



HAL
open science

Clustering multivariate functional data using unsupervised binary trees

Steven Golovkine, Nicolas Klutchnikoff, Valentin Patilea

► **To cite this version:**

Steven Golovkine, Nicolas Klutchnikoff, Valentin Patilea. Clustering multivariate functional data using unsupervised binary trees. *Computational Statistics and Data Analysis*, 2022, 168, pp.article n°107376. 10.1016/j.csda.2021.107376 . hal-03391643

HAL Id: hal-03391643

<https://hal.science/hal-03391643v1>

Submitted on 28 Oct 2024

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

Clustering multivariate functional data using unsupervised binary trees

Steven Golovkine* Nicolas Klutchnikoff† Valentin Patilea‡

September 27, 2021

Abstract

We propose a model-based clustering algorithm for a general class of functional data for which the components could be curves or images. The random functional data realizations could be measured with error at discrete, and possibly random, points in the definition domain. The idea is to build a set of binary trees by recursive splitting of the observations. The number of groups are determined in a data-driven way. The new algorithm provides easily interpretable results and fast predictions for online data sets. Results on simulated datasets reveal good performance in various complex settings. The methodology is applied to the analysis of vehicle trajectories on a German roundabout.

Some key words— Gaussian mixtures; Model-based clustering; Multivariate Functional Principal Components.

1 Introduction

Motivated by a large number of applications ranging from sports to the automotive industry and healthcare, there is a great interest in modeling observation entities in the form of a sequence of possibly vector-valued measurements, recorded intermittently at several discrete points in time. Functional data analysis (FDA) considers such data as being values on the realizations of a stochastic process, recorded with some error, at discrete random times. The purpose of FDA is to study such trajectories, also called curves or functions. See, *e.g.*, [37, 50, 20] for some recent references. The amount of such data collected grows rapidly as does the cost of their labeling. Thus, there is an increasing interest in methods that aim to identify homogeneous groups within functional datasets.

Clustering procedures for functional data have been widely studied in the last two decades, see, for instance, [10, 11, 12, 7, 26] and references therein. See also [6] for a recent textbook.

*Groupe Renault & CREST - UMR 9194, Rennes, France, steven.golovkine@ensai.fr

†Univ Rennes, CNRS, IRMAR - UMR 6625, F-35000 Rennes, France, nicolas.klutchnikoff@univ-rennes2.fr

‡Ensai, CREST - UMR 9194, Rennes, France, valentin.patilea@ensai.fr

In particular, for Gaussian processes, [46] show that the cluster centers found with k -means are linear combinations of the eigenfunctions from the functional Principal Components Analysis (fPCA). A discriminative functional mixture model is developed by [5] for the analysis of a bike sharing system from cities around the world.

Algorithms built to handle multivariate functional data have gained much attention in the last few years. Some of these methods are based on k -means algorithm with a specific distance function adapted to multivariate functional data. See *e.g* [47, 23]. [28] consider Self-Organizing Maps built on the coefficients of the curves into orthonormalized Gaussian basis expansion. [27] developed model-based clustering for multivariate functional data. In their model, they assume a cluster-specific Gaussian distribution for the principal component scores. The eigen-elements are estimated using an approximation of the curves into a finite dimensional functional space and are cluster specific. [42] have recently extended the previous model by modeling all principal components whose estimated variances are non-null. The underlying model for these methods usually consider only amplitude variations. Unlike others, [34] present a specific model for functional data to consider phase variations. Finally, [48] propose a mix between dimension reduction and non-parametric approaches by deriving the envelope and the spectrum from the curves and have applied it to nuclear safety experiments.

We propose a clustering algorithm based on the recursive construction of binary trees to recover the groups. Our idea extends the CUBT method [16] to the functional setting, and we therefore call it **fCUBT**. At each node of the tree, a model selection test is performed, after expanding the multivariate functional data into a well chosen basis. Similarly to [36] and [8], using the Bayesian Information Criterion (BIC), we test whether there is evidence that the data structure is a mixture model or not. A significant advantage of our procedure compared to [27, 42], is its ability to estimate the number of groups within the data while this number have to be pre-specified in the other methods. Moreover, the tree structure allows us to consider only a small number of principal components at each node of the tree and not to estimate a global number of components for the clustering. Considering tree methods designed for functional data, [45] extend the popular Isolation Forest algorithm used for anomaly detection, to the functional context. Our **fCUBT** algorithm also allows for classes defined by certain types of phase variations.

The remainder of the paper is organized as follows. In Section 2, we define a model for a mixture of curves for multivariate functional data with the coordinates having possibly different definition domains. Given a dataset, that is a set of, possibly noisy, intermittent measures of an independent sample of realizations of the stochastic process, in Section 3, we explain how compute the different quantities that are required in the clustering procedure. In Section 4, we develop the construction of our clustering algorithm, named **fCUBT**. In Section 5, we study the behavior of **fCUBT** and compare its performance with competing methods both on simulated and real datasets. Our algorithm performs well to estimate the number of groups in the data as well as grouping similar objects together. Once the tree has been grown, it can be used to predict the labels given new observations. The prediction accuracy is compared with the ones derived from supervised methods, and exhibits good performance. A real data application on vehicle trajectories analysis illustrates the effectiveness of our approach. Section 6 presents an extension of the method to images data based on the eigendecomposition of the image observations using the FCP-TPA algorithm [1]. The proofs are relegated to the Supplementary Material.

2 Model and methodology

2.1 Notion of multivariate functional data

The structure of our data, referred to as *multivariate functional data*, is very similar to that presented in [19]. The data consist of independent trajectories of a vector-valued stochastic process $X = (X^{(1)}, \dots, X^{(P)})^\top$, $P \geq 1$. (Here and in the following, for any matrix A , A^\top denotes its transpose.) For each $1 \leq p \leq P$, let \mathcal{T}_p be a rectangle in some Euclidean space \mathbb{R}^{d_p} with $d_p \geq 1$, as for instance, $\mathcal{T}_p = [0, 1]^{d_p}$. Each coordinate $X^{(p)} : \mathcal{T}_p \rightarrow \mathbb{R}$ is assumed to belong to $\mathcal{L}^2(\mathcal{T}_p)$, the Hilbert space of squared-integrable real-valued functions defined on \mathcal{T}_p , endowed with the usual inner product that we denote by $\langle \cdot, \cdot \rangle_2$. Thus X is a stochastic process indexed by $\mathbf{t} = (t_1, \dots, t_P)$ belonging to the P -fold Cartesian product $\mathcal{T} := \mathcal{T}_1 \times \dots \times \mathcal{T}_P$ and taking values in the P -fold Cartesian product space $\mathcal{H} := \mathcal{L}^2(\mathcal{T}_1) \times \dots \times \mathcal{L}^2(\mathcal{T}_P)$.

We consider the function $\langle\langle \cdot, \cdot \rangle\rangle : \mathcal{H} \times \mathcal{H} \rightarrow \mathbb{R}$,

$$\langle\langle f, g \rangle\rangle := \sum_{p=1}^P \left\langle f^{(p)}, g^{(p)} \right\rangle_2 = \sum_{p=1}^P \int_{\mathcal{T}_p} f^{(p)}(t_p) g^{(p)}(t_p) dt_p, \quad f, g \in \mathcal{H}.$$

Then, \mathcal{H} is a Hilbert space with respect to the inner product $\langle\langle \cdot, \cdot \rangle\rangle$ [19]. We denote by $\|\cdot\|$ the norm induced by $\langle\langle \cdot, \cdot \rangle\rangle$. Let $\mu : \mathcal{T} \rightarrow \mathcal{H}$ denote the mean function of the process X , $\mu(\mathbf{t}) := \mathbb{E}(X(\mathbf{t}))$, $\mathbf{t} \in \mathcal{T}$.

2.2 A mixture model for curves

The standard model-based clustering approaches consider that data is sampled from a mixture of probability densities on a finite dimensional space. As pointed out by [27], this approach is not directly applicable to functional data since the notion of probability density generally does not exist for functional random variables. See also [9]. Consequently, in general, model-based clustering approaches assume a mixture of parametric distributions on the coefficients of the representation of the process X in a basis. This is also the way we proceed in the following.

Let $K \geq 1$ be an integer, and let Z be a random variable taking values in $\{1, \dots, K\}$ such that $\mathbb{P}(Z = k) = p_k$ with $p_k > 0$ and $\sum_{k=1}^K p_k = 1$. The variable Z is a latent variable, also called the class label, representing the cluster membership of the realizations of the process.

Although $X = \{X(\mathbf{t})\}_{\mathbf{t} \in \mathcal{T}}$ could be defined on a set of vectors and could be vector-valued, we shall call *curves* its independent realizations which generate the data. We consider that the stochastic process X follows a *functional mixture model with K components*:

$$X(\mathbf{t}) = \sum_{k=1}^K \mu_k(\mathbf{t}) \mathbf{1}_{\{Z=k\}} + \sum_{j \geq 1} \xi_j \phi_j(\mathbf{t}), \quad \mathbf{t} \in \mathcal{T}, \quad (2.1)$$

where

- $\mu_1, \dots, \mu_K \in \mathcal{H}$ are the mean curves per cluster.
- $\{\phi_j\}_{j \geq 1}$ is an orthonormal basis of \mathcal{H} , $\langle\langle \phi_j, \phi_{j'} \rangle\rangle = 0$ and $\|\phi_j\|^2 = 1$, $\forall 1 \leq j \neq j' < \infty$.

- ξ_j , $j \geq 1$ are real-valued random variables which are conditionally independent given Z . For each $1 \leq k \leq K$, the conditional distribution of ξ_j given $Z = k$ is a zero-mean Gaussian distribution with variance $\sigma_{kj}^2 \geq 0$, for all $j \geq 1$. Moreover, $\sum_{k=1}^K \sum_{j \geq 1} \sigma_{kj}^2 < \infty$ and for any $k \neq k'$, $\sum_{j \geq 1} |\sigma_{kj}^2 - \sigma_{k'j}^2| > 0$.

The condition that $\{\phi_j\}_{j \geq 1}$ is an orthonormal basis is not really necessary; it just allows us to write some technical conditions in a simpler way. Our assumptions imply that $\sum_{j \geq 1} \text{Var}(\xi_j) < \infty$ and thus $\mathbb{E}(\|X\|^2) < \infty$. The definition of the processes X (2.2) does not require us to know the basis $\{\phi_j\}_{j \geq 1}$. However, for inference purposes, one needs to consider a representation in a workable basis. The following result presents the relationship between the two representations.

Lemma 1 *Let X be defined as in (2.2) for some orthonormal basis $\{\phi_j\}_{j \geq 1}$. Let $\{\psi_j\}_{j \geq 1}$ be another orthonormal basis in \mathcal{H} and consider*

$$c_j = \langle X - \mu, \psi_j \rangle, \quad j \geq 1 \quad \text{where} \quad \mu(\cdot) = \sum_{k=1}^K p_k \mu_k(\cdot). \quad (2.2)$$

Then

$$c_j | Z = k \sim \mathcal{N}(m_{kj}, \tau_{kj}^2), \quad \text{where} \quad m_{kj} = \langle \mu_k - \mu, \psi_j \rangle \quad \text{and} \quad \tau_{kj}^2 = \sum_{l \geq 1} \langle \phi_l, \psi_j \rangle^2 \sigma_{kl}^2.$$

Moreover,

$$\text{Cov}(c_i, c_j | Z = k) = \sum_{l \geq 1} \langle \phi_l, \psi_i \rangle \langle \phi_l, \psi_j \rangle \sigma_{kl}^2, \quad i, j \geq 1.$$

Each c_j has a centered Gaussian distribution and, for any $i, j \geq 1$,

$$\text{Cov}(c_i, c_j) = \sum_{k=1}^K p_k \left(\sum_{l \geq 1} \langle \phi_l, \psi_i \rangle \langle \phi_l, \psi_j \rangle \sigma_{kl}^2 + \langle \mu_k - \mu, \psi_i \rangle \langle \mu_k - \mu, \psi_j \rangle \right).$$

Remark 1 *Despite the Gaussian assumption for the ξ_j , the model (2.2) is quite general because there is practically no assumption concerning the basis $\{\phi_j\}_{j \geq 1}$. Lemma 1 shows that, no matter what the user's choice may be for the orthonormal basis $\{\psi_j\}_{j \geq 1}$, the clusters will be preserved after expressing the realizations of the process into this basis. However, depending on the objective, some bases might be more suitable than another.*

Remark 2 *A careful look at the proof of Lemma 1 reveals that it remains true even if $\{\phi_j\}_{j \geq 1}$ is not an orthonormal basis, which could be appealing for extending our framework. For instance, consider the case where the clusters of curves correspond to representations in different bases. As an illustration, let us consider the case $K = 2$ and let $\{\phi_{1,j}\}_{j \geq 1}$ and $\{\phi_{2,j}\}_{j \geq 1}$ be orthonormal systems. The union of these sets is not necessarily an orthonormal system however. The centered realizations from each cluster corresponds to the representations $\sum_{j \geq 1} \eta_{1,j} \phi_{1,j}(\mathbf{t})$ and $\sum_{j \geq 1} \eta_{2,j} \phi_{2,j}(\mathbf{t})$, respectively. Here, $\eta_{1,j}, \eta_{2,j}$, $j \geq 1$ are independent zero-mean Gaussian variables with variances $\sigma_{1,j}^2$ and $\sigma_{2,j}^2$, respectively, such that $\sum_{j \geq 1} \{\sigma_{1,j}^2 + \sigma_{2,j}^2\} < \infty$. We can then write the process X in the form (2.2) : $X(\mathbf{t}) - \mathbb{E}[X(\mathbf{t})] = \sum_{j \geq 1} \xi_j \phi_j(\mathbf{t})$ with*

$$\xi_{2j-1} = \mathbf{1}_{\{Z=1\}} \eta_{1,j} + \mathbf{1}_{\{Z=2\}} \delta_0 \quad \text{and} \quad \xi_{2j} = \mathbf{1}_{\{Z=1\}} \delta_0 + \mathbf{1}_{\{Z=2\}} \eta_{2,j}, \quad j \geq 1,$$

and

$$\phi_{2j-1}(\mathbf{t}) = \phi_{1,j}(\mathbf{t}) \quad \text{and} \quad \phi_{2j}(\mathbf{t}) = \phi_{2,j}(\mathbf{t}), \quad j \geq 1, \mathbf{t} \in \mathcal{T},$$

where δ_0 is the Dirac mass at the origin. Thus each ξ_j is a mixture between two centered normal variables with positive and zero variance, respectively. Given any orthonormal basis $\{\psi_j\}_{j \geq 1}$, the $c_j = \langle X - \mu, \psi_j \rangle$ will still have the properties described in Lemma 1.

Remark 3 Our framework and Lemma 1 could also capture mixtures induced by some phase variations. This kind of modeling has received increasing attention recently. See, e.g., [31, 34]. Indeed, let $h_k : \mathcal{T} \rightarrow \mathcal{T}$, $1 \leq k \leq K$, be different maps defined on \mathcal{T} . For instance, when $P = 1$ and $\mathcal{T}_P = [0, 1]$, the maps could be in the form $h_k(t) = \{b_k t^{a_k} + (1 - b_k)\}^{c_k}$ for some $a_k, c_k > 0$ and $0 < b_k \leq 1$. We can then define $\{\phi_{k,j}\}_{j \geq 1}$ as $\phi_{k,j}(\mathbf{t}) = \bar{\phi}(h_k(\mathbf{t}))$, $\mathbf{t} \in \mathcal{T}$, $k = 1, \dots, K$, where $\{\bar{\phi}_j\}_{j \geq 1}$ is some reference orthonormal basis. We next consider that each cluster corresponds to a representation in a basis $\{\phi_{k,j}\}_{j \geq 1}$, $1 \leq k \leq K$. Remark 2 shows that Lemma 1 remains valid. More generally, each representation in a basis $\{\phi_{k,j}\}_{j \geq 1}$, $1 \leq k \leq K$, could be a mixture of two or more components as in (2.2). After gathering the K representations, the result would still be a functional mixture in the sense of (2.2).

Remarks 1 to 3 indicate that our modeling approach is quite flexible and can capture many interesting situations. In applications, one cannot use an infinite number of terms in the representation of X , and such a representation must be truncated. We show in Lemma A.1 in the Supplementary Material that such a truncation could be arbitrarily accurate.

2.3 Multivariate Karhunen-Loève representation

A convenient basis for representing the realizations of X is that given by the eigenfunctions of the covariance operator. In this section, following [19], we recall the formal definition of that basis in the context of our model.

Let C denote the $P \times P$ matrix-valued covariance function which, for $\mathbf{s}, \mathbf{t} \in \mathcal{T}$, is defined as

$$C(\mathbf{s}, \mathbf{t}) := \mathbb{E} \left(\{X(\mathbf{s}) - \mathbb{E}(X(\mathbf{s}))\} \{X(\mathbf{t}) - \mathbb{E}(X(\mathbf{t}))\}^\top \right), \quad \mathbf{s}, \mathbf{t} \in \mathcal{T}.$$

More precisely, for $1 \leq p, q \leq P$, the (p, q) th entry of the matrix $C(\mathbf{s}, \mathbf{t})$ is the covariance function between the p th and the q th components of the process X :

$$C_{p,q}(s_p, t_q) := \mathbb{E} \left(\{X^{(p)}(s_p) - \mathbb{E}(X^{(p)}(s_p))\} \{X^{(q)}(t_q) - \mathbb{E}(X^{(q)}(t_q))\} \right), \quad s_p \in \mathcal{T}_p, t_q \in \mathcal{T}_q.$$

Following [19], it is assumed herein that

$$\max_{1 \leq p \leq P} \sup_{s_p \in \mathcal{T}_p} \int_{\mathcal{T}_q} C_{p,q}^2(s_p, t_q) dt_q < \infty,$$

and that, for all $1 \leq p \leq P$, $C_{p,q}(s_p, \cdot)$ is uniformly continuous in the sense that

$$\forall \epsilon > 0 \exists \delta > 0 : |t_q - t'_q| < \delta \Rightarrow \max_{1 \leq p \leq P} \sup_{s_p \in \mathcal{T}_p} |C_{p,q}(s_p, t_q) - C_{p,q}(s_p, t'_q)| < \epsilon.$$

In particular, $C_{p,q}(\cdot, \cdot)$ belongs to $\mathcal{L}^2(\mathcal{T}_p \times \mathcal{T}_q)$.

Let $\Gamma : \mathcal{H} \mapsto \mathcal{H}$ denote the covariance operator of X , defined as the integral operator with kernel C . That is, for $f \in \mathcal{H}$ and $\mathbf{t} \in \mathcal{T}$, the q th component of $\Gamma f(\mathbf{t})$ is given by

$$(\Gamma f)^{(q)}(t_q) := \langle\langle C_{\cdot, q}(\cdot, t_q), f(\cdot) \rangle\rangle, \quad t_q \in \mathcal{T}_q.$$

By the results in [19], and the theory of Hilbert-Schmidt operators, *e.g.* [40, Chapter VI], there exists a complete orthonormal basis $\{\varphi_j\}_{j \geq 1} \subset \mathcal{H}$ and a sequence of real numbers $\lambda_1 \geq \lambda_2 \geq \dots \geq 0$ such that $\Gamma \varphi_j = \lambda_j \varphi_j$ and $\lambda_j \rightarrow 0$ as $j \rightarrow \infty$. The λ_j 's are the eigenvalues of the covariance operator Γ and the φ_j 's are the associated eigenfunctions. Then, X as defined in (2.2) allows for the Karhunen-Loève representation

$$X(\mathbf{t}) = \mu(\mathbf{t}) + \sum_{j \geq 1} \mathbf{c}_j \varphi_j(\mathbf{t}), \quad \mathbf{t} \in \mathcal{T}, \quad \text{with} \quad \mathbf{c}_j = \langle\langle X - \mu, \varphi_j \rangle\rangle, \quad (2.3)$$

and $\text{Cov}(\mathbf{c}_j, \mathbf{c}_l) = \lambda_j \mathbf{1}_{\{j=l\}}$. Let us call $\{\varphi_j\}_{j \geq 1}$ the multivariate functional principal component analysis (MFPCA) basis.

Let $J \geq 1$ and assume that $\lambda_1 > \lambda_2 > \dots > \lambda_J > \lambda_{J+1}$, which in particular, implies that the first J eigenvalues are nonzero. By an easy extension of [20, Theorem 3.2], we can deduce that, up to a sign, the elements of the MFPCA basis are characterized by the following property :

$$\begin{aligned} \varphi_1 &= \arg \max_{\varphi} \langle\langle \Gamma \varphi, \varphi \rangle\rangle \quad \text{such that} \quad \|\varphi\| = 1, \\ \varphi_j &= \arg \max_{\varphi} \langle\langle \Gamma \varphi, \varphi \rangle\rangle \quad \text{such that} \quad \|\varphi\| = 1 \text{ and } \langle\langle \varphi, \varphi_l \rangle\rangle = 0, \quad \forall l < j \leq J. \end{aligned} \quad (2.4)$$

The MFPCA basis is the one which will induce the most accurate truncation for a given J (see the Lemma A.2 for a proof). Therefore, among the workable bases one could use in practice, the MFPCA basis is likely to be a privileged one.

By Lemma 1, whenever X is defined as in (2.2), for any $J \geq 1$, the distribution of the vector $\mathbf{c} = (\mathbf{c}_1, \dots, \mathbf{c}_J)^\top$ defined in (2.3) is a centered (multivariate) Gaussian mixture model (GMM) distribution

$$g(\mathbf{c}) = \sum_{k=1}^K p_k f_k(\mathbf{c} | \mathbf{m}_{J,k}, \Sigma_{J,k}), \quad \mathbf{c} \in \mathbb{R}^J, \quad (2.5)$$

where, for each $1 \leq k \leq K$, $f_k(\cdot | \mathbf{m}_{J,k}, \Sigma_{J,k})$ is the probability density function of a multivariate Gaussian distribution of the k th given the values of its parameters $\mathbf{m}_{J,k}$ and $\Sigma_{J,k}$ such that

$$\mathbf{m}_{J,k} = (\langle\langle \mu_k - \mu, \varphi_1 \rangle\rangle, \dots, \langle\langle \mu_k - \mu, \varphi_J \rangle\rangle)^\top,$$

and the (i, j) -entry of the matrix $\Sigma_{J,k}$ given by

$$\text{Cov}(\mathbf{c}_i, \mathbf{c}_j | Z = k) = \sum_{l \geq 1} \langle\langle \phi_l, \varphi_i \rangle\rangle \langle\langle \phi_l, \varphi_j \rangle\rangle \sigma_{kl}^2, \quad 1 \leq i, j \leq J.$$

We point out that the GMM defined in (2.3) is consistent with respect to J in the sense that, for any $J \geq 1$, $(\mathbf{c}_1, \dots, \mathbf{c}_J)^\top$ and $(\mathbf{c}_1, \dots, \mathbf{c}_J, \mathbf{c}_{J+1})^\top$ have the same label Z , which also coincides with the label of the curve X . In particular, in general, the label Z of a curve in the sample, could be identified by \mathbf{c}_1 . Moreover, since $\text{Var}(\mathbf{c}_1) > \text{Var}(\mathbf{c}_j)$, $\forall j > 1$, it is likely that the first coefficient in the Karhunen-Loève representation is informative for identifying the mixture structure.

Let

$$X_{\lceil J \rceil}(\mathbf{t}) = \mu(\mathbf{t}) + \sum_{j=1}^J \mathbf{c}_j \varphi_j(\mathbf{t}), \quad \mathbf{t} \in \mathcal{T}, \quad J \geq 1, \quad (2.6)$$

be the truncated Karhunen-Loève expansion of the process X and

$$X_{\lceil J^{(p)} \rceil}^{(p)}(t_p) = \mu^{(p)}(t_p) + \sum_{j=1}^{J^{(p)}} \mathbf{c}_j^{(p)} \rho_j^{(p)}(t_p), \quad t \in \mathcal{T}_p, \quad J^{(p)} \geq 1, \quad 1 \leq p \leq P, \quad (2.7)$$

be the truncated Karhunen-Loève expansion of the components of the process X . [19] derive a direct relationship between the truncated representations (2.3) of the single elements $X^{(p)}$ and the truncated representation (2.3) of the multivariate functional data X . We recall this result in the following lemma.

Lemma 2 ([19], Proposition 5) *Let $X_{\lceil J \rceil}$ be the truncation of X as in (2.3) and $X_{\lceil J^{(p)} \rceil}^{(p)}$ be the truncation of $X^{(p)}$ for each $1 \leq p \leq P$ as in (2.3).*

1. *Let $\Gamma^{(p)}$ be the univariate covariance operator associated with $X^{(p)}$. The positive eigenvalues of $\Gamma^{(p)}$, $\lambda_1^{(p)} \geq \dots \geq \lambda_{J^{(p)}}^{(p)} > 0$, $J^{(p)} < J$ correspond the positive eigenvalues of the matrix $\mathbf{A}^{(p)} \in \mathbb{R}^{J \times J}$ with entries*

$$\mathbf{A}_{jj'}^{(p)} = (\lambda_j \lambda_{j'})^{1/2} \left\langle \varphi_j^{(p)}, \varphi_{j'}^{(p)} \right\rangle_2, \quad j, j' = 1, \dots, J.$$

The eigenfunctions of $\Gamma^{(p)}$ are given by

$$\rho_j^{(p)}(t_p) = \left(\lambda_j^{(p)} \right)^{-1/2} \sum_{j'=1}^J \lambda_{j'}^{1/2} [\mathbf{u}_j^{(p)}]_{j'} \varphi_{j'}^{(p)}(t_p), \quad t_p \in \mathcal{T}, \quad j = 1, \dots, J^{(p)},$$

where $\mathbf{u}_j^{(p)}$ denotes an (orthonormal) eigenvector of $\mathbf{A}^{(p)}$ associated with eigenvalue $\lambda_j^{(p)}$ and $[\mathbf{u}_j^{(p)}]_{j'}$ denotes the j' -th entry of this vector. Finally, the univariate scores are

$$\mathbf{c}_j^{(p)} = \left\langle X^{(p)}, \rho_j^{(p)} \right\rangle_2 = \left(\lambda_j^{(p)} \right)^{-1/2} \sum_{j'=1}^J \lambda_{j'}^{1/2} [\mathbf{u}_j^{(p)}]_{j'} \sum_{j''=1}^J \mathbf{c}_{j''} \left\langle \varphi_{j'}^{(p)}, \varphi_{j''}^{(p)} \right\rangle_2.$$

2. *The positive eigenvalues of Γ , $\lambda_1, \dots, \lambda_J > 0$, with $J \leq \sum_{p=1}^P J^{(p)} =: J_+$ are the positive eigenvalues of the matrix $\mathbf{Z} \in \mathbb{R}^{J_+ \times J_+}$ consisting of blocks $\mathbf{Z}^{(pp')} \in \mathbb{R}^{J^{(p)} \times J^{(p)'}}$ with entries*

$$\mathbf{Z}_{jj'}^{(pp')} = \text{Cov} \left(\mathbf{c}_j^{(p)}, \mathbf{c}_{j'}^{(p')} \right), \quad j = 1, \dots, J^{(p)}, \quad j' = 1, \dots, J^{(p')}, \quad p, p' = 1, \dots, P.$$

The eigenfunctions of Γ are given by $\varphi_j^{(p)}(t_p) = \sum_{j'=1}^{J^{(p)}} [\mathbf{v}_j]_{j'}^{(p)} \rho_j^{(p)}(t_p)$, $t_p \in \mathcal{T}_p$, $j = 1, \dots, J$, where $[\mathbf{v}_j]^{(p)} \in \mathbb{R}^{J^{(p)}}$ denotes the p -th block of an (orthonormal) eigenvector \mathbf{v}_j of \mathbf{Z} associated with eigenvalue λ_j . The scores $\mathbf{c}_j \in \mathbb{R}$ are given by

$$\mathbf{c}_j = \sum_{p=1}^P \sum_{j'=1}^{J^{(p)}} [\mathbf{v}_j]_{j'}^{(p)} \mathbf{c}_{j'}^{(p)}.$$

Lemma 2 allows us to compute the scores of the P -dimensional stochastic process X using the scores of each of the P components $X^{(p)}$ as univariate building blocks. However, when P grows, it might not be suitable to perform P univariate fPCA from a computational point of view. Recently, [21] extend this result to large-dimensional processes by imposing a sparsity assumption. This possible extension will be investigated in future work.

3 Learning the parameters

In real data applications, the realizations of X are usually measured with error at discrete, and possibly random, points in the definition domain. Therefore, let us consider N curves $X_1, \dots, X_n, \dots, X_N$ generated as a random sample of the P -dimensional stochastic process X with continuous trajectories. For each $1 \leq n \leq N$, and given a vector of positive integers $\mathbf{M}_n = (M_n^{(1)}, \dots, M_n^{(P)}) \in \mathbb{R}^P$, let $T_{n,\mathbf{m}} = (T_{n,m_1}^{(1)}, \dots, T_{n,m_P}^{(P)})$, $1 \leq m_p \leq M_n^{(p)}$, $1 \leq p \leq P$, be the random observation times for the curve X_n . These times are obtained as independent copies of a variable \mathbf{T} taking values in \mathcal{T} . The vectors $\mathbf{M}_1, \dots, \mathbf{M}_N$ represent an independent sample of an integer-valued random vector \mathbf{M} with expectation $\boldsymbol{\mu}_M$ which increases with N . We assume that the realizations of X , \mathbf{M} and \mathbf{T} are mutually independent. The observations associated with a curve, or trajectory, X_n consist of the pairs $(Y_{n,\mathbf{m}}, T_{n,\mathbf{m}}) \in \mathbb{R}^P \times \mathcal{T}$, where $\mathbf{m} = (m_1, \dots, m_P)$, $1 \leq m_p \leq M_n^{(p)}$, $1 \leq p \leq P$, and $Y_{n,\mathbf{m}}$ is defined as

$$Y_{n,\mathbf{m}} = X_n(T_{n,\mathbf{m}}) + \varepsilon_{n,\mathbf{m}}, \quad 1 \leq n \leq N, \quad (3.1)$$

with the $\varepsilon_{n,\mathbf{m}}$ being independent copies of a centered error random vector $\varepsilon \in \mathbb{R}^P$ with finite variance. We use the notation $X_n(\mathbf{t})$ for the value at \mathbf{t} of the realization X_n of X . The N -sample of X is composed of two sub-populations: a *learning set* of N_0 curves to estimate the mixture components of the process X and a set of N_1 curves to be classified using the previous grouping as a classifier that we call the *online set*. Thus, $1 \leq N_0, N_1 < N$ and $N_0 + N_1 = N$. Let X_1, \dots, X_{N_0} denote the curves corresponding to the *learning set*. The learning sample will be used to estimate the mean and covariance functions, as well as the eigencomponents, of the process X . Our first objective is to construct a partition \mathcal{U} of the space \mathcal{H} using the learning sample. Then, the second aim is to use the partition \mathcal{U} as a classifier for a possibly very large set of N_1 new curves. Let $X_{[1]} = X_{N_0+1}, \dots, X_{[N_1]} = X_N$, denote the curves from the *online set* to be classified using the partition \mathcal{U} .

3.1 Estimation of mean and covariance

We develop estimators for the mean and the covariance functions of a component $X^{(p)}$, $1 \leq p \leq P$ from the process X . These estimators are used to compute estimators of eigenvalues and eigenfunctions of $X^{(p)}$ for the expansion (2.3). It is worthwhile to notice that, because of Lemma 2, we do not need to estimate the covariance between $X^{(p)}$ and $X^{(q)}$ for $p \neq q$.

Let $\widehat{X}_n^{(p)}$ be a suitable nonparametric estimator of the curve $X_n^{(p)}$ applied with the $M_n^{(p)}$ pairs $(Y_{n,m_p}^{(p)}, T_{n,m_p}^{(p)})$, $n = 1, \dots, N_0$, as for instance a local polynomial estimator such as the one defined in [17]. With at hand, the \widehat{X}_n 's tuned for the mean estimation, let

$$\widehat{\mu}_{N_0}^{(p)}(t_p) = \frac{1}{N_0} \sum_{n=1}^{N_0} \widehat{X}_n^{(p)}(t_p), \quad t_p \in \mathcal{T}_p.$$

For the covariance function, following [52], we distinguish the diagonal from the non-diagonal points. With at hand, the $\widehat{X}_n^{(p)}$'s tuned for the covariance function estimation,

$$\widehat{C}_{p,p}(s_p, t_p) = \frac{1}{N_0} \sum_{n=1}^{N_0} \widehat{X}_n^{(p)}(s_p) \widehat{X}_n^{(p)}(t_p) - \widehat{\mu}_{N_0}^{(p)}(s_p) \widehat{\mu}_{N_0}^{(p)}(t_p), \quad s_p, t_p \in \mathcal{T}_p, \quad s_p \neq t_p. \quad (3.2)$$

The diagonal of the covariance is then estimated using two-dimensional kernel smoothing with $\widehat{C}_{p,p}(s_p, t_p)$, $s_p \neq t_p$ as input data. See [52] for the details.

3.2 Derivation of the MFPCA components

Following [19], using Lemma 2, we estimate the multivariate components for X by plugging the univariate components computed from each $X^{(p)}$. These estimations are done as follows. First, we perform an univariate fPCA on each of the components of X separately. For a component $X^{(p)}$, the eigenfunctions and eigenvectors are computed as a matrix analysis of the estimated covariance $\widehat{C}_{p,p}$, from (3.1). This results in a set of eigenfunctions $(\widehat{\rho}_1^{(p)}, \dots, \widehat{\rho}_{J^{(p)}}^{(p)})$ associated with a set of eigenvalues $(\widehat{\lambda}_1^{(p)}, \dots, \widehat{\lambda}_{J^{(p)}}^{(p)})$ for a given truncation integer $J^{(p)}$. Then, the univariate scores for a realization $X_n^{(p)}$ of $X^{(p)}$ are given by $\widehat{\mathbf{c}}_{j,n}^{(p)} = \langle \widehat{X}_n^{(p)}, \widehat{\rho}_j^{(p)} \rangle_2$, $1 \leq j \leq J^{(p)}$. These scores might be estimated by numerical integration. However, in some cases, *e.g.* for sparse data, it may be more suitable to use the PACE method (see [52]). We then define the matrix $\mathcal{Z} \in \mathbb{R}^{N_0 \times J_+}$, where on each row we concatenate the scores obtained for the P components of the n th observation : $(\widehat{\mathbf{c}}_{1,n}^{(1)}, \dots, \widehat{\mathbf{c}}_{J^{(1)},n}^{(1)}, \dots, \widehat{\mathbf{c}}_{1,n}^{(P)}, \dots, \widehat{\mathbf{c}}_{J^{(P)},n}^{(P)})$. An estimate $\widehat{\mathbf{Z}} \in \mathbb{R}^{J_+ \times J_+}$ of the matrix \mathbf{Z} , from Lemma 2, is given by $\widehat{\mathbf{Z}} = (N_0 - 1)^{-1} \mathcal{Z}^\top \mathcal{Z}$. An eigenanalysis of the matrix $\widehat{\mathbf{Z}}$ is done to estimate the eigenvectors $\widehat{\mathbf{v}}_j$ and eigenvalues $\widehat{\lambda}_j$. The multivariate eigenfunctions are estimated with $\widehat{\varphi}_j^{(p)}(t_p) = \sum_{j'=1}^{J^{(p)}} [\widehat{\mathbf{v}}_j]_{j'}^{(p)} \widehat{\rho}_{j'}^{(p)}(t_p)$, $t_p \in \mathcal{T}_p$, $1 \leq j \leq J_+$, $1 \leq p \leq P$. and the multivariate scores with $\widehat{\mathbf{c}}_{j,n} = \mathcal{Z}_n \cdot \widehat{\mathbf{v}}_j$, $1 \leq n \leq N_0$, $1 \leq j \leq J_+$. The multivariate Karhunen-Loève expansion of the process X is thus

$$\widehat{X}_n(\mathbf{t}) = \widehat{\mu}_{N_0}(\mathbf{t}) + \sum_{j=1}^J \widehat{\mathbf{c}}_{j,n} \widehat{\varphi}_j(\mathbf{t}), \quad \mathbf{t} \in \mathcal{T}.$$

where $\widehat{\mu}_{N_0}(\cdot) = (\widehat{\mu}_{N_0}^{(1)}(\cdot), \dots, \widehat{\mu}_{N_0}^{(P)}(\cdot))$ is the vector of the estimated mean functions.

4 Multivariate functional clustering

Let \mathcal{S} be a sample of realizations of the process X , defined in (2.2). We consider the problem of learning a partition \mathcal{U} such that every element U of \mathcal{U} gathers similar elements of \mathcal{S} . Our clustering procedure follows the idea of clustering using unsupervised regression trees (CUBT), considered by [16], which we adapt to functional data. In the following, we describe in detail the Functional Clustering Using Unsupervised Binary Trees (fCUBT) algorithm.

4.1 Building the maximal tree

Let $\mathcal{S}_{N_0} = \{X_1, \dots, X_{N_0}\}$ be a training sample composed of N_0 independent realizations of the stochastic process $X \in \mathcal{H}$ defined in (2.2). In the following, a tree \mathfrak{T} is a full binary tree, meaning every node has zero or two children, which represents a nested partition of the sample \mathcal{S}_{N_0} . We will denote the depth of a tree \mathfrak{T} by $\mathfrak{D} \geq 1$.

A tree \mathfrak{T} starts with the root node $\mathfrak{S}_{0,0}$ to which we assign the whole space sample \mathcal{S}_{N_0} . Next, every node $\mathfrak{S}_{\mathfrak{d},j} \subset \mathcal{S}_{N_0}$ is indexed by the pair (\mathfrak{d}, j) where \mathfrak{d} is the depth index of the node, with $0 \leq \mathfrak{d} < \mathfrak{D}$, and j is the node index, with $0 \leq j < 2^{\mathfrak{d}}$. A non-terminal node

(\mathfrak{d}, j) has two children, corresponding to disjoint subsets $\mathfrak{S}_{\mathfrak{d}+1, 2j}$ and $\mathfrak{S}_{\mathfrak{d}+1, 2j+1}$ of \mathcal{S}_{N_0} such that $\mathfrak{S}_{\mathfrak{d}, j} = \mathfrak{S}_{\mathfrak{d}+1, 2j} \cup \mathfrak{S}_{\mathfrak{d}+1, 2j+1}$. A terminal node (\mathfrak{d}, j) has no children.

A tree \mathfrak{T} is thus defined using a top-down procedure by recursively splitting. At each stage, a node (\mathfrak{d}, j) is possibly split into two subnodes, namely the left and right child, with indices $(\mathfrak{d} + 1, 2j)$ and $(\mathfrak{d} + 1, 2j + 1)$, respectively, provided it fulfills some condition. A multivariate functional principal components analysis as presented in Section 3.2, with J components, is then conducted on the elements of $\mathfrak{S}_{\mathfrak{d}, j}$. This results in a set of eigenvalues $\Lambda_{\mathfrak{d}, j} = (\lambda_{\mathfrak{d}, j}^1, \dots, \lambda_{\mathfrak{d}, j}^J)$ associated with a set of eigenfunctions $\Phi_{\mathfrak{d}, j} = (\varphi_{\mathfrak{d}, j}^1, \dots, \varphi_{\mathfrak{d}, j}^J)$. The matrix of scores $C_{\mathfrak{d}, j}$ is then defined with the columns built with the projections of the elements of $\mathcal{S}_{\mathfrak{d}, j}$ onto the elements of $\Phi_{\mathfrak{d}, j}$. More precisely, to each $X_n \in \mathfrak{S}_{\mathfrak{d}, j}$, there is a corresponding column of size J defined as

$$\mathbf{c}_{n, \mathfrak{d}, j} = (\langle X_n - \mu_{\mathfrak{d}, j}, \varphi_{\mathfrak{d}, j}^1 \rangle, \dots, \langle X_n - \mu_{\mathfrak{d}, j}, \varphi_{\mathfrak{d}, j}^J \rangle)^\top, \quad (4.1)$$

where $\mu_{\mathfrak{d}, j}$ is the mean curve within the node $\mathfrak{S}_{\mathfrak{d}, j}$.

We can retrieve the groups (clusters) of curves considering a mixture model as in Section 2.2 for the columns of the matrix of scores. At each node $\mathfrak{S}_{\mathfrak{d}, j}$, for each $K = 1, \dots, K_{max}$, we fit a GMM as in (2.3) to the columns of the matrix $C_{\mathfrak{d}, j}$. The resulting models are denoted as $\{\mathcal{M}_1, \dots, \mathcal{M}_{K_{max}}\}$. To fit \mathcal{M}_1 , we use the standard mean and variance matrix estimation of Gaussian distributions. To fit each of the models $\{\mathcal{M}_2, \dots, \mathcal{M}_{K_{max}}\}$, we use an EM algorithm [13]. In particular, the EM algorithm assigns a label to each curve in the node. We next consider the BIC, defined in [43], and determine

$$\widehat{K}_{\mathfrak{d}, j} = \arg \max_{K=1, \dots, K_{max}} \text{BIC}(\mathcal{M}_K) = \arg \max_{K=1, \dots, K_{max}} \{2 \log(\mathcal{L}_K) - \kappa \log |\mathfrak{S}_{\mathfrak{d}, j}|\}, \quad (4.2)$$

where \mathcal{L}_K is the likelihood function of the K -components multivariate Gaussian mixture model \mathcal{M}_K for the data at the node $\mathfrak{S}_{\mathfrak{d}, j}$, that is

$$\mathcal{L}_K = \prod_{n=1}^{|\mathfrak{S}_{\mathfrak{d}, j}|} \sum_{k=1}^K p_k f_k(\mathbf{c}_{n, \mathfrak{d}, j} \mid \mathbf{m}_{J, k}, \Sigma_{J, k}),$$

$\kappa = K + KJ + KJ(J + 1)/2 - 1$ is the dimension of the model \mathcal{M}_K and $|\mathfrak{S}_{\mathfrak{d}, j}|$ is the cardinality of the set $\mathfrak{S}_{\mathfrak{d}, j}$. If $\widehat{K}_{\mathfrak{d}, j} > 1$, we split $\mathfrak{S}_{\mathfrak{d}, j}$ using the model \mathcal{M}_2 , that is a mixture of two Gaussian vectors. Otherwise, the node is considered to be a terminal node and the construction of the tree is stopped for this node.

The recursive procedure continues downward until one of the following stopping rules are satisfied: there are less than `minsize` observations in the node or the estimation $\widehat{K}_{\mathfrak{d}, j}$ of the number of clusters in the mode $\mathfrak{S}_{\mathfrak{d}, j}$ is equal to 1. The value of the positive integer `minsize` is set by the user. When the algorithm ends, a label is assigned to each leaf (terminal node). The resulting tree is referred to as the maximal binary tree. A quasi-formal description of Algorithm 1, closer to the high-level computer language, is provided in the Section B in the Supplementary Material.

Our algorithm provides a partition of the sample. Each observation belongs to a leaf which is associated with a unique label. In a perfect case, this tree will have the same number of leaves as the number of mixture components of X . In practice, it is rarely the case, and the number of leaves may be much larger than the number of clusters. That is why an agglomerative step, that we call the joining step, should also be considered. Note that we do not have to pre-specify

the number of clusters before performing the joining step. Moreover, if the number of clusters in the maximal tree is that wanted by the user, it is not necessary to run the joining step.

The original procedure, developed in [16], included a pruning step. For each sibling node, this step computes a measure of dissimilarities between them and collapses them if this measure is lower than a predefined threshold. They need this step because their splitting criteria is based on the deviance reduction between a node and its children. The algorithm is stopped when this deviance reduction is less than a predefined threshold. The pruning step is possibly used therefore to revise an undesirable split. In our case, the splitting criteria is not based on how much deviance we gain at each node of the tree but on an estimation of the number of modes of a Gaussian mixture model. Thus, the pruning step is not useful in our case.

This method has three hyperparameters that should be set by the user. The first one is J , the number of components kept when an MFPCA is run. In our implementation, $J = \sum_{p=1}^P J^{(p)}$, and the user chooses the values $J^{(p)}$. According to the notation in Lemma 2, for each $1 \leq p \leq P$, $J^{(p)}$ represents the numbers of scores for each coordinate (or variable) $X^{(p)}$. The default value for each $J^{(p)}$ is set to explain 95% of the variance of the variable $X^{(p)}$ within the data. Let us point out that the $J^{(p)}$ need not be large in order to detect a mixture structure. In fact, as explained in Section B.1 in the Supplementary material, the first score could already detect the mixture in many situations. Therefore, when the rule of percentage of explained variance applied for each p , leads to large J , we recommend considering only one or two scores for each, or at least some coordinates $X^{(p)}$. One can thus keep J reasonably low. Higher dimension J for the vector of scores for which we consider a Gaussian mixture model could result in less performance for detecting a mixture structure in the nodes, especially for those with few data. The second hyperparameter is the minimum number of elements within a node to be considered to be split (**minsize**). In practice, it should be larger than 2, the default value is set to 10. The last one, K_{max} , refers to the number of Gaussian mixture models to try in order to decide if there is at least two clusters in the data. Its default value is set to 5 but, in practice, to reduce computation, it could be set to 2. Thus, we will just compare the model with two modes against that with one group. In some rare cases, we noticed that $K_{max} = 2$ did not allow a mixture structure to be detected while this was possible with slightly larger K_{max} . This was the reason for proposing a default K_{max} larger than 2, even if, as pointed out by a reviewer, $K_{max} = 2$ would be the natural default value with a binary splitting, as we propose.

4.2 Joining step

In this step, the idea is to join terminal nodes which do not necessarily share the same direct ascendant. Let $\mathcal{G} = (V, E)$ be a graph where $V = \{\mathfrak{S}_{\mathfrak{d},j}, 0 \leq j < 2^{\mathfrak{d}}, 0 \leq \mathfrak{d} < \mathfrak{D} \mid \mathfrak{S}_{\mathfrak{d},j}$ is a terminal node $\}$ is a set of vertices and $E \subseteq \mathbf{E}$ is a set of edges with \mathbf{E} the complete set of unordered pairs of vertices $\{(\mathfrak{S}_{\mathfrak{d},j}, \mathfrak{S}_{\mathfrak{d}',j'}) \mid \mathfrak{S}_{\mathfrak{d},j}, \mathfrak{S}_{\mathfrak{d}',j'} \in V \text{ and } \mathfrak{S}_{\mathfrak{d},j} \neq \mathfrak{S}_{\mathfrak{d}',j'}\}$. Let us clarify the definition of the set of edges E . Consider $(\mathfrak{S}_{\mathfrak{d},j}, \mathfrak{S}_{\mathfrak{d}',j'}) \in \mathbf{E}$. We are interested in the union of the nodes in the pair, $\mathfrak{S}_{\mathfrak{d},j} \cup \mathfrak{S}_{\mathfrak{d}',j'}$. After performing an MFPCA on the set $\mathfrak{S}_{\mathfrak{d},j} \cup \mathfrak{S}_{\mathfrak{d}',j'}$, we compute the matrix $C_{(\mathfrak{d},j) \cup (\mathfrak{d}',j')}$ using (4.1). For $K = 1, \dots, K_{max}$, we fit a K -component Gaussian mixture model using an EM algorithm to $C_{(\mathfrak{d},j) \cup (\mathfrak{d}',j')}$. Then, if the estimated number of clusters $\widehat{K}_{(\mathfrak{d},j) \cup (\mathfrak{d}',j')}$ using (4.1), is equal to 1, we add the pair to E . Finally, we have

$$E = \left\{ (\mathfrak{S}_{\mathfrak{d},j}, \mathfrak{S}_{\mathfrak{d}',j'}) \mid \mathfrak{S}_{\mathfrak{d},j}, \mathfrak{S}_{\mathfrak{d}',j'} \in V, \mathfrak{S}_{\mathfrak{d},j} \neq \mathfrak{S}_{\mathfrak{d}',j'} \text{ and } \widehat{K}_{(\mathfrak{d},j) \cup (\mathfrak{d}',j')} = 1 \right\}. \quad (4.3)$$

We associate with each element $(\mathfrak{S}_{\mathfrak{d},j}, \mathfrak{S}_{\mathfrak{d}',j'})$ of E , the value of the BIC that corresponds to $\widehat{K}_{(\mathfrak{d},j)\cup(\mathfrak{d}',j')}$. The edge of \mathcal{G} that corresponds to the maximum of the BIC is then removed and the associated vertices are joined. Thus, there is one cluster less. This procedure is run recursively until no pair of nodes can be joined according to the BIC or only one node in the tree remains. Finally, unique labels are associated with each remaining element of V and the partition of $\mathfrak{S}_{0,0}$, denoted as \mathcal{U} , is returned. A quasi-formal description of Algorithm 2, closer to a high-level computer language, is provided in the Section B in the Supplementary Material.

Compared to other clustering algorithms, our approach inherits the interpretability of the trees algorithm. The joining step we propose likely reduces the number of spurious clusters, which is supposed to improve the interpretability. Moreover, when two clusters are joined, one can still investigate them and interpret the fact that the algorithm merges the two clusters.

4.3 Classification of new observations

With at hand, a partition \mathcal{U} of \mathcal{S}_{N_0} obtained from a learning set of N_0 realizations of the process X , we aim to classify N_1 new trajectories of X from what we call, the online dataset. Denote \mathcal{S}_{N_1} this set of new trajectories.

Let \mathfrak{X} be an element of the set \mathcal{S}_{N_1} . The descent of the tree \mathfrak{T} is performed as follows. Let $\mathfrak{S}_{\mathfrak{d},j}, 0 \leq j < 2^{\mathfrak{d}}, 0 \leq \mathfrak{d} < \mathfrak{D}$ be the node at hand such that $\mathfrak{S}_{\mathfrak{d},j}$ is not a terminal node. We compute the projection \mathfrak{X} onto the eigenfunctions $\Phi_{\mathfrak{d},j}$. This results in the vector

$$\left(\langle \mathfrak{X} - \mu_{\mathfrak{d},j}, \varphi_{\mathfrak{d},j}^1 \rangle, \dots, \langle \mathfrak{X} - \mu_{\mathfrak{d},j}, \varphi_{\mathfrak{d},j}^J \rangle \right)^\top.$$

Using the 2-component GMM fitted on this node, we then compute the posterior probability of \mathfrak{X} belonging to each of the components. At this point, we have the probability for \mathfrak{X} belonging to each node given that it belongs to the parent node. We write $\text{Pa}(\mathfrak{S})$, the set of parent nodes of the node \mathfrak{S} which includes the node itself. We denote $\mathbb{P}^*(\cdot) := \mathbb{P}(\cdot \mid X_1, \dots, X_{N_0})$. Then, the probability for \mathfrak{X} to be in the node $\mathfrak{S}_{\mathfrak{d},j}$ is then given by

$$\mathbb{P}^*(\mathfrak{X} \in \mathfrak{S}_{\mathfrak{d},j}) = \prod_{\mathfrak{S} \in \text{Pa}(\mathfrak{S}_{\mathfrak{d},j})} \mathbb{P}^*(\mathfrak{X} \in \mathfrak{S} \mid \mathfrak{X} \in \text{Pa}(\mathfrak{S})).$$

We have $\mathbb{P}^*(\mathfrak{X} \in \mathfrak{S}_{0,0}) = 1$. In order to compute the probabilities of belonging to each element of the partition \mathcal{U} , let $\mathfrak{S}_{\mathfrak{d},j}$ and $\mathfrak{S}_{\mathfrak{d}',j'}$ be two terminal nodes that have been joined in the joining step into the node $\mathfrak{S}_{\mathfrak{d},j} \cup \mathfrak{S}_{\mathfrak{d}',j'}$. Because the sets $\mathfrak{S}_{\mathfrak{d},j}$ and $\mathfrak{S}_{\mathfrak{d}',j'}$ are disjoint,

$$\mathbb{P}^*(\mathfrak{X} \in \mathfrak{S}_{\mathfrak{d},j} \cup \mathfrak{S}_{\mathfrak{d}',j'}) = \mathbb{P}^*(\mathfrak{X} \in \mathfrak{S}_{\mathfrak{d},j}) + \mathbb{P}^*(\mathfrak{X} \in \mathfrak{S}_{\mathfrak{d}',j'}).$$

Finally, each cluster is associated with a probability of the observation \mathfrak{X} such that $\sum_{U \in \mathcal{U}} \mathbb{P}^*(\mathfrak{X} \in U) = 1$. We assign \mathfrak{X} to the cluster with the largest probability.

This procedure is run for each observation within \mathcal{S}_{N_1} and results in a partition \mathcal{V} of \mathcal{S}_{N_1} . A quasi-formal description of Algorithm 3, closer to a high-level computer language, is provided in the Section B in the Supplementary Material.

5 Empirical analysis

Using simulated data, in this section, we illustrate the behavior of our clustering algorithm and compare it with some competitors. A real data application on a vehicle trajectory dataset is

also carried out.

Our `fCUBT` procedure is compared to diverse competitors in the literature that are both designed for univariate and multivariate functional data: `FunHDDC` ([4, 42] and [41] for the **R** implementation) and `Funclust` ([25, 27] and [44] for the **R** implementation). We use the model $[a_{kj}b_kQ_kD_k]$ for `FunHDDC`. We also considered the model $[ab_kQ_kD_k]$, which yields similar results. Moreover, our approach competes with the methodology described in [23], which corresponds to the k -means algorithm with a suitable distance for functional data. In particular, the methodology in [23] uses the following distances:

$$d_1(X, Y) = \left(\sum_{p=1}^P \int_{\mathcal{T}_p} \left(X^{(p)}(t_p) - Y^{(p)}(t_p) \right)^2 dt_p \right)^{1/2} \quad \text{and}$$

$$d_2(X, Y) = \left(\sum_{p=1}^P \int_{\mathcal{T}_p} \left(\frac{dX^{(p)}(t_p)}{dt_p} - \frac{dY^{(p)}(t_p)}{dt_p} \right)^2 dt_p \right)^{1/2},$$

where $dX^{(p)}(t_p)/dt_p$ is the first derivative of $X^{(p)}(t_p)$. These two methods are denoted as `k-means-d1` and `k-means-d2` in the following. We use the implementation developed in a 2019 Madrid university report by A. Hernando Bernabé for both univariate and multivariate functional data. We also compare our algorithm with a Gaussian Mixture Model, fitted using an EM algorithm, on the coefficients of a functional principal components analysis on the dataset with a fixed number of components, quoted as `FPCA+GMM` in the following. Finally, we consider only the first step of the `fCUBT` algorithm, which is the growth of the tree, to point out the usefulness of the joining step. The method will be referred as `Growing` in the following.

We are greatly interested in the ability of the algorithms to retrieve the true number of clusters K . When the true labels are available, the estimated partitions are compared with the true partition using the Adjusted Rand Index (ARI) ([22], see Section C in the Supplementary Material for a definition).

5.1 Simulation experiments

We consider three simulations scenarios with varying degrees of difficulty. Each experiment is repeated 500 times.

Scenario 1. In our first scenario, we consider that the random curves are observed without noise. The number of clusters is fixed at $K = 5$, $P = 1$, $\mathcal{T}_1 = [0, 1]$. An independent sample of $N = 1000$ univariate curves is simulated according to the following model: for $t \in [0, 1]$:

$$\begin{aligned} \text{Cluster 1: } X(t) &= \mu_1(t) + c_{11}\phi_1(t) + c_{12}\phi_2(t) + c_{13}\phi_3(t), \\ \text{Cluster 2: } X(t) &= \mu_1(t) + c_{21}\phi_1(t) + c_{22}\phi_2(t) + c_{23}\phi_3(t), \\ \text{Cluster 3: } X(t) &= \mu_2(t) + c_{11}\phi_1(t) + c_{12}\phi_2(t) + c_{13}\phi_3(t), \\ \text{Cluster 4: } X(t) &= \mu_2(t) + c_{21}\phi_1(t) + c_{22}\phi_2(t) + c_{23}\phi_3(t), \\ \text{Cluster 5: } X(t) &= \mu_2(t) + c_{21}\phi_1(t) + c_{22}\phi_2(t) + c_{23}\phi_3(t) - 15t, \end{aligned}$$

where ϕ_k 's are the eigenfunctions of the Wiener process which are defined by $\phi_k(t) = \sqrt{2} \sin\{(k - 0.5)\pi t\}$, $k = 1, 2, 3$, and the mean functions by $\mu_1(t) = 20/\{1 + \exp(-t)\}$ and $\mu_2(t) = -25/\{1 +$

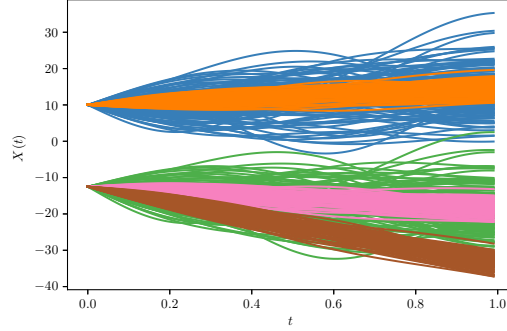


Figure 1: Simulated data for Scenario 1.

$\exp(-t)\}$. The c_{ij} 's are random normal variables defined by

$$\begin{aligned} c_{11} &\sim \mathcal{N}(0, 16), & c_{12} &\sim \mathcal{N}(0, 64/9), & c_{13} &\sim \mathcal{N}(0, 16/9), & (5.1) \\ c_{21} &\sim \mathcal{N}(0, 1), & c_{22} &\sim \mathcal{N}(0, 4/9), & c_{23} &\sim \mathcal{N}(0, 1/9). \end{aligned}$$

The mixing proportions are equal, and the curves are observed on 101 equidistant points. As shown in Figure 1, the five clusters cannot be retrieved using only the mean curve per cluster: cluster 1 (blue) and 2 (orange) share the same mean function μ_1 and similarly cluster 3 (green) and 4 (pink) share the same mean function μ_2 . As a consequence, clustering algorithms based on distances to the mean of the clusters, such as k -means, are not expected to perform well in this case. For **FunHDDC** and **Funclust**, the functional form of the data is reconstructed using a cubic B-spline basis, smoothing with 25 basis functions.

Scenario 2. The second simulation is a modification of the data simulation process of [42, scenario C]. Here, we consider that the measurements of the random curves are noisy. Thus, for this scenario, the number of clusters is fixed at $K = 5$, $P = 2$, $\mathcal{T}_1 = \mathcal{T}_2 = [0, 1]$. An independent sample of $N = 1000$ bivariate curves is simulated according to the following model : for $t_1, t_2 \in [0, 1]$,

$$\begin{aligned} \text{Cluster 1: } & X^{(1)}(t_1) = h_1(t_1) + b_{0.9}(t_1), & X^{(2)}(t_2) &= h_3(t_2) + 1.5 \times b_{0.8}(t_2), \\ \text{Cluster 2: } & X^{(1)}(t_1) = h_2(t_1) + b_{0.9}(t_1), & X^{(2)}(t_2) &= h_3(t_2) + 0.8 \times b_{0.8}(t_2), \\ \text{Cluster 3: } & X^{(1)}(t_1) = h_1(t_1) + b_{0.9}(t_1), & X^{(2)}(t_2) &= h_3(t_2) + 0.2 \times b_{0.8}(t_2), \\ \text{Cluster 4: } & X^{(1)}(t_1) = h_2(t_1) + 0.1 \times b_{0.9}(t_1), & X^{(2)}(t_2) &= h_2(t_2) + 0.2 \times b_{0.8}(t_2), \\ \text{Cluster 5: } & X^{(1)}(t_1) = h_3(t_1) + b_{0.9}(t_1), & X^{(2)}(t_2) &= h_1(t_2) + 0.2 \times b_{0.8}(t_2). \end{aligned}$$

The functions h are defined, by $h_1(t) = (6 - |20t - 6|)_+ / 4$, $h_2(t) = (6 - |20t - 14|)_+ / 4$ and $h_3(t) = (6 - |20t - 10|)_+ / 4$, for $t \in [0, 1]$. (Here, $(\cdot)_+$ denotes the positive part of the expression between the brackets.) The functions b_H are defined, for $t \in [0, 1]$, by $b_H(t) = (1+t)^{-H} B_H(1+t)$ where $B_H(\cdot)$ is a fractional Brownian motion with Hurst parameter H . The mixing proportions are set to be equal.

The data to which we apply the clustering are obtained as in (3). Each component curve is observed at 101 equidistant points over $[0, 1]$. The bivariate error vectors have zero-mean Gaussian independent components with variance $1/2$. Figure 2 presents the smoothed curves

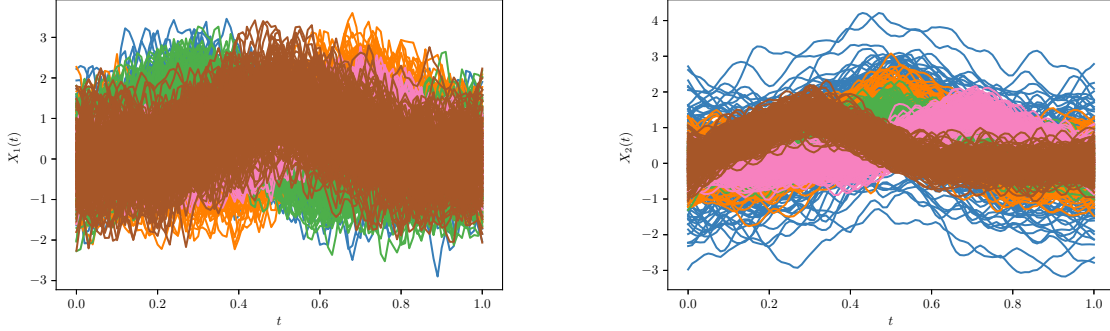


Figure 2: Simulated data for Scenario 2.

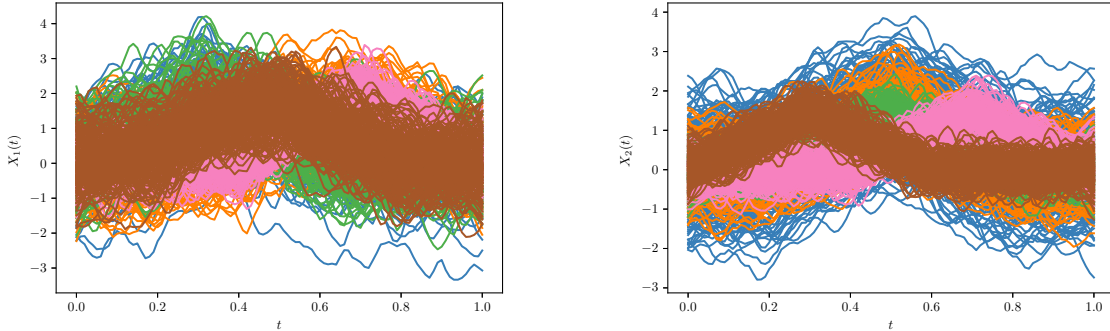


Figure 3: Simulated data for Scenario 3.

from the simulated data. Smoothing was done using the methodology of [17]. As pointed out by [42], the different clusters cannot be identified using only one variable: cluster 1 (blue) is similar to cluster 3 (green) for variable $X^{(1)}(t)$ and in like manner, cluster 1 (blue) is like cluster 2 (orange) and cluster 3 (green) for variable $X^{(2)}(t)$. The brown and pink groups might be considered as “noise” clusters that aim to make discrimination between the other groups harder. Hence, clustering methods that are specialized for univariate data, should fail to retrieve true membership using only $X^{(1)}(t)$ or $X^{(2)}(t)$. For **FunHDDC** and **Funclust**, the functional form of the data is reconstructed using a cubic B-spline basis, smoothing with 25 basis functions.

Scenario 3. The last simulation is the same as the second one, except we add some correlation between the components. So, for each $n \in \{1, \dots, N\}$, we observe a realization of the vector $X = (X^{(1)} + \alpha X^{(2)}, X^{(2)})^\top$, where $\alpha = 0.4$. Figure 3 presents the smoothed version, using the methodology of [17], of the simulated data. For **FunHDDC** and **Funclust**, the functional form of the data is reconstructed using a cubic B-spline basis, smoothing with 25 basis functions.

5.1.1 Model selection

We investigate the selection of the number of clusters for each of methods on each of the simulations. The way to return the number of clusters in a dataset depends on the algorithm. Thus,

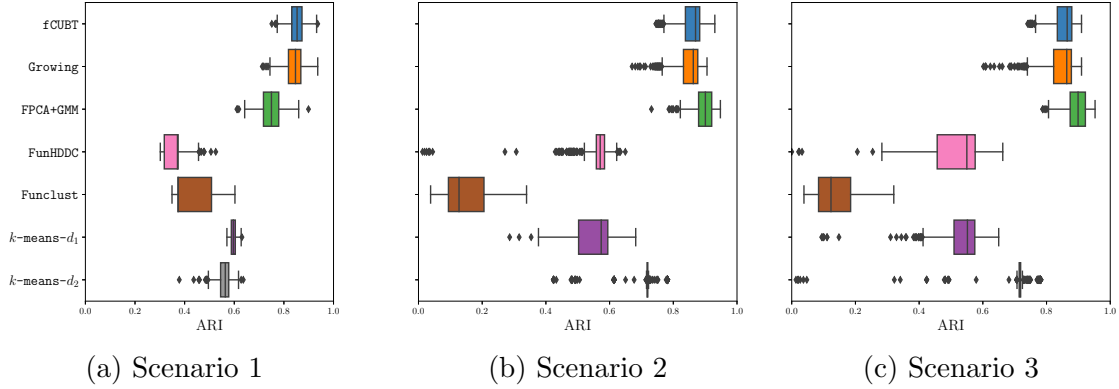


Figure 4: Estimation of ARI for all tested models on 500 simulations.

the selection for the model `fCUBT` and `Growing` is based on the BIC. Similarly, the BIC is also used for the `FunHDDC` algorithm. For all the other methods, we test all the models between $K = 1$ and $K = 8$, and return the model that maximizes the ARI criteria as the selected model. The simulation settings have been repeated 500 times and the model $[a_{kj}b_kQ_kD_k]$ is used for the `FunHDDC` algorithm.

Table 1 summarizes the results of the 500 simulations for each scenario. We remark that the `fCUBT` algorithm performs well in retrieving the right number of clusters in all the scenarios, being the first or second method in terms of retrieving percentage. Quite surprisingly, the `k-means-d2` algorithm performs very well for the second and third scenarios. It indicates that the distances between the derivatives of the curves are much more informative than the distance between the original ones. The accuracy of the selection of the number of clusters in competitors, designed for functional data, `FunHDDC` and `Funclust`, is very poor. This result has also been pointed out in [53] where the simulated data are much simpler. Finally, the results on `Growing` show the usefulness of the joining step. Thus, at the end of the growing step, we may have a large number clusters (even greater than 10) but with very few data in some of them, and so the joining step allows us to get rid of them, and thus have more relevant clusters.

5.1.2 Benchmark with existing methods

Our algorithm is compared to competitors in the literature with respect to the ARI criterion on the three scenario settings. All the competitors are applied for $K = 1$ to $K = 8$ groups for each of the scenarios and we return the best ARI found regardless of the number of clusters.

Figure 4 presents clustering results for all the tested models for the ARI criterion. We see that our algorithm performs well in all the scenarios. On the contrary, the `FunHDDC` and `Funclust` competitors do not perform well on this simulated data. Both `k-means-d1` and `k-means-d2` demonstrate acceptable ARI although not as good as `fCUBT`. The `FPCA+GMM` algorithm has similar results as `fCUBT` in terms of ARI because ARI is not penalized when the number of clusters is not the true one. So, as long as the clusters are not mixed, the ARI will be good, even if a large cluster is split into multiple small ones. The same phenomenon appears in the case of `Growing` compare to `fCUBT`.

	Method	Number of clusters K									
		1	2	3	4	5	6	7	8	9	10+
Scenario 1	fCUBT	-	-	-	-	98	2	-	-	-	-
	Growing	-	-	-	-	59	23	8	3	3	4
	FPCA+GMM	-	-	-	1	53	32	11	3	-	-
	FunHDDC	-	35	46	16	2	1	-	-	-	-
	Funclust	-	44	44	11	1	-	-	-	-	-
	k -means- d_1	-	-	15	15	60	2	7	7	-	-
	k -means- d_2	-	-	-	-	5	29	36	30	-	-
Scenario 2	fCUBT	-	-	-	-	70	18	10	2	-	-
	Growing	-	-	-	-	52	18	12	9	3	6
	FPCA+GMM	-	-	-	-	27	45	25	3	-	-
	FunHDDC	-	1	1	-	2	4	5	4	25	59
	Funclust	-	28	17	15	16	14	7	3	-	-
	k -means- d_1	-	-	-	2	5	8	18	67	-	-
	k -means- d_2	-	-	5	12	82	1	-	-	-	-
Scenario 3	fCUBT	-	-	-	-	67	24	7	2	-	-
	Growing	-	-	-	-	60	18	8	6	3	5
	FPCA+GMM	-	-	-	-	41	40	16	3	-	-
	FunHDDC	-	1	-	3	7	11	11	15	20	32
	Funclust	-	7	18	19	20	20	13	3	-	-
	k -means- d_1	-	-	-	-	3	14	21	62	-	-
	k -means- d_2	-	1	1	9	87	1	-	1	-	-

Table 1: Number of clusters selected for each model, expressed as a percentage over 500 simulations

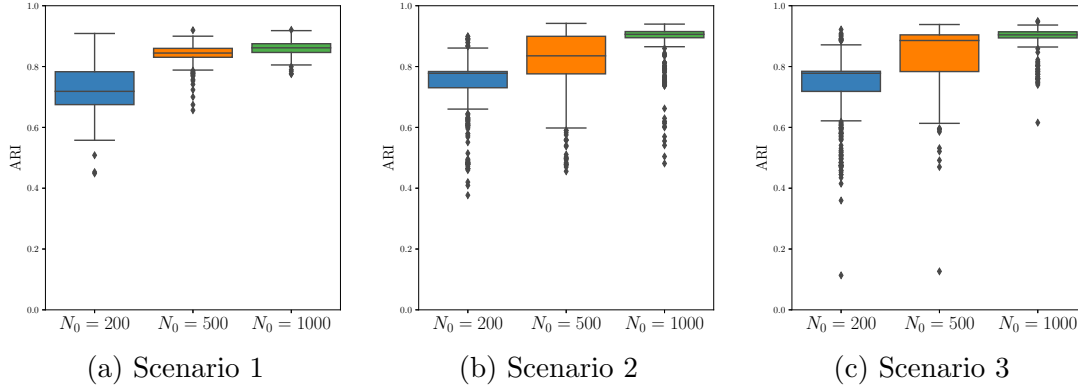


Figure 5: Estimation of ARI with respect to the size of the learning dataset when the tree is used as a supervised classifier.

5.1.3 Comments on the classification of new set of curves

We now analyze the performance of the algorithm for the classification of a new set of curves. For each of the simulated scenarios, we apply the following process. We learn a tree \mathfrak{T} as well as the partition \mathcal{U} from the learning set \mathcal{S}_{N_0} . Different sizes of learning sets are considered, $N_0 = 200, 500$ or 1000 . We generate a new set of data, \mathcal{S}_{N_1} , referred to as the online dataset, of size $N_1 = 1000$. As the data are simulated, we know the true labels of each observations within \mathcal{S}_{N_1} . We denote the true partition of \mathcal{S}_{N_1} by \mathcal{V} . We then classify new observations from the online set \mathcal{S}_{N_1} and denote the obtained partition by \mathcal{V}' . Partitions are compared using $\text{ARI}(\mathcal{V}, \mathcal{V}')$.

The simulations are performed 500 times, and the results are plotted in Figure 5. The three scenarios present similar patterns. Thus, when $N_0 = 200$, the partition \mathcal{U} obtained using the `fCUBT` algorithm is not able to capture all elements within each cluster. In this case, the ARI is less than 0.8. When $N_0 = 500$, the partition \mathcal{U} is now sufficiently accurate to represents the clusters ($\text{ARI} > 0.8$). However, the stability of the clusters is not guaranteed, given the large variance in the estimation of the ARI. Finally, when $N_0 = 1000$, we have both accurate partitioning \mathcal{U} ($\text{ARI} > 0.8$) and stable clusters (low variance).

5.1.4 A comparison with supervised methods

We also compare our functional clustering procedure with two supervised approaches available in [35], that are a Gaussian Process Classifier (GPC) and a Random Forest Classifier (`Random Forest`). See, for instance, [39] for the description of GPC. We first perform an MFPCA to extract features that explain 99% of the variance for each component $X^{(p)}$ within the data. We then fit GPC and `Random Forest` to the extracted features.

For each scenario, we generate samples of $N = 1000$ curves. We then randomly sample $2/3$ of the N curves to build the training subset and the remaining ones are gathered in the test subset. The supervised models are trained on the training subset, using the true value of K , and we predict the outcome on the test subset. The ARI is finally computed using the true labels and the prediction. For the `fCUBT` method, we consider the training subset to learn the clusters. We next predict the outcome on the test subset considering it as a set of new observations,

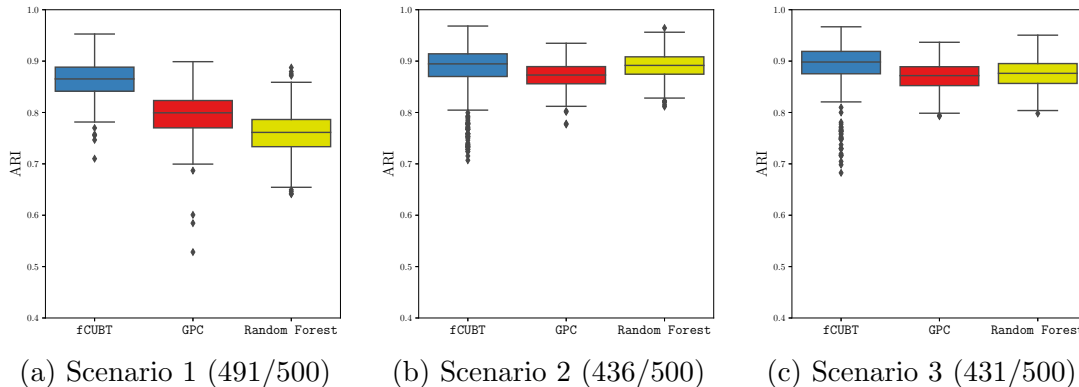


Figure 6: Estimation of ARI for the comparison with supervised models using 500 replications. The number of replications among the 500 experiments where **fCUBT** recovers the correct number of clusters K , is given between parenthesis

and using the procedure proposed in Section 4.3. For comparison purposes, the ARI for **fCUBT** is computed using only the replications where our approach detected the correct number of clusters K . We repeated the experiments 500 times, and the results are plotted in Figure 6. We remark that our unsupervised method is as good as the supervised ones when the true number of clusters is correctly estimated. The good performance of **fCUBT** compared to **Random Forest** could be explained by the model-based construction of our approach, while the random forest is nonparametric. The good performance of **fCUBT** compared to **GPC**, especially in Scenario 1, is somehow more surprising. An additional simulation experiment where **fCUBT** is used directly on the test subset, is reported in the Supplementary Material. Our approach performs well in that case also.

5.2 Real data analysis: the round dataset

In this section, our method is applied to a part of the round dataset [29], which are “naturalistic road user trajectories recorded at German roundabouts”. For our illustration, we consider a subset of the round dataset, that corresponds to one particular roundabout with four exits. Details are provided in the Supplementary Material.

The dataset we use contains 18 minutes of trajectories for road users at 7a.m. To describe the motion of the vehicles, we use six coordinates $X^{(p)}$ given by the position, the speed and the acceleration, each of them decomposed into a two-dimensional coordinate. The speed limit is 50km/h. In total, the dataset contains trajectories, velocities and accelerations for $N_0 = 348$ individual road users that passed through this roundabout during this period, recorded every 0.04s. The number of measurements for each curve varies from 131 to 1265. We rescale the measurement times for each of the 348 curves such that the first measurement corresponds to $t = 0$ and the last one to $t = 1$. Figure 7 presents a random sample of five observations extracted from the data. In order to have comparable results, we remove the pedestrians and the bicycles from the data. Moreover, curves with less than 200 or more than 800 measurements are also removed. The curves with less than 200 points are probably the road users that are present when the recording starts (or ends), and thus, their trajectory will not be complete. The curves with more than 800 are likely to be trajectories with inconsistency. Finally, there

are 311 remaining observations in the dataset. We aim to provide a clustering and give some physical interpretation of the clusters. Thus, our clustering procedure `fCUBT` is run on the cleaned data. We chose $J^{(p)} = 1$ for both the growing and joining step, and we set $K_{max} = 3$ and `minsize` = 20. It returns 23 groups. Figure 8 presents an example of a clusters we obtain. The trajectories with different entries and exits in the roundabout are well split into different clusters. Several clusters correspond to the same entries and exists and they are distinguished by different velocity and acceleration profiles. In particular, the algorithm differentiates the vehicles which stopped before entering the roundabout from those which did not (see Figure 9). Vehicles with atypical trajectories, such as those making a complete additional loop before taking the exit are also weel separated. The Gaussian assumption used in our model-based clustering seems reasonable in this application, according to the results from several normality tests we performed following [18]. The details are given in the Supplementary Material.

6 Extension to images

The `fCUBT` algorithm was introduced above for multivariate functional data which could be defined on different domains, possibly of different dimensions. In this section, we present the results of a simulation experiment with a process X with two components, one defined over a compact interval on the real line, the other one defined over a square in the plane. In such situations, the univariate `fPCA`, performed for each component for the computation of the `MFPCA` basis, is replaced by a suitable basis expansion for higher dimensional functions. In particular, the eigendecomposition of image data can be performed using the `FCP-TPA` algorithm for regularized tensor decomposition [1]. See also [19] and [49].

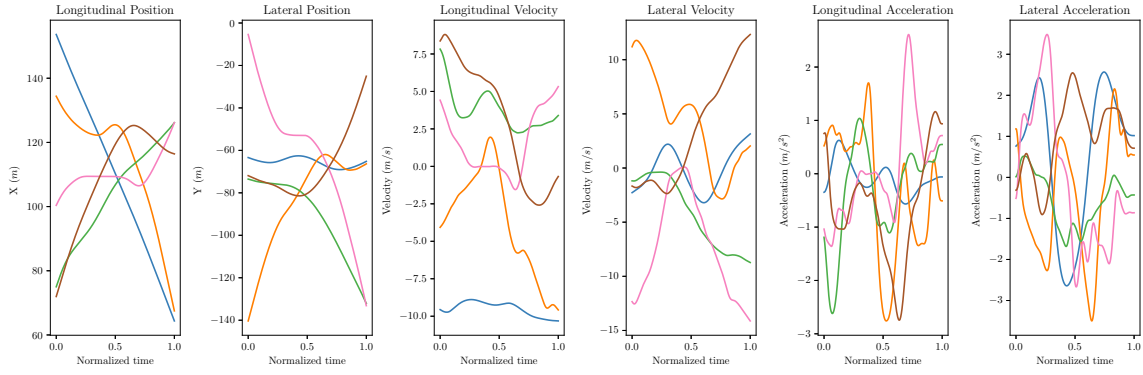
Scenario 4. As in the previous scenarios, the number of clusters is fixed at $K = 5$. Moreover, $P = 2$, $\mathcal{T}_1 = [0, 1]$ and $\mathcal{T}_2 = [0, 1] \times [0, 1]$. A sample of $N = 500$ curves is simulated according to the following model, for $s, t \in [0, 1]$:

$$\begin{aligned} X^{(1)}(t) &= a\phi_1(t) + b\phi_2(t) + c\phi_3(t), \\ X^{(2)}(s, t) &= d\phi_1(s)\phi_1(t) + e\phi_1(s)\phi_2(t) + f\phi_2(s)\phi_1(t) + g\phi_2(s)\phi_2(t), \end{aligned}$$

where ϕ_k 's are the eigenfunctions of the Wiener process. The coefficients a, b, c, d, e, f, g are random normal variables with parameters defined in Table 2. The mixing proportions are taken to be equal. The noisy curves are observed over 100 equidistant points and the noisy images are observed over a 2-D grid of 100×100 points. The measurement errors are introduced as in (3). The errors for $X^{(1)}$ are independent, zero-mean, Gaussian variables of variance $\sigma^2 = 0.05$, while the errors for $X^{(2)}$ are bivariate, zero-mean, Gaussian vectors with independent components of variance $\sigma^2 = 0.05$. The experiment was repeated 500 times.

By construction, the clusters cannot be retrieved only using only the noisy curves or the noisy images. Thus, the clustering algorithm has to considered these features for the grouping. Examples of realizations from this simulation experiment are shown in Figure 10.

The results of the `fCUBT` procedure are given in the Table 3a and 3b for both the estimated number of clusters and the ARI. We remark that in more than half of the cases, our algorithm estimates the number of clusters correctly. However, as cluster 4 and 5 are hard to discriminate, it returns four estimated clusters. This phenomenon is also reflected in the ARI results. In fact, the ARI presents a bimodal distribution where one mode is centered around 0.98 and the



(a)

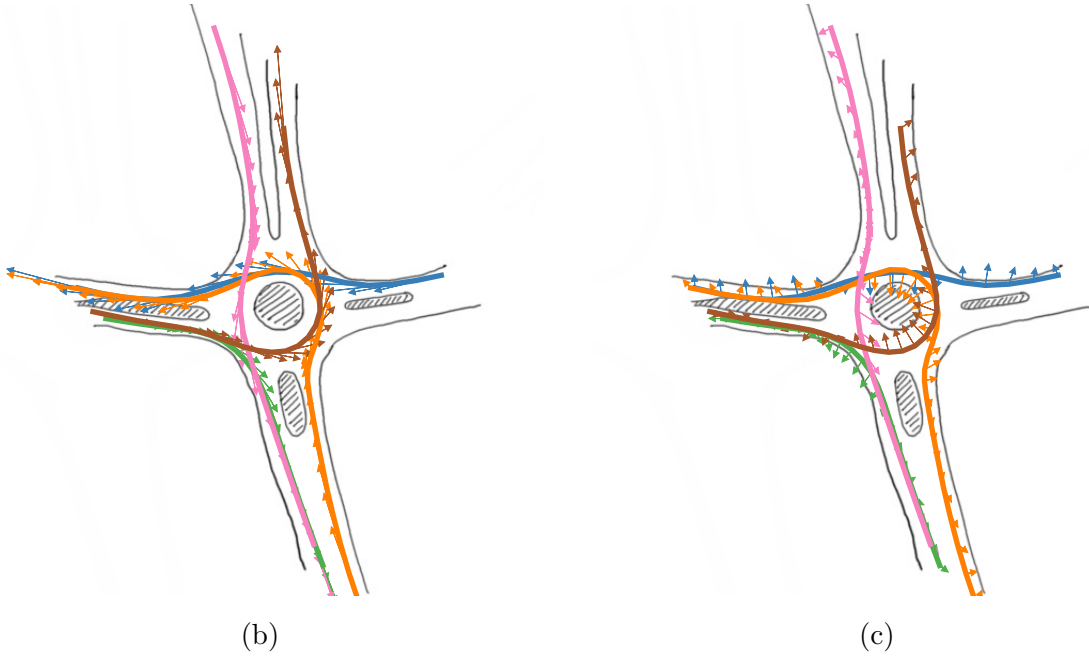


Figure 7: roundD dataset illustration – a sample of five trajectories (a). The trajectories in the roundabout reference frame where the arrows represent the magnitude and direction of the velocity (b) and acceleration (c).

	Coefficients (mean / std)						
	a	b	c	d	e	f	g
Cluster 1	3, 0.5	2, 1.66	1, 1.33	4, 1	0, 0.5	0, 0.1	-2, 0.05
Cluster 2	1, 0.5	-2, 1	0, 1	4, 0.8	0, 0.7	0, 0.08	-2, 0.07
Cluster 3	1, 0.4	-2, 0.8	0, 0.8	-3, 1	-4, 0.5	0, 0.1	0, 0.05
Cluster 4	-2, 1	0, 2	-1, 2	0, 0.1	2, 0.1	0, 0.05	0, 0.025
Cluster 5	-2, 0.2	0, 0.5	-1, 0.5	0, 2	2, 1	0, 0.2	1, 0.1

Table 2: Scenario 4 – Coefficient for $X^{(1)}(t)$ and $X^{(2)}(s, t)$.

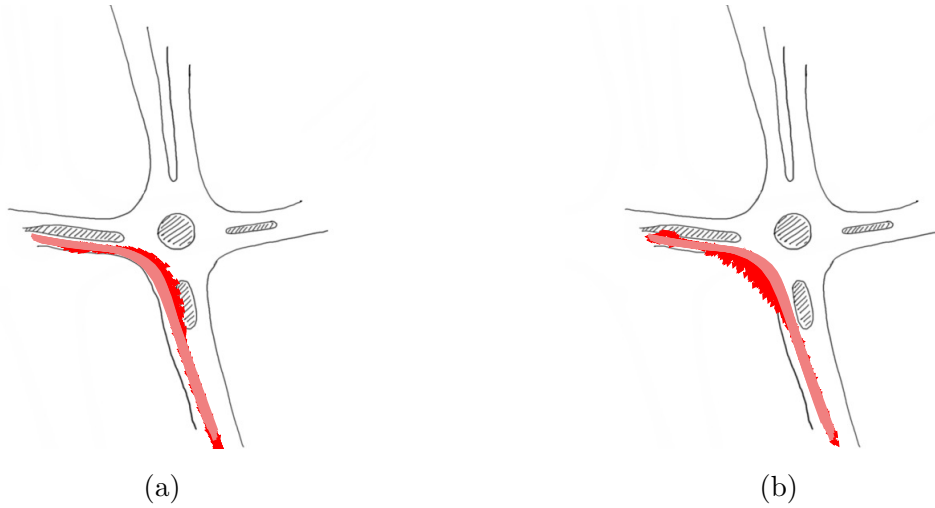


Figure 8: roundD dataset: An example of a cluster found using the `fCUBT` method. The trajectories are plotted in the roundabout reference frame. Each red curve represents a trajectory of an observation and the red arrows represent the magnitude and direction of the velocity (a) and acceleration (b).

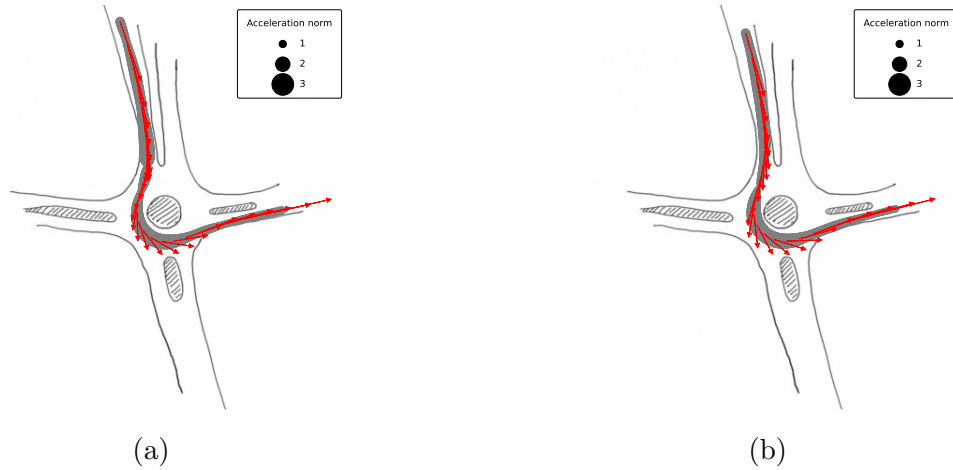


Figure 9: roundD dataset: Two different clusters with similar trajectory shape but with different velocity and acceleration profiles: (a) Stop when arriving at the roundabout. (b) Do not stop when arriving at the roundabout.

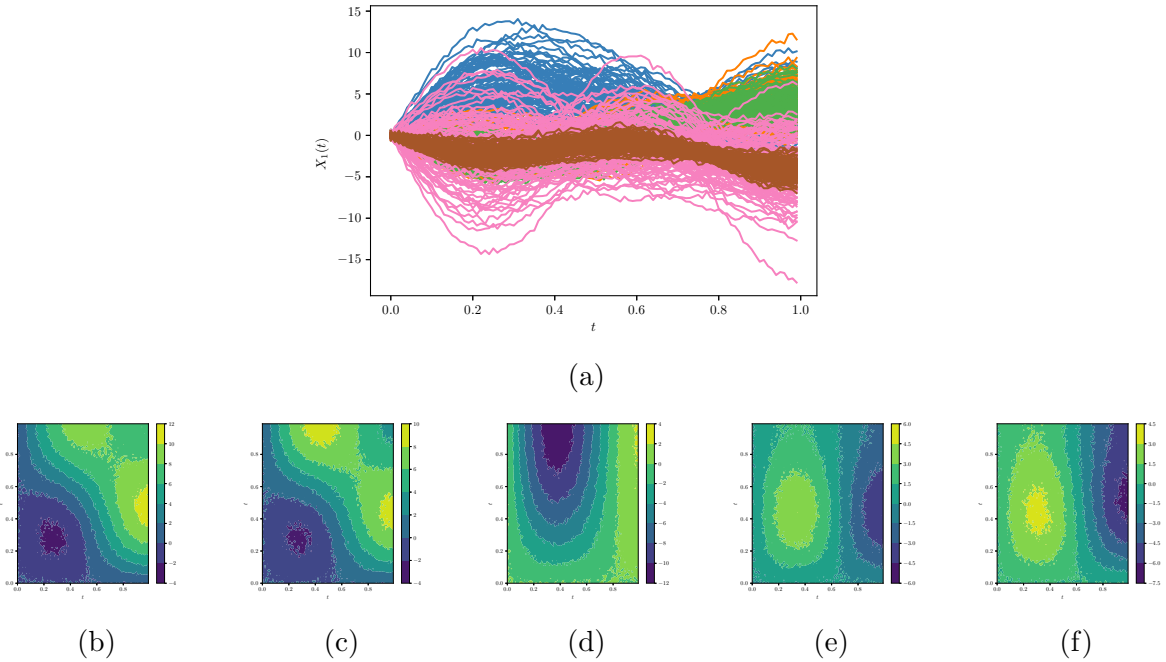


Figure 10: Examples of simulated data for Scenario 4 : (a) 500 realizations of the process $X^{(1)}$. (b)–(f) One realization of each of the clusters from the process $X^{(2)}$.

other one around 0.77. It appears that when the ARI is around 0.98, the number of clusters is well estimated, while when the ARI is close to 0.77, it is not. Nevertheless, even if in some replications, the number of clusters is wrongly estimated, overall the results are good. This simulation experiment provides strong evidence that our algorithm could also be used in such complex situations to which, apparently, the existing clustering algorithms have not yet been extended.

Number of clusters K	4	5	6	9
fCUBT	43	52	4	1

(a) Scenario 4 – Number of clusters selected for fCUBT for 500 simulations as a percentage.

Quantile	0.0	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9	1.0
fCUBT	0.70	0.76	0.77	0.77	0.78	0.78	0.96	0.98	0.99	0.99	1.0

(b) Scenario 4 – Quantile of the ARI for 500 simulations.

Table 3: Results for the Scenario 4.

7 Conclusion

The fCUBT algorithm has been proposed which is a model-based clustering method for functional data based on unsupervised binary trees. It works both on univariate and multivariate functional data defined in possibly different, multidimensional domains. The method is particularly suitable for finding the correct number of clusters within the data with respect to the model assumption. When the complete tree has been grown, fCUBT can be used for supervised classification. The open-source implementation can be accessed at <https://github.com/StevenGolovkine/FDApy> while scripts to reproduce the simulation and real-data analysis are at <https://github.com/StevenGolovkine/fcubt>.

A Reviewer asked about whether the Gaussian assumption imposed in model (2.2) is strict. First, one can use, for instance, the tests proposed in [18] to check this assumption using the terminal nodes of the tree, before or after the joining step. Some evidence of the effectiveness of these tests is provided in the Supplement. On the other hand, we also provide in the Supplement some simulation-based evidence on the robustness of our algorithm to departures from the Gaussian assumption. The conclusion of our simulation experiments is that, in general, the performance of our algorithm diminishes when the Gaussianity condition is not met, but remains good and comparable to that of the competitors, at least for some types of departures.

Acknowledgment

The authors wish to thank Groupe Renault and the ANRT (French National Association for Research and Technology) for their financial support via the CIFRE convention no. 2017/1116. Valentin Patilea gratefully acknowledges support from the Joint Research Initiative “Models and mathematical processing of very large data” under the aegis of Risk Foundation, in partnership with MEDIAMETRIE and GENES, France.

A Lemmas and Proofs

Proof of the Lemma 1. Using the linearity of the inner product, we may rewrite for each $j \geq 1, c_j$ as

$$c_j = \langle X - \mu, \psi_j \rangle = \sum_{k=1}^K \langle \mu_k, \psi_j \rangle \mathbf{1}_{\{Z=k\}} - \langle \mu, \psi_j \rangle + \sum_{l \geq 1} \xi_l \langle \phi_l, \psi_j \rangle.$$

Since a linear combination of independent Gaussian distributions is still Gaussian, the conditional distribution $c_j | Z = k$ has a Gaussian distribution for all $k \in \{1, \dots, K\}, j \geq 1$. Moreover, by the definition of X and the linearity of the inner product, for any $i, j \geq 1$ and $k \in \{1, \dots, K\}$,

$$\begin{aligned} \mathbb{E}(c_j | Z = k) &= \sum_{k'=1}^K \langle \mu_{k'}, \psi_j \rangle \mathbb{E}(\mathbf{1}_{\{Z=k'\}} | Z = k) - \langle \mu, \psi_j \rangle + \sum_{l \geq 1} \mathbb{E}(\xi_l | Z = k) \langle \phi_l, \psi_j \rangle \\ &= \langle \mu_k - \mu, \psi_j \rangle. \end{aligned}$$

Next, for any $i, j \geq 1$ and $k \in \{1, \dots, K\}$,

$$\begin{aligned} \text{Cov}(c_i, c_j | Z = k) &= \mathbb{E}(c_i c_j | Z = k) - \mathbb{E}(c_i | Z = k) \mathbb{E}(c_j | Z = k) \\ &= \sum_{p=1}^P \sum_{q=1}^P \int_{[0,1]} \int_{[0,1]} \mathbb{E} \left((X - \mu)^{(p)}(s_p) (X - \mu)^{(q)}(t_q) | Z = k \right) \psi_i^{(p)}(s_p) \psi_j^{(q)}(t_q) ds_p dt_q \\ &\quad - \langle \mu_k - \mu, \psi_i \rangle \langle \mu_k - \mu, \psi_j \rangle. \end{aligned}$$

By definition, for any $1 \leq p, q \leq P$,

$$\begin{aligned} &\mathbb{E} \left((X - \mu)^{(p)}(s_p) (X - \mu)^{(q)}(t_q) | Z = k \right) \\ &= \sum_{k'=1}^K \sum_{k''=1}^K \mu_{k'}^{(p)}(s_p) \mu_{k''}^{(q)}(t_q) \mathbb{E}(\mathbf{1}_{\{Z=k'\}} | Z = k) \mathbb{E}(\mathbf{1}_{\{Z=k''\}} | Z = k) \\ &\quad + \sum_{k'=1}^K \mu_{k'}^{(p)}(s_p) \sum_{j \geq 1} \phi_j^{(q)}(t_q) \mathbb{E}(\xi_j \mathbf{1}_{\{Z=k'\}} | Z = k) \\ &\quad + \sum_{k''=1}^K \mu_{k''}^{(q)}(t_q) \sum_{l \geq 1} \phi_l^{(p)}(s_p) \mathbb{E}(\xi_l \mathbf{1}_{\{Z=k''\}} | Z = k) \\ &\quad + \sum_{j \geq 1} \sum_{l \geq 1} \phi_j^{(p)}(s_p) \phi_l^{(q)}(t_q) \mathbb{E}(\xi_j \xi_l | Z = k) \\ &\quad - \mu^{(q)}(t_q) \sum_{k'=1}^K \mu_{k'}^{(p)}(s_p) \mathbb{E}(\mathbf{1}_{\{Z=k'\}} | Z = k) \\ &\quad - \mu^{(p)}(s_p) \sum_{k''=1}^K \mu_{k''}^{(q)}(t_q) \mathbb{E}(\mathbf{1}_{\{Z=k''\}} | Z = k) \\ &\quad - \mu^{(p)}(s_p) \sum_{l \geq 1} \phi_l^{(q)}(t_q) \mathbb{E}(\xi_l | Z = k) \\ &\quad - \mu^{(q)}(t_q) \sum_{l \geq 1} \phi_l^{(p)}(s_p) \mathbb{E}(\xi_l | Z = k) \\ &\quad + \mu^{(p)}(s_p) \mu^{(q)}(t_q) \\ &= \mu^{(p)}(s_p) \mu^{(q)}(t_q) + \mu_k^{(p)}(s_p) \mu_k^{(q)}(t_q) - \mu^{(p)}(s_p) \mu_k^{(q)}(t_q) - \mu_k^{(p)}(s_p) \mu^{(q)}(t_q) \\ &\quad + \sum_{l \geq 1} \sigma_{kl}^2 \phi_l^{(p)}(s_p) \phi_l^{(q)}(t_q) \\ &= \left(\mu_k^{(p)}(s_p) - \mu^{(p)}(s_p) \right) \left(\mu_k^{(q)}(t_q) - \mu^{(q)}(t_q) \right) + \sum_{l \geq 1} \sigma_{kl}^2 \phi_l^{(p)}(s_p) \phi_l^{(q)}(t_q). \end{aligned}$$

Thus,

$$\text{Cov}(c_i, c_j | Z = k) = \langle \mu_k - \mu, \psi_i \rangle \langle \mu_k - \mu, \psi_j \rangle + \sum_{l \geq 1} \sigma_{kl}^2 \langle \phi_l, \psi_i \rangle \langle \phi_l, \psi_j \rangle$$

$$\begin{aligned}
& - \langle \mu_k - \mu, \psi_i \rangle \langle \mu_k - \mu, \psi_j \rangle \\
& = \sum_{l \geq 1} \sigma_{kl}^2 \langle \phi_l, \psi_i \rangle \langle \phi_l, \psi_j \rangle.
\end{aligned}$$

Taking $i = j$ in the conditional covariance, we deduce

$$\tau_{kj}^2 = \text{Var}(c_j | Z = k) = \sum_{l \geq 1} \sigma_{kl}^2 \langle \phi_l, \psi_j \rangle^2.$$

For the marginal distribution of the c_j , the zero-mean is obtained as follows:

$$\mathbb{E}(c_j) = \sum_{k=1}^K \mathbb{P}(Z = k) \mathbb{E}(c_j | Z = k) = \sum_{k=1}^K p_k \langle \mu_k - \mu, \psi_j \rangle = 0.$$

For the marginal covariance, we can write

$$\begin{aligned}
\text{Cov}(c_i, c_j) & = \mathbb{E}(c_i c_j) \\
& = \sum_{k=1}^K p_k \mathbb{E}(c_i c_j | Z = k) \\
& = \sum_{k=1}^K p_k (\text{Cov}(c_i, c_j | Z = k) + \mathbb{E}(c_i | Z = k) \mathbb{E}(c_j | Z = k)) \\
& = \sum_{k=1}^K p_k \left(\sum_{l \geq 1} \langle \phi_l, \psi_i \rangle \langle \phi_l, \psi_j \rangle \sigma_{kl}^2 + \langle \mu_k - \mu, \psi_i \rangle \langle \mu_k - \mu, \psi_j \rangle \right).
\end{aligned}$$

This concludes the proof. ■

In applications, one cannot use an infinite number of terms in the representation of the process X , and has to truncate such a representation. The following lemma shows that such a truncation could be arbitrarily accurate.

Lemma A.1 *Let X be defined as in (2.2) for some orthonormal basis $\{\phi_j\}_{j \geq 1}$. Let $\{\psi_j\}_{j \geq 1}$ be another orthonormal basis in \mathcal{H} with to which X has the decomposition*

$$X(\mathbf{t}) = \sum_{j \geq 1} c_j \psi_j(\mathbf{t}), \quad \mathbf{t} \in \mathcal{T}.$$

Then

$$\lim_{J \rightarrow \infty} \mathbb{E}(\|X - X_{\lceil J} \|^2) = 0, \quad \text{where } X_{\lceil J}(t) = \sum_{j=1}^J c_j \psi_j(\mathbf{t}), \mathbf{t} \in \mathcal{T}.$$

Proof of Lemma A.1. Given $\{\psi_j\}_{j \geq 1}$ an orthonormal basis of \mathcal{H} , X can be written in the

form $X(t) = \sum_{j \geq 1} c_j \psi_j(t)$, $t \in \mathcal{T}$, with random variables $c_j = \langle X - \mu, \psi_j \rangle$. Then

$$\begin{aligned}
\|X - X_{[J]}\|^2 &= \left\| \sum_{j=J+1}^{\infty} c_j \psi_j \right\|^2 \\
&= \sum_{p=1}^P \int_{\mathcal{T}_p} \left(\sum_{j=J+1}^{\infty} c_j \psi_j^p(t) \right) \left(\sum_{j'=J+1}^{\infty} c_{j'} \psi_{j'}^p(t) \right) dt \\
&= \sum_{j=J+1}^{\infty} \sum_{j'=J+1}^{\infty} c_j c_{j'} \langle \psi_j, \psi_{j'} \rangle \\
&= \sum_{j=J+1}^{\infty} c_j^2.
\end{aligned}$$

Moreover,

$$\mathbb{E} \left(\sum_{j \geq 1} c_j^2 \right) = \mathbb{E} \left(\sum_{j \geq 1} \langle X - \mu, \psi_j \rangle^2 \right) = \mathbb{E} (\langle X - \mu \rangle^2) < \infty.$$

From this, and the fact that the remainder of a convergent series tends to zero, we have

$$\mathbb{E} (\|X - X_{[J]}\|^2) = \mathbb{E} \left(\sum_{j=J+1}^{\infty} c_j^2 \right) \xrightarrow{J \rightarrow \infty} 0.$$

This concludes the proof. ■

The following lemma shows that the MFPCA basis is the one which will induce the most accurate truncation for a given truncation number J . Therefore, among the workable bases one could use in practice, the MFPCA basis is likely to be a privileged one.

Lemma A.2 *Let X be defined as in (2.2). Let $\{\psi_j\}_{j \geq 1}$ be some orthonormal basis in \mathcal{H} and $\{\varphi_j\}_{j \geq 1}$ be the MFPCA basis. Let μ be the mean curve as defined in (1). Then, for any $J \geq 1$ such that $\lambda_1 > \lambda_2 > \dots > \lambda_J > \lambda_{J+1}$,*

$$\mathbb{E} \left(\left\| X - \mu - \sum_{j=1}^J \langle X - \mu, \psi_j \rangle \psi_j \right\|^2 \right) \geq \mathbb{E} \left(\left\| X - \mu - \sum_{j=1}^J c_j \varphi_j \right\|^2 \right).$$

Proof of Lemma A.2. First, let us note that, since the bases are orthogonal, the minimization of the truncation error for a given J is equivalent to the maximization of the sum of the variances

$\langle\langle \Gamma\psi_j, \psi_j \rangle\rangle$, $1 \leq j \leq J$. Moreover, for each $j \geq 1$, we have

$$\begin{aligned}
& \mathbb{E} [\langle\langle X - \mu, \varphi_j \rangle\rangle^2] \\
&= \mathbb{E} \left[\left(\sum_{p=1}^P \langle (X - \mu)^{(p)}, \varphi_j^{(p)} \rangle_2 \right) \left(\sum_{q=1}^P \langle (X - \mu)^{(q)}, \varphi_j^{(q)} \rangle_2 \right) \right] \\
&= \mathbb{E} \left[\left(\sum_{p=1}^P \int_{[0,1]} (X - \mu)^{(p)}(s_p) \varphi_j^{(p)}(s_p) ds_p \right) \left(\sum_{q=1}^P \int_{[0,1]} (X - \mu)^{(q)}(t_q) \varphi_j^{(q)}(t_q) dt_q \right) \right] \\
&= \sum_{p=1}^P \int_{[0,1]} \sum_{q=1}^P \int_{[0,1]} \mathbb{E} \left[(X - \mu)^{(p)}(s_p) (X - \mu)^{(q)}(t_q) \right] \varphi_j^{(p)}(s_p) \varphi_j^{(q)}(t_q) ds_p dt_q \\
&= \sum_{p=1}^P \int_{[0,1]} \sum_{q=1}^P \int_{[0,1]} C_{p,q}(s_p, t_q) \varphi_j^{(p)}(s_p) \varphi_j^{(q)}(t_q) ds_p dt_q \\
&= \sum_{q=1}^P \int_{[0,1]} \sum_{p=1}^P \langle C_{p,q}(\cdot, t_q), \varphi_j^{(p)(\cdot)} \rangle_2 \varphi_j^{(q)}(t_q) dt_q \\
&= \sum_{q=1}^P \int_{[0,1]} \langle\langle C_{\cdot,q}(\cdot, t_q), \varphi_j(\cdot) \rangle\rangle \varphi_j^{(q)}(t_q) dt_q \\
&= \sum_{q=1}^P \int_{[0,1]} (\Gamma\varphi_j)^{(q)}(t_q) \varphi_j^{(q)}(t_q) dt_q \\
&= \sum_{q=1}^P \langle\langle (\Gamma\varphi_j)^{(q)}(t_j), \varphi_j^{(q)}(t_q) \rangle\rangle_2 \\
&= \langle\langle \Gamma\varphi_j, \varphi_j \rangle\rangle
\end{aligned}$$

Since the MfPCA basis is characterized by the property (2.3), for any orthonormal basis $\{\psi_j\}_{j \geq 1}$, we necessarily have

$$\sum_{j=1}^J \text{Var}(\langle\langle X - \mu, \varphi_j \rangle\rangle) = \sum_{j=1}^J \langle\langle \Gamma\varphi_j, \varphi_j \rangle\rangle \geq \sum_{j=1}^J \langle\langle \Gamma\psi_j, \psi_j \rangle\rangle.$$

This concludes the proof. ■

B Algorithms

An example of a maximal tree is given in Figure 11. It corresponds to a simulated dataset defined in Section 5, Scenario 1, where $J^{(p)}$ is set to explain 95% of the variance at each node of the tree, $K_{max} = 5$ and `minsize` = 10. This tree has six leaves, whereas Scenario 1 contains only five clusters. In this illustration, the joining step will be helpful to join the group with only 4 curves, corresponding to the node $\mathfrak{S}_{3,2}$, with another group, hopefully the node $\mathfrak{S}_{3,1}$.

Algorithm 1: Construction of a tree \mathfrak{T}

Input: A training sample $\mathcal{S}_{N_0} = \{X_1, \dots, X_{N_0}\} \subset \mathcal{H}$, $J^{(p)}$, K_{max} and `minsize`.

Initialization: Set $(\mathfrak{d}, \mathfrak{j}) = (0, 0)$ and $\mathfrak{S}_{0,0} = \mathcal{S}_{N_0}$.

Computation of the MFPCA components: Perform a MFPCA with J components on the data in the node $\mathfrak{S}_{\mathfrak{d},\mathfrak{j}}$ and get the set of eigenvalues $\Lambda_{\mathfrak{d},\mathfrak{j}}$ associated with a set of eigenfunctions $\Phi_{\mathfrak{d},\mathfrak{j}}$. Build the matrix $C_{\mathfrak{d},\mathfrak{j}}$ defined in (4.1).

Estimation of the number of clusters: For each $K = 1, \dots, K_{max}$, fit K -components GMM using an EM algorithm on the columns of the matrix $C_{\mathfrak{d},\mathfrak{j}}$. The models are denoted by $\{\mathcal{M}_1, \dots, \mathcal{M}_{K_{max}}\}$. The number of mixture components is estimated by $\widehat{K}_{\mathfrak{d},\mathfrak{j}}$ defined in (4.1) using the BIC.

Stopping criterion: Test if the node indexed by $(\mathfrak{d}, \mathfrak{j})$ is a terminal node, that is if $\widehat{K}_{\mathfrak{d},\mathfrak{j}} = 1$ or if there are less than `minsize` elements in $\mathfrak{S}_{\mathfrak{d},\mathfrak{j}}$. If the node is terminal, then stop the construction of the tree for this node, otherwise go to the next step.

Children nodes construction: A non-terminal node indexed by $(\mathfrak{d}, \mathfrak{j})$ is split into two subnodes as follows:

1. Fit a 2-component GMM using an EM algorithm.
2. For each element of $\mathfrak{S}_{\mathfrak{d},\mathfrak{j}}$, compute the posterior probability to belong to the first component.
3. Form the children nodes as
 $\mathfrak{S}_{\mathfrak{d}+1,2\mathfrak{j}} = \{\text{elements of } \mathfrak{S}_{\mathfrak{d},\mathfrak{j}} \text{ with posterior probability } \geq 1/2\}$ and
 $\mathfrak{S}_{\mathfrak{d}+1,2\mathfrak{j}+1} = \mathfrak{S}_{\mathfrak{d},\mathfrak{j}} \setminus \mathfrak{S}_{\mathfrak{d}+1,2\mathfrak{j}}$.

Recursion: Continue the procedure by applying the **Computation of the MFPCA components** step to the nodes $(\mathfrak{d} + 1, 2\mathfrak{j})$ and $(\mathfrak{d} + 1, 2\mathfrak{j} + 1)$.

Output: A set of nodes $\{\mathfrak{S}_{\mathfrak{d},\mathfrak{j}}, 0 \leq \mathfrak{j} < 2^{\mathfrak{d}}, 0 \leq \mathfrak{d} < \mathfrak{D}\}$.

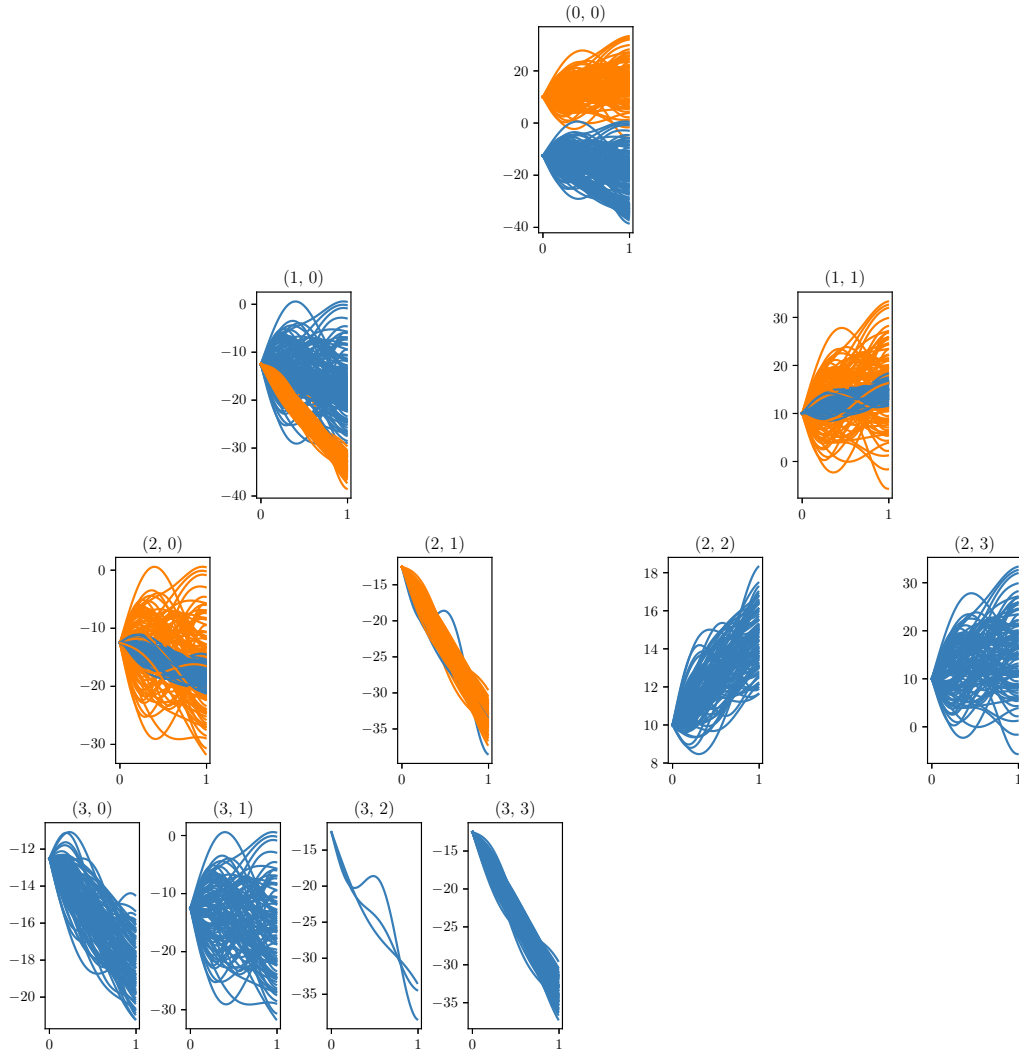


Figure 11: Illustration of maximal tree for Scenario 1 simulated data (different scale for each graph).

Algorithm 2: Joining step

Input: A set of nodes $\{\mathfrak{S}_{\mathfrak{d},j}, 0 \leq j < 2^{\mathfrak{d}}, 0 \leq \mathfrak{d} < \mathfrak{D}\}$, $J^{(p)}$, and K_{max} .

Initialization: Build the set of terminal nodes

$$V = \{\mathfrak{S}_{\mathfrak{d},j}, 0 \leq j < 2^{\mathfrak{d}}, 0 \leq \mathfrak{d} < \mathfrak{D} \mid \mathfrak{S}_{\mathfrak{d},j} \text{ is a terminal node}\}.$$

Creation of the graph: Build the set E defined in (4.2) and denote by \mathcal{G} the graph (V, E) . Associate with each edge $(\mathfrak{S}_{\mathfrak{d},j}, \mathfrak{S}_{\mathfrak{d}',j'})$ the value of the BIC that corresponds to $\widehat{K}_{(\mathfrak{d},j) \cup (\mathfrak{d}',j')}$.

Stopping criterion: If E is empty or V is reduced to a unique element, stop the algorithm.

Aggregation of two nodes: Let $(\mathfrak{S}_{\mathfrak{d},j}, \mathfrak{S}_{\mathfrak{d}',j'})$ be the edge with the maximum BIC value. Then, remove this edge and replace the associated vertice by $\mathfrak{S}_{\mathfrak{d},j} \cup \mathfrak{S}_{\mathfrak{d}',j'}$.

Recursion: Continue the procedure by applying the **Creation of the graph** step with $\{V \setminus \{\mathfrak{S}_{\mathfrak{d},j}, \mathfrak{S}_{\mathfrak{d}',j'}\}\} \cup \{\mathfrak{S}_{\mathfrak{d},j} \cup \mathfrak{S}_{\mathfrak{d}',j'}\}$.

Output: A partition \mathcal{U} of $\mathfrak{S}_{0,0}$ and labels associated to each element of \mathcal{S}_{N_0} .

Algorithm 3: Classify one observation

Input: A new realization \mathfrak{X} of the process X , the complete tree \mathfrak{T} and the partition \mathcal{U} .

for $\mathfrak{S}_{\mathfrak{d},j} \in \{\mathfrak{S}_{\mathfrak{d},j}, 0 \leq j < 2^{\mathfrak{d}}, 0 \leq \mathfrak{d} < \mathfrak{D}\}$ **do**

1. Compute the vector

$$\left(\langle \mathfrak{X} - \mu_{\mathfrak{d},j}, \varphi_{\mathfrak{d},j}^1 \rangle, \dots, \langle \mathfrak{X} - \mu_{\mathfrak{d},j}, \varphi_{\mathfrak{d},j}^J \rangle \right)^\top;$$

2. Compute the posterior probability to belong to each component of the 2-components GMM fitted on $\mathfrak{S}_{\mathfrak{d},j}$;

3. Compute the probability to be in $\mathfrak{S}_{\mathfrak{d},j}$ as

$$\mathbb{P}^*(\mathfrak{X} \in \mathfrak{S}_{\mathfrak{d},j}) = \prod_{\mathfrak{S} \in \text{Pa}(\mathfrak{S}_{\mathfrak{d},j})} \mathbb{P}^*(\mathfrak{X} \in \mathfrak{S} \mid \mathfrak{X} \in \text{Pa}(\mathfrak{S})).$$

end

for $U \in \mathcal{U}$ **do**

 | Compute $\mathbb{P}^*(\mathfrak{X} \in U)$.

end

Output: A label for \mathfrak{X} which is defined as $\arg \max_{U \in \mathcal{U}} \mathbb{P}^*(\mathfrak{X} \in U)$.

B.1 On the number of scores $J^{(p)}$ and J

In many situations, considering one score for each coordinate $X^{(p)}$ (*i.e.*, taking $J^{(p)} = 1$) suffices to detect a mixture structure. This allows J to be kept small when P is large. We provide here some details on why one score could be sufficient for revealing a mixture structure. For simplicity, let $P = 1$ and $K = 2$. Let

$$X(t) = \sum_{k=1}^K \mu_k(t) \mathbf{1}_{\{Z=k\}} + \sum_{j \geq 1} \xi_j \phi_j(t), \quad t \in \mathcal{T},$$

with some basis $\{\phi_j\}_{j \geq 1}$, and $k \in \{1, 2\}$. Let $\{\varphi_j\}_{j \geq 1}$ be the FPCA basis. By Lemma 1, considering only φ_1 , we have

$$c_1 = \langle\langle X - \mu, \varphi_1 \rangle\rangle, \quad \text{where } \mu(\cdot) = p_1 \mu_1(\cdot) + (1 - p_1) \mu_2(\cdot), \quad \text{and } p_1 = \mathbb{P}(Z = 1) \in (0, 1).$$

Then, for $k \in \{1, 2\}$,

$$c_1 | Z = k \sim \mathcal{N}(m_{k1}, \tau_{k1}^2), \quad \text{where } m_{k1} = \langle\langle \mu_k - \mu, \varphi_1 \rangle\rangle \quad \text{and } \tau_{k1}^2 = \sum_{l \geq 1} \langle\langle \phi_l, \varphi_1 \rangle\rangle^2 \sigma_{kl}^2.$$

Let us list the cases where the mixture structure can be detected by the first score.

Case 1. We have $m_{11} \neq m_{21}$, that is $\langle\langle \mu_1 - \mu_2, \varphi_1 \rangle\rangle \neq 0$. In particular, this requires $\mu_1 \neq \mu_2$.

Case 2. We have $m_{11} = m_{21}$, but $\tau_{11}^2 \neq \tau_{21}^2$, that is

$$\sum_{l \geq 1} \langle\langle \phi_l, \varphi_1 \rangle\rangle^2 \{\sigma_{1l}^2 - \sigma_{2l}^2\} \neq 0.$$

In particular, if $\langle\langle \phi_l, \varphi_1 \rangle\rangle = 0$ when $l > 1$, we need $\sigma_{11}^2 \neq \sigma_{22}^2$.

C Numerical illustrations

We use the Adjusted Rand Index to compare the different clustering algorithms. In the following, we provide a description of this criterion. When the true labels are available, the estimated partitions are compared with the true partition using the Adjusted Rand Index (ARI) [22], which is an ‘‘adjusted for chance’’ version of the Rand Index [38]. Let $\mathcal{U} = \{U_1, \dots, U_r\}$ and $\mathcal{V} = \{V_1, \dots, V_s\}$ be two different partitions of \mathcal{S}_N , *i.e.*

$$U_i \subset \mathcal{S}_N, \quad 1 \leq i \leq r, \quad V_j \subset \mathcal{S}_N, \quad 1 \leq j \leq s,$$

$$\mathcal{S}_N = \bigcup_{i=1}^r U_i = \bigcup_{j=1}^s V_j,$$

$$\text{and } U_i \cap U_{i'} = \emptyset, \quad 1 \leq i, i' \leq r, \quad V_j \cap V_{j'} = \emptyset, \quad 1 \leq j, j' \leq s.$$

We denote by $n_{ij} := |U_i \cap V_j|$, $1 \leq i \leq r$, $1 \leq j \leq s$, the number of elements of \mathcal{S}_N that are common to the sets U_i and V_j . $n_{i\cdot} := |U_i|$ (or $n_{\cdot j} := |V_j|$) then corresponds to the number of elements in U_i (or V_j). With these notations, the ARI is defined as

$$\text{ARI}(\mathcal{U}, \mathcal{V}) = \frac{\sum_{i=1}^r \sum_{j=1}^s \binom{n_{ij}}{2} - \left[\sum_{i=1}^r \binom{n_{i\cdot}}{2} \sum_{j=1}^s \binom{n_{\cdot j}}{2} \right] / \binom{N}{2}}{\frac{1}{2} \left[\sum_{j=1}^s \binom{n_{\cdot j}}{2} + \sum_{i=1}^r \binom{n_{i\cdot}}{2} \right] - \left[\sum_{i=1}^r \binom{n_{i\cdot}}{2} \sum_{j=1}^s \binom{n_{\cdot j}}{2} \right] / \binom{N}{2}}.$$

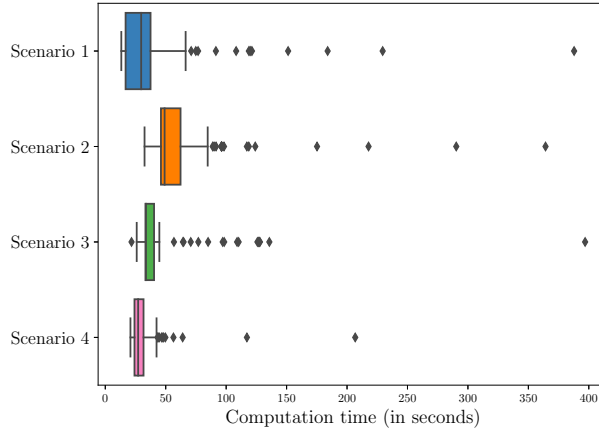


Figure 12: Computation times for all the scenarios from 100 experiments with $N = 1000$ for the Scenarios 1, 2 and 3 and $N = 500$ for the Scenario 4.

C.1 Computation times

In Figure 12 we report the computation times for $R = 100$ replications of each of the scenarios. For the Scenarios 1, 2 and 3, we consider $N = 1000$ and the curves are observed over 101 equidistant points. The $J^{(p)}$ parameter is set to represent 95% of the variance in the data, and K_{max} and $minsize$ are also set to their default values. For Scenario 4, where $P = 2$, we consider $N = 500$ and the curves are observed over 100 equidistant points and the images are observed over a 2-D grid of 100×100 points. We choose $J^{(1)}$ that represents 95% of the variance for the curves, and $J^{(2)} = 2$ for the images. We keep K_{max} and $minsize$ as default. In all situations considered, the computation time is inferior to one minute most of the time. All the computations were performed on a MacBook Pro mid-2014 with 2.6GHz Inter Core i5 with 8Go RAM.

C.2 Influence of $J^{(p)}$

The results in this section correspond to 500 experiments with $N = 1000$, for all Scenarios. In Figure 13 we present a comparison of the performance of the algorithm for different values of $J^{(p)}$. For Scenario 1, the performance does not improve as $J^{(p)}$ increases. For Scenario 2 and 3, larger $J^{(p)}$ leads to slightly larger ARI.

Table 4 presents the estimation of the number of clusters and ARI when $J^{(p)}$, and thus J , varies between the nodes. We use $J^{(p)} = 4$ for the first split and we then reduce to $J^{(p)} = 2$ the rest of the construction of the tree. We remark that the number of clusters is well estimated. The ARI does not really change compared to the case where we do not change $J^{(p)}$ through the tree. The investigation of an alternative data-driven choice of $J^{(p)}$ is left for future work.

C.3 On the Gaussian assumption

A Reviewer asked whether the multivariate Gaussian distribution is a strict assumption. Our Lemma 1 crucially uses the Gaussian assumption. If the ξ_j in (2.1) are not Gaussian, in general the change of basis does not preserve the distributions. If one assumes that the basis $\{\phi_j\}_{j \geq 1}$ is

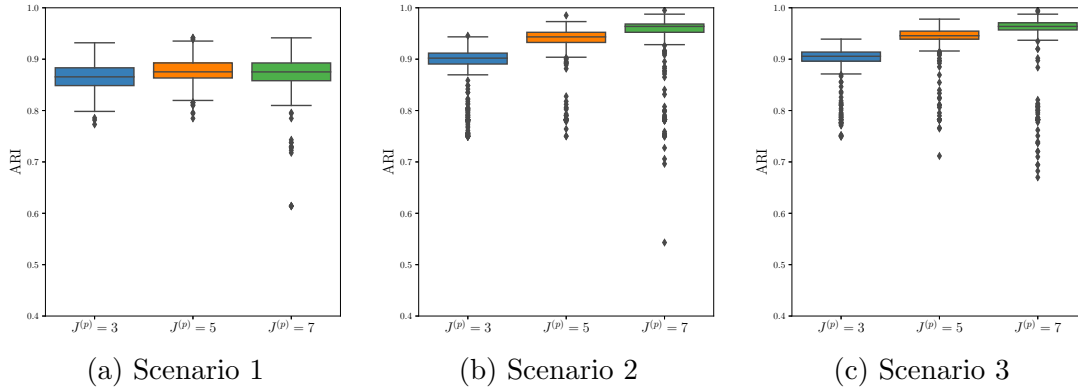


Figure 13: Estimation of ARI for the influence of $J^{(p)}$

Number of clusters	5	6	7	8
fCUBT	95	3	1	1

(a) Scenario 2 – Number of clusters selected for fCUBT for 500 simulations as a percentage.

Quantile	0.0	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9	1.0
fCUBT	0.59	0.87	0.89	0.90	0.90	0.90	0.91	0.91	0.92	0.92	0.94

(b) Scenario 2 – Quantile of the ARI for 500 simulations.

Table 4: Results for Scenario 2 with different $J^{(p)}$ per nodes

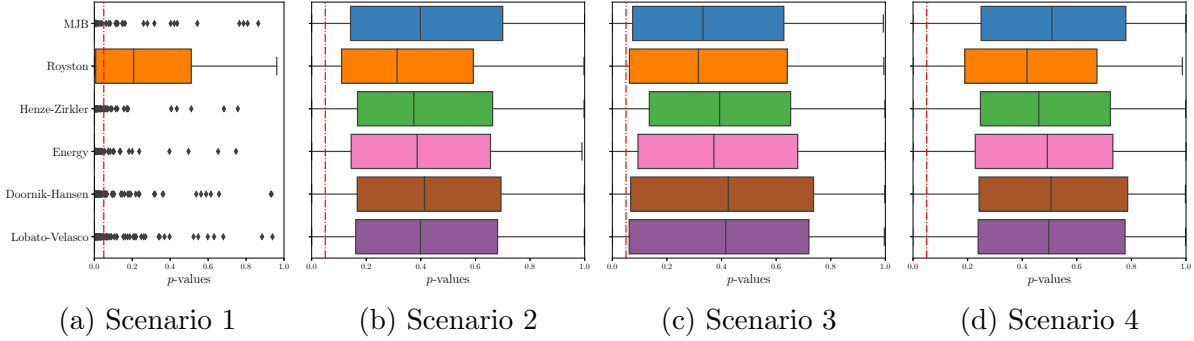


Figure 14: p -values for the different normality tests for different Scenarios. The red dashed line represents the 5% level.

given, it could be possible to consider mixture models with other distributions for the ξ_j . This would be a different modeling approach, which will be considered elsewhere.

The Gaussian assumption could be tested using, for instance, the tests proposed in [18]. We investigated the effectiveness of such tests in our simulation framework. We consider six multivariate normality tests: Mardia’s (MJB), Royston’s, Henze-Zirkler’s, Energy, Doornik-Hansen’s and Lobato-Velasco’s normality tests. The tests are run on the leaves of the grown trees for 100 replications of each of the four Scenarios considered above. As the number of leaves can be different for each tree, we test each leaf independently, and gather the results. Figure 14 reports the p -values of the different tests under the null hypothesis for our different scenarios, with $N = 1000$. For Scenario 1, only the Royston’s multivariate normality test performs satisfactory, that is in most of the replications it does not reject the Gaussian scores hypothesis at the 5% level. For Scenarios 2, 3 and 4, all the tests perform well and indicate the data are likely to our Gaussian assumption.

Finally, in reply to one Reviewer’s inquiry, we also investigate the robustness of our approach to departures from the Gaussian assumption. For this purpose, we reconsider Scenario 1 where the coefficients $c_{11}, c_{12}, \dots, c_{23}$ are generated according to a double exponential distribution with densities $f(x) = \exp(-|x/\beta|)/(2\beta)$, $x \in \mathbb{R}$, where the different values of the β parameter are fixed in a such way as to preserve the variances considered in (1). The results obtained from 500 replications are reported in Table 5.

We notice that our algorithm has a lesser performance for detecting the correct number of clusters, though it is able to do it in a majority of cases (53%). The quantiles of ARI are also worse than with the Gaussian coefficients $c_{11}, c_{12}, \dots, c_{23}$. However, the performance of the algorithm remains good and comparable to the performance of the competing approaches as we reported in Table 1, in the Gaussian coefficients setup. In conclusion, our approach seems to be robust with respect to some departures from the Gaussian assumption.

C.4 Another comparison with supervised methods

This part is similar to Section 5.1.4, except that, here, we only learn the clusters on the test subset for the fCUBT algorithm. We performed the simulations 500 times, and the results are plotted in Figure 15. Between parenthesis, we have written the number of times fCUBT gets the right number of clusters over the 500 simulations, and so the number of simulations we examine

Number of clusters	5	6	7	8	9
fCUBT	53	35	8	3	1

(a) Scenario 1 – Number of clusters selected for fCUBT for 500 simulations as a percentage.

Quantile	0.0	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9	1.0
fCUBT	0.49	0.65	0.68	0.69	0.71	0.72	0.73	0.74	0.75	0.77	0.82

(b) Scenario 1 – Quantile of the ARI for 500 simulations.

Table 5: Results for Scenario 1 with double exponential coefficients

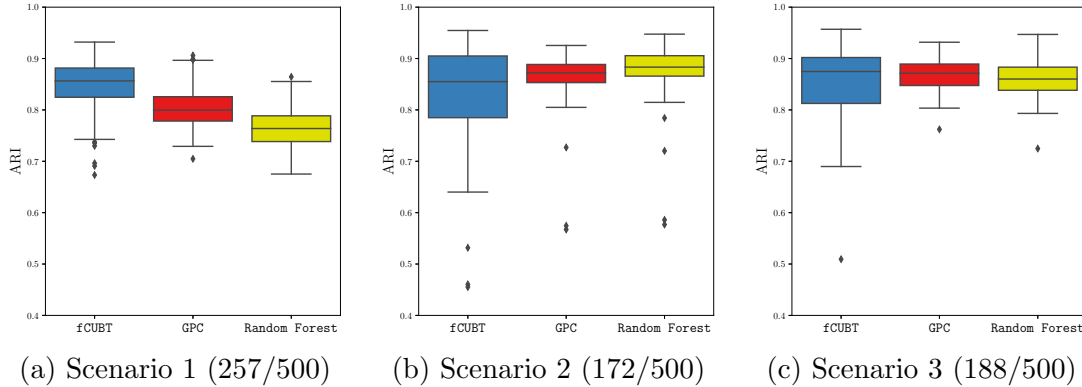


Figure 15: Estimation of ARI for the comparison with supervised models. The number of replications among the 500 where fCUBT recovers the correct number of clusters K is given between parenthesis

for the computation of the ARI. We point out that this retrieval percentage is much smaller than that in Section 5.1.4 but it is due to the fact that here, we only considers a dataset of size $N_1 = 330$ for applying our algorithms. We remark that our unsupervised method is as good as supervised ones when the true number of classes is found.

D Real data analysis: the round dataset

In this section, we provided details on the round dataset. The aerial view of the roundabout is presented in Figure 16. This dataset is part of a set of vehicle trajectory data provided by the Institute for Automotive Engineering (ika) in RWTH Aachen University. One may cite the highD dataset (about highways) [30] and the inD dataset (about intersections) [3], such others produced by ika. These datasets are particularly useful for studying the behavior of road users in some specific situations. They start to replace the Next Generation Simulation (NGSIM) study [15], widely used in traffic flow studies, as a benchmark for models about trajectory prediction or classification, because they provide more accurate data (see *e.g.* [24, 33, 32, 14, 2, 51] for some references).

Figure 17 presents the first eigenfunction for each coordinate $X^{(p)}$ in the first MFPCA at



Figure 16: roundD dataset – the considered roundabout. Source: Google Maps

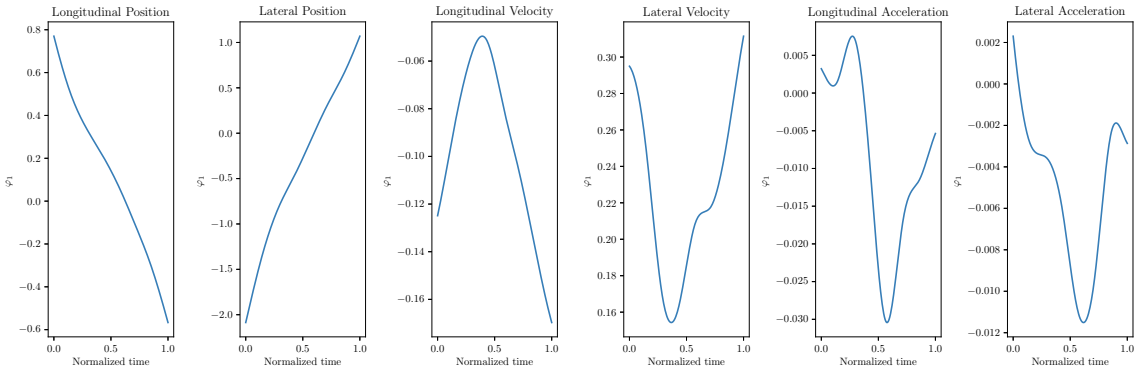


Figure 17: The first eigenfunction for each coordinate $X^{(p)}$ of the first node of the tree.

the beginning of the tree.

Figure 18 presents the first three levels of the tree, starting from the root, obtained using a small subsample (that represents 311 multivariate curves) of the roundD dataset. This sub-tree helps to understand how the clusters are built. The two colors represents the binary splitting at this node. We see that the different entry/exit combinations are already separated in the three steps. The split inside an entry/exit scenario is produced in the subsequent binary splits of the tree.

We tested whether the Gaussian assumption used for our model-based clustering is reasonable on the roundD dataset. We consider six multivariate normality tests: Mardia’s (MJB), Royston’s, Henze-Zirkler’s, Energy, Doornik-Hansen’s and Lobato- Velasco’s normality tests. The tests are performed on the leaves of the tree with more than 10 observations. Figure 19 presents the p -values computed for each test. In a majority of the cases, we do not reject the null hypothesis. The data are thus likely to have a mixed Gaussian distribution.

References

- [1] G. I. Allen. Multi-way functional principal components analysis. In *2013 5th IEEE International Workshop on Computational Advances in Multi-Sensor Adaptive Processing (CAMSAP)*, pages 220–223, Dec. 2013.
- [2] A. Bhattacharyya, M. Hanselmann, M. Fritz, B. Schiele, and C.-N. Straehle. Conditional Flow Variational Autoencoders for Structured Sequence Prediction. *arXiv:1908.09008 [cs, stat]*, Aug. 2020.
- [3] J. Bock, R. Krajewski, T. Moers, S. Runde, L. Vater, and L. Eckstein. The inD Dataset: A Drone Dataset of Naturalistic Road User Trajectories at German Intersections. *arXiv:1911.07602 [cs, eess]*, Nov. 2019.
- [4] C. Bouveyron and J. Jacques. Model-based Clustering of Time Series in Group-specific Functional Subspaces. *Advances in Data Analysis and Classification*, 5(4):281–300, 2011.
- [5] C. Bouveyron, E. Côme, and J. Jacques. The discriminative functional mixture model for a comparative analysis of bike sharing systems. *The Annals of Applied Statistics*, 9(4): 1726–1760, Dec. 2015.
- [6] C. Bouveyron, G. Celeux, T. B. Murphy, and A. E. Raftery. *Model-Based Clustering and Classification for Data Science: With Applications in R*. Cambridge Series in Statistical and Probabilistic Mathematics. Cambridge University Press, 2019.
- [7] R. J. Carroll, A. Delaigle, and P. Hall. Unexpected properties of bandwidth choice when smoothing discrete data for constructing a functional data classifier. *Annals of Statistics*, 41(6):2739–2767, Dec. 2013.
- [8] S.-S. Cheng, H.-M. Wang, and H.-C. Fu. A model-selection-based self-splitting Gaussian mixture learning with application to speaker identification. *EURASIP Journal on Advances in Signal Processing*, 2004:2626–2639, Jan. 2004.
- [9] A. Delaigle and P. Hall. Defining probability density for a distribution of random functions. *The Annals of Statistics*, 38(2):1171–1193, Apr. 2010.
- [10] A. Delaigle and P. Hall. Achieving near perfect classification for functional data. *Journal of the Royal Statistical Society. Series B (Statistical Methodology)*, 74(2):267–286, 2012.
- [11] A. Delaigle and P. Hall. Classification using censored functional data. *Journal of the American Statistical Association*, 108(504):1269–1283, 2013.
- [12] A. Delaigle, P. Hall, and T. Pham. Clustering functional data into groups by using projections. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 81(2): 271–304, 2019.
- [13] A. P. Dempster, N. M. Laird, and D. B. Rubin. Maximum Likelihood from Incomplete Data via the EM Algorithm. *Journal of the Royal Statistical Society. Series B (Methodological)*, 39(1):1–38, 1977.

- [14] F. Diehl, T. Brunner, M. T. Le, and A. Knoll. Graph Neural Networks for Modelling Traffic Participant Interaction. In *2019 IEEE Intelligent Vehicles Symposium (IV)*, pages 695–701, June 2019.
- [15] FHWA, U.S. Department of Transportation. NGSIM–Next Generation SIMulation, 2006.
- [16] R. Fraiman, B. Ghattas, and M. Svarc. Interpretable clustering using unsupervised binary trees. *Advances in Data Analysis and Classification*, 7(2), June 2013.
- [17] S. Golovkine, N. Klutchnikoff, and V. Patilea. Learning the smoothness of noisy curves with application to online curve estimation. *arXiv:2009.03652 [math, stat]*, Sept. 2020.
- [18] T. Górecki, L. Horvath, and P. Kokoszka. Tests of normality of functional data. *International Statistical Review*, 88(3):677–697, 2020.
- [19] C. Happ and S. Greven. Multivariate Functional Principal Component Analysis for Data Observed on Different (Dimensional) Domains. *Journal of the American Statistical Association*, 113(522):649–659, Apr. 2018.
- [20] L. Horvath and P. Kokoszka. *Inference for Functional Data with Applications*. Springer Series in Statistics. Springer-Verlag, New York, 2012.
- [21] X. Hu and F. Yao. Sparse Functional Principal Component Analysis in High Dimensions. *arXiv:2011.00959 [stat]*, Nov. 2020.
- [22] L. Hubert and P. Arabie. Comparing partitions. *Journal of Classification*, 2(1):193–218, Dec. 1985.
- [23] F. Ieva, A. M. Paganoni, D. Pigoli, and V. Vitelli. Multivariate functional clustering for the morphological analysis of electrocardiograph curves. *Journal of the Royal Statistical Society. Series C (Applied Statistics)*, 62(3):401–418, 2013.
- [24] R. Izquierdo, A. Quintanar, I. Parra, D. Fernandez-Llorca, and M. A. Sotelo. Vehicle Trajectory Prediction in Crowded Highway Scenarios Using Bird Eye View Representations and CNNs. *arXiv:2008.11493 [cs]*, Aug. 2020.
- [25] J. Jacques and C. Preda. Funclust: A curves clustering method using functional random variables density approximation. *Neurocomputing*, 112:164–171, July 2013.
- [26] J. Jacques and C. Preda. Functional data clustering: a survey. *Advances in Data Analysis and Classification*, 8(3):24, Jan. 2014.
- [27] J. Jacques and C. Preda. Model-based clustering for multivariate functional data. *Computational Statistics and Data Analysis*, 71:92–106, June 2014.
- [28] M. Kayano, K. Dozono, and S. Konishi. Functional Cluster Analysis via Orthonormalized Gaussian Basis Expansions and Its Application. *Journal of Classification*, 27(2):211–230, Sept. 2010.

- [29] R. Krajewski, T. Moers, J. Bock, L. Vater, and L. Eckstein. The round Dataset: A Drone Dataset of Road User Trajectories at Roundabouts in Germany. submitted.
- [30] R. Krajewski, J. Bock, L. Kloeker, and L. Eckstein. The highD Dataset: A Drone Dataset of Naturalistic Vehicle Trajectories on German Highways for Validation of Highly Automated Driving Systems. *arXiv:1810.05642 [cs, stat]*, Oct. 2018.
- [31] J. S. Marron, J. O. Ramsay, L. M. Sangalli, and A. Srivastava. Functional data analysis of amplitude and phase variation. *Statist. Sci.*, 30(4):468–484, 11 2015.
- [32] K. Messaoud, I. Yahiaoui, A. Verroust-Blondet, and F. Nashashibi. Non-local Social Pooling for Vehicle Trajectory Prediction. In *2019 IEEE Intelligent Vehicles Symposium (IV)*, pages 975–980, June 2019.
- [33] K. Messaoud, I. Yahiaoui, A. Verroust, and F. Nashashibi. Attention Based Vehicle Trajectory Prediction. *IEEE Transactions on Intelligent Vehicles*, 2020.
- [34] J. Park and J. Ahn. Clustering multivariate functional data with phase variation. *Biometrics*, 73(1):324–333, 2017.
- [35] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.
- [36] D. Pelleg and A. Moore. X-means: Extending K-means with Efficient Estimation of the Number of Clusters. In *In Proceedings of the 17th International Conf. on Machine Learning*, pages 727–734. Morgan Kaufmann, 2000.
- [37] J. Ramsay and B. W. Silverman. *Functional Data Analysis*. Springer Series in Statistics. Springer-Verlag, New York, 2 edition, 2005.
- [38] W. M. Rand. Objective Criteria for the Evaluation of Clustering Methods. *Journal of the American Statistical Association*, 66(336):846–850, 1971.
- [39] C. Rasmussen and C. Williams. *Gaussian Processes for Machine Learning*. Adaptive Computation and Machine Learning. MIT Press, Cambridge, MA, USA, Jan. 2006.
- [40] M. Reed and B. Simon. *Methods of Modern Mathematical Physics: Functional analysis*. Academic Press, 1980.
- [41] A. Schmutz, J. Jacques, and C. Bouveyron. *funHDDC: Univariate and Multivariate Model-Based Clustering in Group-Specific Functional Subspaces*, 2019. R package version 2.3.0.
- [42] A. Schmutz, J. Jacques, C. Bouveyron, L. Cheze, and P. Martin. Clustering multivariate functional data in group-specific functional subspaces. *Computational Statistics*, 2020.
- [43] G. Schwarz. Estimating the Dimension of a Model. *The Annals of Statistics*, 6(2):461–464, 1978.

- [44] M. Soueidatt, C. Preda, J. Jacques, and V. Kubicki. *Funclustering: A package for functional data clustering.*, 2014. R package version 1.0.1.
- [45] G. Staerman, P. Mozharovskyi, S. Cléménçon, and F. d’Alché Buc. Functional Isolation Forest. In *Asian Conference on Machine Learning*, pages 332–347. PMLR, Oct. 2019.
- [46] T. Tarpey and K. K. J. Kinateder. Clustering Functional Data. *Journal of Classification*, 20(1):093–114, May 2003.
- [47] S. Tokushige, H. Yadohisa, and K. Inada. Crisp and fuzzy k-means clustering algorithms for multivariate functional data. *Computational Statistics*, 22(1):1–16, Apr. 2007.
- [48] O. I. Traore, P. Cristini, N. Favretto-Cristini, L. Pantera, P. Vieu, and S. Viguier-Pla. Clustering acoustic emission signals by mixing two stages dimension reduction and non-parametric approaches. *Computational Statistics*, 34(2):631–652, June 2019.
- [49] J. Wang, R. K. W. Wong, and X. Zhang. Low-rank covariance function estimation for multidimensional functional data. *Journal of the American Statistical Association*, 0(0):1–14, 2020.
- [50] J.-L. Wang, J.-M. Chiou, and H.-G. Müller. Functional Data Analysis. *Annual Review of Statistics and Its Application*, 3(1):257–295, 2016.
- [51] Y. Wu, J. Hou, G. Chen, and A. Knoll. Trajectory Prediction Based on Planning Method Considering Collision Risk. In *2020 5th International Conference on Advanced Robotics and Mechatronics (ICARM)*, pages 466–470, Dec. 2020.
- [52] F. Yao, H.-G. Müller, and J.-L. Wang. Functional Data Analysis for Sparse Longitudinal Data. *Journal of the American Statistical Association*, 100(470):577–590, June 2005.
- [53] A. Z. Zambom, J. A. Collazos, and R. Dias. Selection of the Number of Clusters in Functional Data Analysis. *arXiv:1905.00977 [stat]*, May 2019.

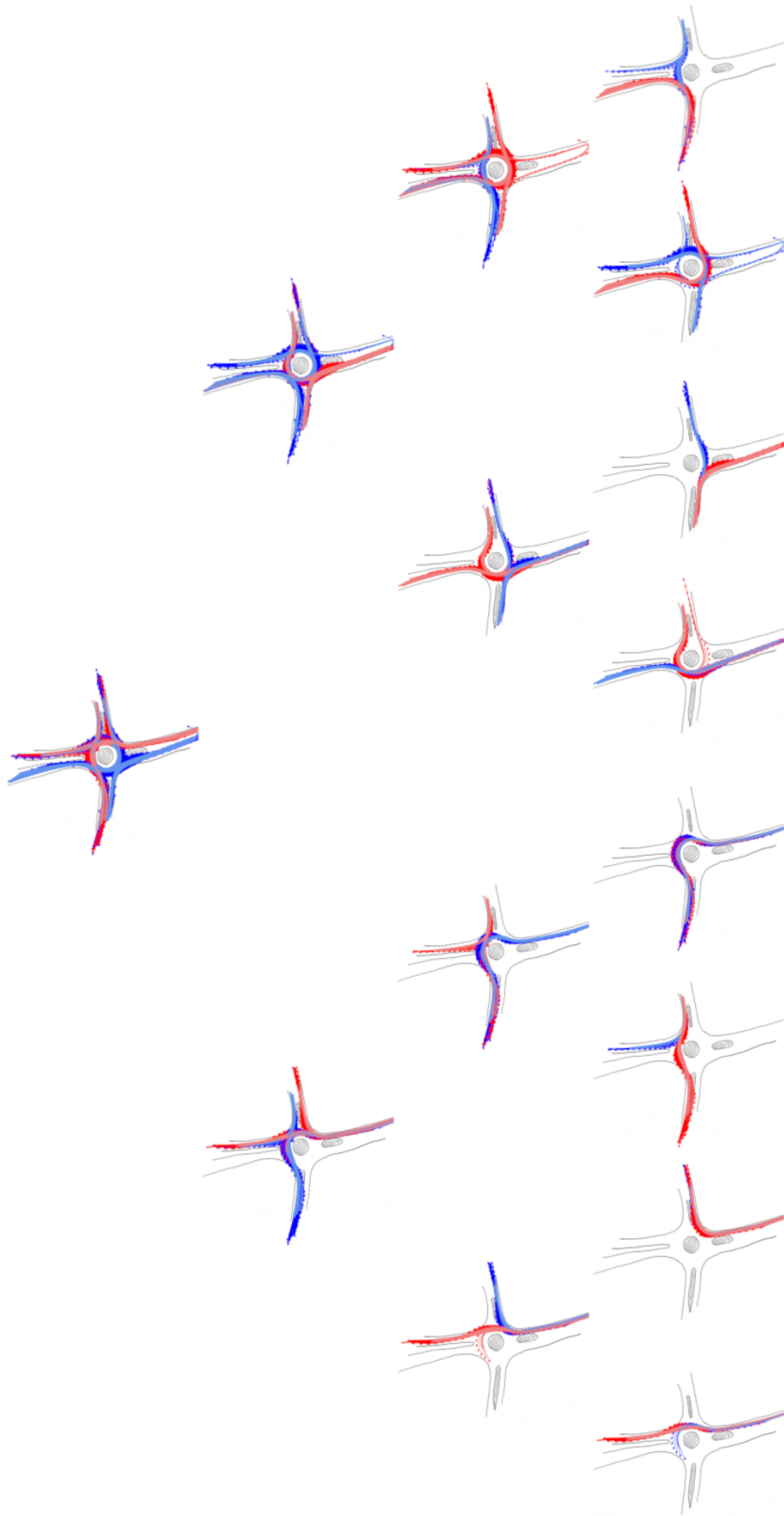


Figure 18: Beginning of the tree for the roundD dataset.

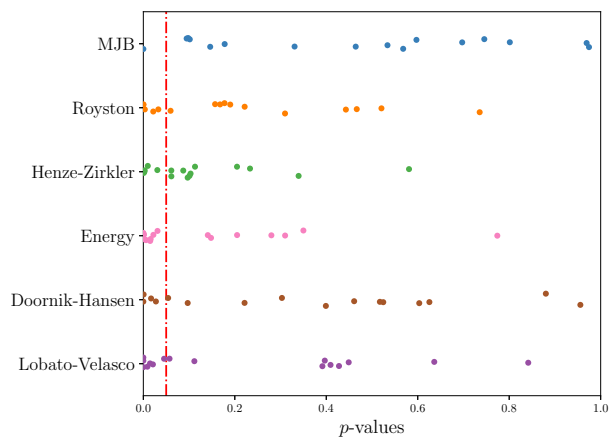


Figure 19: p -values for the different normality tests for the roundD dataset. Each point represents one leaf of the tree with more than 10 observations. The red dashed line represents the 5% level.