Annual ACM-SIAM Symposium on Discrete Algorithms pp 1027–1035

Celebi ME (2011) Improving the performance of k-means for color quantization. Image Vis Comput 29(4):260–271. https://doi.org/10.1016/j.imavis.2010.10.002

Celebi ME, Kingravi HA, Vela PA (2013) A comparative study of efficient initialization methods for the k-means clustering algorithm. Expert Syst Appl 40(1):200–210. https://doi.org/10.1016/j.eswa.2012.07.021

Dau HA, Keogh E, Kamgar K, et al (2018) The UCR Time Series Classification Archive. https://www.cs.ucr.edu/~eamonn/time_series_data_2018/

Ferreira L, Hitchcock DB (2009) A comparison of hierarchical methods for clustering functional data. Communications in Statistics - Simulation and Computation 38(9):1925–1949. https://doi.org/10.1080/03610910903168603

Forgy E (1965) Cluster analysis of multivariate data: efficiency versus interpretability of classifications. Biometrics 21:768–780

Garey MR, Johnson DS (1979) Computers and intractability: a guide to the theory of NP-completeness. Freeman, New York
Hall P (2018) Principal component analysis for functional data: Methodology, theory, and discussion. Oxford University Press Inc., New York. https://doi.org/10.1093/oxfordhb/9780199568444.013.8
He J, Lan M, Tan CL, et al (2004) Initialization of cluster refinement algorithms: a review and comparative study. In: IEEE International Joint Conference on Neural Networks, Budapest, Hungary, pp 297–302, https://doi.org/10.1109/IJCNN.2004.1379917
Hubert L, Arabie P (1985) Comparing partitions. J Classif 2:193–198. https://doi.org/10.1007/BF01908075
Inselberg A (1985) The plane with parallel coordinates. Vis Comput 1:69–91. https://doi.org/10.1007/BF01898350
Jacques J, Preda C (2014) Functional data clustering: a survey. Adv Data Anal Classif 8(3):231–255. https://doi.org/10.1007/s11634-013-0158-y
Kaufman L, Rousseeuw PJ (1990) Finding groups in data: an introduction to cluster analysis. Wiley, New York
Leng X, Müller HG (2006) Classification using functional data analysis for temporal gene expression data. Bioinformatics 22(1):68–76. https://doi.org/10.1093/bioinformatics/bti742
Leroy A, Marc A, Dupas O et al (2018) Functional data analysis in sport science: Example of swimmers' progression curves clustering. Appl Sci 8(10):1766. https://doi.org/10.3390/app8101766
López-Pintado S, Romo J (2003) Depth-based classification for functional data. In: Data Depth: Robust Multivariate Analysis, Computational Geometry and Applications, pp 103–119, https://doi.org/10.1090/dimacs/072/08
López-Pintado S, Romo J (2009) On the concept of depth for functional data. J Am Stat Assoc 104(486):718–734. https://doi.org/10.1198/jasa.2009.0108
Olszewski R (2001) Generalized feature extraction for structural pattern recognition in time-series data. PhD thesis, School of Computer Science, Carnegie Mellon University
R Core Team (2017) R: a language and environment for statistical computing. R Foundation for Statistical Computing, Vienna, Austria, https://www.R-project.org/
Ramsay JO, Li X (1998) Curve registration. Journal of the Royal Statistical Society: Series B (Statistical Methodology) 60(2):351–363. https://doi.org/10.1111/1467-9868.00129
Sangalli LM, Secchi P, Vantini S et al (2010) K-mean alignment for curve clustering. Comput Stat Data Anal 54(5):1219–1233. https://doi.org/10.1016/j.csda.2009.12.008
Selim SZ, Ismail MA (1984) K-means type algorithms: a generalized convergence theorem and characterization of local optimality. IEEE Transactions on Pattern Analysis 6(1):81–87. https://doi.org/10.1109/TPAMI.1984.4767478
Steinley D (2006) Profiling local optima in k-means clustering: developing a diagnostic technique. Psychol Methods 11(2):178–192. https://doi.org/10.1037/1082-989X.11.2.178
Steinley D, Brusco MJ (2007) Initializing k-means batch clustering: a critical evaluation of several techniques. J Classif 24:99–121. https://doi.org/10.1007/s00357-007-0003-0
Sun Y, Genton M (2011) Functional boxplots. J Comput Graph Stat 20(2):316–334. https://doi.org/10.1198/jcgs.2011.09224
Torrente A, Romo J (2021) Initializing k-means clustering by bootstrap and data depth. J Classif 38(2):232–256. https://doi.org/10.1007/s00357-020-09372-3
Torrente A, López-Pintado S, Romo J (2013) DepthTools: an R package for a robust analysis of gene expression data. BMC Bioinformatics 14:237. https://doi.org/10.1186/1471-2105-14-237
Ward JH Jr (1963) Hierarchical grouping to optimize an objective function. J Am Stat Assoc 58(301):236–244. https://doi.org/10.1080/01621459.1963.10500845
Zhang JT (2013) Analysis of variance for functional data. Chapman and Hall, New York