Clustering Functional Data Using a Predictive Likelihood

Tong Shan[1] and David B. Hitchcock[1*]

[1*]Department of Statistics, University of South Carolina, Columbia, SC, 29208, USA.

*Corresponding author(s). E-mail(s): hitchcock@stat.sc.edu;
Contributing authors: tshan@email.sc.edu;

Abstract

The analysis of functional data is an increasingly relevant part of statistics. The exploratory data analytic method of cluster analysis plays an important role in different fields, including those in which functional data are collected. To improve the accuracy when clustering functional data, we propose a predictive likelihood function which serves as an objective function to be optimized in order to determine the most appropriate clusters. To optimize the objective function over the space of clustering partitions, we produce a Markov chain of partitions. At each step, we generate via a simulated annealing algorithm new partitions for which the predictive likelihood is evaluated. A schedule for adjusting the temperature parameter enhances the efficiency of the chain. Compared with a functional K-means algorithm and the algorithm of the R function `funFEM`, our method can achieve higher performance as measured by the Rand index on simulated data. Our approach is also applied to two real data sets, a vertical density profile data set and a yeast gene data set, and it achieves meaningful clustering results.

Keywords: Cluster analysis; Simulated annealing; Density profile data; Yeast gene data

1 Introduction

Cluster analysis is a common methodology for learning from data in statistics and is widely used in marketing, biology, geography, and other fields. The goal of clustering is to group objects that have similar characteristics based on data measured on them.

Functional data analysis is the analysis of data that are curves (Ferraty and Vieu, 2006). Functional data can be treated as functions of time so that we observe measurements for every curve over some time domain $\mathcal{T}$ which without loss of generality can be taken to be the interval $[0, T]$. The trend exhibited by functional data is usually complex and not simply linear. This requires some choices be made when modeling the trends of the signal curves before the main analysis. In particular, we must choose an appropriate basis with which to represent the observed functional data.

In our approach, we will use predictive likelihood in functional data analysis. Predictive likelihood can be used to perform cluster analysis via choosing the clustering partition to maximize the predictive likelihood. By comparing the predictive likelihood values across a large set of clustering partitions and picking the largest value, we hope to obtain the best clustering for curves based on that criterion. The predictive likelihood expression we will derive depends only on the number of curves in each cluster, the dimension of a vector of basis coefficients for the curve representations, and a measure of variability within each cluster.

This article is organized as follows. In Section 2, we review some general clustering literature and literature on clustering functional data. In Section 3, we derive the predictive likelihood for our functional data clustering model. We also propose a stochastic search method based on simulated annealing and explain how we achieve a suitable acceptance rate by iterating between a set of "random pick" steps and steps based on silhouette width. In Section 4, we simulate data following various functional data models and describe how our algorithm clusters such data. In Section 5, we use this improved algorithm to cluster real data sets such as vertical density profile data and data on yeast gene expression ratios.

2 Literature Review

Cluster analysis, a form of unsupervised learning, has a long history. Early types of clustering methods included dissimilarity-based hierarchical methods and partitioning methods. More recently, model-based clustering (e.g., Banfield and Raftery (1993)), entailing specifying an explicit statistical model for the data, has come to the forefront of clustering research. When the data set is functional in nature, a major approach to dealing with infinite-dimensional functional data is dimension reduction or functional versions of model-based clustering. Since distances cannot be precisely calculated without an explicit analytic representation of functions, dissimilarity-based approaches often fail as explained by Jacques and Preda (2014). Practically speaking, a popular method entails transforming the curves into a space of functions with finite dimensions. Then, clustering techniques for data with finite dimensions can be used, approximating the distance between the true functions (Jacques and Preda, 2014).

Numerous authors have proposed methods for the cluster analysis of functional data. These include early work such as Tarpey and Kinateder (2003), who connected principal points of functions (optimal approximations of the functions) to cluster means of a K-means clustering of basis coefficients, establishing the importance of basis representations in clustering functional data. In another early work, Rossi et al (2004) used a Self-Organizing Map, again on a basis expansion of the functions. Two

methods commonly used in practice, partially due to their ease of implementation in provided software, are the functional K-means approach (Febrero-Bande and de la Fuente, 2012) and the funFEM algorithm (Bouveyron et al, 2015), each of which we revisit later in our simulation study. More recent approaches include the the principal curve clustering of Wu et al (2022), in which the clustering is implemented on the principal scores from a functional principal components analysis of the data and the projection-based approach of Delaigle et al (2019), which employs a weighted version of the functional k-means method. Model-based functional data clustering approaches include that of Nguyen and Gelfand (2011), which assumes a Dirichlet process model for the functions, and Hébrail et al (2010) assumed a simple regression model for each function and used that as the foundation for the clustering. We will do something similar in employing regression for each curve, although we allow our regression model to be more flexible than in Hébrail et al (2010) and our method is more similar to the general model-based approaches of Chamroukhi and Nguyen (2019) and Chamroukhi (2016), although our objective function and optimization algorithm is different from theirs. Our method eschews an overtly dissimilarity-based approach such as the one described in Chen et al (2014) and is more reminiscent of model-based approaches like Chamroukhi and Nguyen (2019) in which the basis representation is central. Extensive reviews of functional data cluster analysis can be found in Zhang and Parnell (2023), Jacques and Preda (2014), and Hitchcock and Greenwood (2015).

3 Method

The subfield of statistics known as functional data analysis (FDA) (Ramsay and Silverman, 2002; Ferraty and Vieu, 2006) deals with data that are functions, i.e., intrinsically infinite dimensional. The data are inherently stochastic processes $\{y_i(t), i = 1, 2, ..., N\}$ in a Hilbert space. In practice, the response is measured densely on a grid of points $t_1, \ldots, t_n$ over some domain $[0, T]$. A general model for a functional observation can be represented as

$$y_i(t_m) = \sum_{b=1}^{B} c_{ib}\phi_b(t_{im}) + \epsilon_{ij} = \mu(t_{im}) + \epsilon_{im} \tag{1}$$

where $i = 1, \ldots, N$, $m = 1, \ldots, n$, and $\{\phi_b(t), b = 1, \ldots, B\}$ is a basis system to represent the signal function $\mu(t)$, $\epsilon_{im} = \epsilon_i(t_m)$ are measurement errors or noise, and c_{ib} are the coefficients of the basis functions. The data curves are represented via a basis expansion in the functional space, for example with a B-spline approximation or a P-spline approximation in a non-periodic functional observation. The choice of basis can be informed by prior knowledge before analysis; for example, a Fourier basis using sine and cosine functions for basis functions can be used to represent a periodic functional observation. A wavelet basis can be used to represent irregular, spiky functions. We will use the basis representation explicitly within the predictive likelihood.

3.1 Predictive Likelihood for Functional Data Clustering

The prediction of unobserved or missing data can be handled with a predictive likelihood (Butler, 1986). Bjørnstad (1990) suggested that the predictive likelihood can be viewed as a non-Bayesian analogue of the posterior predictive density for unobserved random quantities (not requiring the specification of a prior) and reviewed numerous formulations of the predictive likelihood in various settings. We propose a predictive likelihood objective function for the purpose of cluster analysis.

Suppose we have as our observed data N functional observations $(y_1(t), y_2(t), \ldots, y_N(t))$, each having random components following a Gaussian distribution at each measurement point along the curve. (In Section 4, we will investigate the sensitivity of our method to this assumption.) For object i where $i = 1, \ldots, N$, we can define an indicator vector $\mathbf{Z}_i = (Z_{i1}, \ldots, Z_{iN})'$ such that $Z_{ij} = 1$ if object i belongs to cluster j and 0 otherwise. Each partition in the space of clustering partitions corresponds to a different $\mathbf{Z} = (\mathbf{Z}_1, \ldots, \mathbf{Z}_N)$, and the particular choice of $\mathbf{Z}$ then determines the clustering structure. Note that if the number of clusters $K < N$, then $Z_{i,K+1} = \cdots = Z_{iN} = 0$ for all i, so to simplify the notation, henceforth we will have the index for the number of clusters stop at K, although we keep in mind that the value of K may vary between 1 and N within the broad family of models (and may vary within the optimization algorithm we will use).

One can frame the goal of cluster analysis as the prediction of the values of the $\mathbf{Z}_i$'s based on the data $\mathbf{Y}$ (i.e., the data curves $(y_1(t), \ldots, y_N(t))$. This is along the lines of the classification maximum likelihood approach (see, e.g., Everitt et al (2011)). We assume the $\mathbf{Z}_i$'s are i.i.d. unit multinomial vectors with unknown probability vector $\mathbf{p} = (p_1, \ldots, p_K)$. Also, we assume a model $f(\mathbf{Y}|\mathbf{Z})$ for the distribution of the data given the partition. The joint density

$$f(\mathbf{Y}, \mathbf{Z}) = \prod_{i=1}^{N} \prod_{j=1}^{K} [p_j f(y_i(t_1), \ldots, y_i(t_n))|z_{ij})]^{z_{ij}}$$

is a sensible choice to serve as an objective function, where $y_i(t_1), \ldots, y_i(t_n)$ is the discretized observation of the ith curve $y_i(t)$. Note that the parameters in $f(\mathbf{Y}|\mathbf{Z})$, and $\mathbf{p}$, are unknown. In model-based clustering, these parameters are estimated by finding the values of $\mathbf{Z}$ and the data model parameters that maximize the likelihood, given the observed $\mathbf{Y}$. If the data model $f(\mathbf{Y}|\mathbf{Z})$ allows it, a *predictive likelihood* for $\mathbf{Z}$ can be constructed by conditioning on the sufficient statistics $r(\mathbf{Y}, \mathbf{Z})$ of the parameters in $f(\mathbf{Y}, \mathbf{Z})$ (Butler, 1986; Bjørnstad, 1990). We choose to let this predictive likelihood, $f(\mathbf{y}, \mathbf{z})/f(r(\mathbf{y}, \mathbf{z}))$, serve as the objective function. Bjørnstad (1990) notes that, when $\mathbf{Y}$ or $\mathbf{Z}$ is continuous, an alternative predictive likelihood with a Jacobian factor for the transformation $r(\mathbf{Y}, \mathbf{Z})$ is possible. (Including the Jacobian factor makes the predictive likelihood independent of the choice of minimal sufficient statistic; excluding it makes the predictive likelihood invariant to scale changes of $\mathbf{Z}$.)

Consider a functional datum $y_i(t)$ and its corresponding observed discretized curve $\mathbf{Y}_i$. For our approach to cluster analysis, we will make the admittedly restrictive assumption that functional observations in the same cluster come from the same linear

smoothing model with a normal error structure. Therefore, the distribution of $\mathbf{Y}_i$, given that it belongs to cluster j, is

$$\mathbf{Y}_i \mid z_{ij} = 1 \sim N(\mathbf{T}\boldsymbol{\beta}_j, \sigma_j^2 \mathbf{I})$$

for $i = 1, \ldots, N$. Here $\mathbf{T}$ is the design matrix, whose n rows contain the independent variables in the regression model defined by the functional form we assume for $y_i(t)$. This model for $\mathbf{Y}_i$ allows for a variety of parametric and nonparametric linear basis smoothers, including standard linear regression, regression splines, Fourier series models, B-splines, wavelets, and smoothing splines, as long as the basis coefficients are estimated using least squares.

We must propose some basis for the curves, such as a linear, quadratic, cubic, spline, or Fourier basis (or a basis that combines multiple such components), having an associated design matrix denoted by $\mathbf{T}$. Then, $\hat{\boldsymbol{\beta}}_j$ is the least squares estimator of the coefficient vector of the regression for objects in cluster j (this is shown in Appendix A.1 to be the average of the least-squares estimates of the coefficient vectors from the individual regression fits for all the curves in cluster j).

Once we assume a model for the discretized functional data, we derive the predictive likelihood to use as an objective function for the clustering algorithm. The details of this derivation are given in Appendix A, but we outline the broad steps of it here:

1. We write $f(\mathbf{y}, \mathbf{z})$ based on the model assumption given above that the discretized curves $\mathbf{Y}_i$'s are normal, i.i.d. within the same cluster and with cluster-specific means and covariance matrices across different clusters.
2. We write $f(r(\mathbf{y}, \mathbf{z}))$ for the cluster-specific sufficient statistics $(m_j, \hat{\boldsymbol{\beta}}_j, \hat{\sigma}_j^2), j = 1, \ldots, K$, under the model assumptions that the m_j's are multinomial with parameters N and $p_1, \ldots, p_K$; $\hat{\boldsymbol{\beta}}_j$ is multivariate normal; and a properly scaled version of $\hat{\sigma}_j^2$ has a chi-square distribution.
3. We show that in the predictive likelihood expression $f(\mathbf{y}, \mathbf{z})/f(r(\mathbf{y}, \mathbf{z}))$, several terms in the numerator and denominator cancel and the resulting simplified expression serves as our objective function that we seek to maximize in the clustering algorithm.

Let $J = \{j | z_{ij} = 1 \text{ for some } i\}$ denote the set of the K nonempty clusters. The unknown parameters in $f(\mathbf{Y}, \mathbf{Z}), j \in J$, are $(p_j, \boldsymbol{\beta}_j, \sigma_j^2)$. The corresponding sufficient statistics are $(m_j, \hat{\boldsymbol{\beta}}_j, \hat{\sigma}_j^2)$. Given a value of $\mathbf{Z}$, which defines the clustering partition, the sufficient statistics $(m_j, \hat{\boldsymbol{\beta}}_j, \hat{\sigma}_j^2)$ can be obtained: $m_j = \sum_{i=1}^{N} z_{ij}$ is the number of objects in the (proposed) jth cluster, which is a direct consequence of $\mathbf{Z}$; $\hat{\boldsymbol{\beta}}_j$ is the p^*-dimensional least squares estimator of $\boldsymbol{\beta}_j$ (based on a regression of a vectorized version of *all* objects in the proposed jth cluster using the design matrix of the chosen basis) and thus also follows from the proposed cluster partition; and $\hat{\sigma}_j^2$, the unbiased estimator of σ_j^2, is the sum of squared errors of the regression of *all* objects in the proposed jth cluster, and thus also follows from the proposed cluster partition. As shown in Appendix A, the estimators $\hat{\boldsymbol{\beta}}_j$ and $\hat{\sigma}_j^2$ can be expressed in terms of the estimators $\hat{\boldsymbol{\beta}}_{ij}$ and $\hat{\sigma}_{ij}^2$ (for all i such that $z_{ij} = 1$), obtained from the regressions of the *individual* functional objects using the design matrix of the chosen basis, so that

the regression models can be fit initially on each individual curve, and need not be refit each time the structure of the clusters changes during the steps of the algorithm.

Then, as shown in Appendix A, the predictive likelihood for the unknown cluster indicator matrix $\mathbf{Z}$ can be written:

$$g(\mathbf{z}) = \frac{f(y,z)}{\prod_{j\in J} f(m_j, \hat{\beta}_j, \hat{\sigma}_j^2)}$$
$$\propto \prod_{j\in J} m_j!(m_j)^{-\frac{1}{2}}(\hat{\sigma}_j^2)^{-\frac{m_j n-p^*}{2}+1}\Gamma\left(\frac{m_j n-p^*}{2}\right)\left(\frac{m_j n-p^*}{2}\right)^{-\frac{m_j n-p^*}{2}}$$

The parts of this formula that vary across clustering solutions are the number of objects in the j-th cluster m_j and $\hat{\sigma}_j$, which is a measure of within-cluster variability. The value of m_j is a characteristic of the clustering partition proposed at the current step in the algorithm. We show in Appendix A.1 how to calculate $\hat{\sigma}_j$. Based on the formula in Appendix A.1, this $\hat{\sigma}_j$ can be interpreted intuitively as measuring both the spread of the discretized curves around their own smoothed representations and the inherent variation of discretized curves within a single cluster. In short, $\hat{\sigma}_j$ is a type of measure of the variability within cluster j. Since desirable clustering partitions have small within-cluster variances, small values of $\hat{\sigma}_j$ yield good clustering partitions, and the typically negative exponent of $\hat{\sigma}_j$ in $g(\mathbf{z})$ indicates that small within-cluster variances yield large values of $g(\mathbf{z})$, so that maximizing the predictive likelihood is associated with a good clustering. Clusters containing more curves (having large m_j values) will have a more substantially negative exponent and so the within-cluster variances of the larger clusters will more heavily drive this phenomenon than the within-cluster variances of smaller clusters. The other factors of $g(\mathbf{z})$ depend only on the m_j's. Given the dominance of the $m_j!$ piece, these extra factors can be viewed as penalty terms that penalize partitions that have many clusters with small m_j values, since multiplying a few large $m_j!$ values will yield a higher predictive likelihood than multiplying many smaller $m_j!$ values.

Note that the number of measurement points n on each discretized curve has a role in the predictive likelihood expression. The value of n is the same across clustering partitions, so its value will not affect the selection of the best partition. However, if the curves are measured very densely and n is extremely large, as is the case with some functional observations, the predictive likelihood could be unwieldy or practically impossible to compute. In such cases, we recommend an interpolation along the domain that would work with a smaller subset of the measurement points (the same subset across all N curves). For such an interpolation to make sense, the curves would need to be relatively smooth — at least not so jagged that interpolation would sacrifice important characteristics of the functional data. We also note that in practice, we work with the logarithm of the predictive likelihood to reduce the chance of numerical overflow when calculating the predictive likelihood.

3.2 Simulated Annealing

Simulated Annealing (SA) is a stochastic optimization algorithm proposed in Kirkpatrick et al (1983) and Černỳ (1984) for solving highly nonlinear problems. The procedure was first used in a physical annealing context by Metropolis et al (1953). It can effectively solve optimization problems in which many local optima exist. The algorithm produces a stochastic process of states (which in our context are clustering partitions of the functional data) and includes a temperature parameter T which is a tool to control the acceptance rate for newly proposed states.

In our algorithm, since our predictive likelihood serves as an objective function we wish to maximize, both the predictive likelihood values and the temperature affect the probability of accepting the proposed partition at each step of the simulated annealing process. A "cooling schedule" determines how fast the temperature decreases throughout the algorithm. While a logarithmic cooling schedule has theoretical advantages, as explained in (Nourani and Andresen, 1998), in many applications it can produce an impractically slow decrease. However, given the massive partition space in our clustering setup, we intentionally choose the doubly iterated logarithm in the denominator to yield a slowly decreasing temperature schedule that allows a wider exploration of the partition space. Setting the tuning parameter in the numerator of the temperature formula to 100 allows us to keep this acceptance rate between 20 to 50%, as suggested by, e.g., Robert and Casella (1999):

$$T_c = \frac{100}{\log(\log(1+c))} \tag{2}$$

where c is the iteration number of the algorithm. At each iteration, we accept the new partition with a probability determined by the temperature and the change between the new and current predictive likelihood value. The formula is as follows:

$$p_c = \min\left(\frac{\exp(value_{new} - value_{old})}{T_c}, 1\right) \tag{3}$$

where $value_{new}$ is the predictive likelihood value corresponding to the new partition and $value_{old}$ is the predictive likelihood value corresponding to the current partition. The form of this acceptance probability formula is taken from the classic Metropolis step introduced in Metropolis et al (1953).

In the stochastic search algorithm, we need a way to sensibly generate a new state (which, recall, is a clustering partition) at each step. We now present two techniques for generating a new partition from the current partition. We call these the "Silhouette Width Technique" and the "Random Pick Technique". Our algorithm will alternate between these techniques across the iterations.

3.3 Silhouette Width Technique

One of the most common strategies for assessing cluster validity is the silhouette width. The concepts of the silhouette width for an object in a clustering solution, and the average silhouette width among all objects, were introduced by Rousseeuw

(1987). The silhouette width shows how well each object fits into its respective cluster. A way to measure goodness of a clustering partition is called the average silhouette width, which was first proposed by Rousseeuw (1987). The average silhouette width measures cluster quality and ranges between -1 and 1, where an average silhouette width of 1 means all objects are properly matched, while an average silhouette width of -1 means all objects are not matched correctly (Rousseeuw, 1987). A high positive value of the average silhouette width indicates the clustering partition is a good result. The silhouette width is defined for each individual object, while the average silhouette width (averaged across all objects in the data set) is a measure of the goodness of the overall clustering partition. Let $d(i, j)$ represent the distance between the i-th and j-th objects in a data set. For our purposes, the distance measure for the curves is defined as Euclidean distance (L_2 distance). The formula for the L_2 distance between curve i and curve j is as follows:

$$d_{L_2}(i,j) = \sqrt{\int [y_i(t) - y_j(t)]^2 dt} \tag{4}$$

which can be approximated readily for observed discretized functions. The formula for the silhouette width of the j-th object (which is assigned to, say, the J-th cluster, denoted C_J, which contains $|C_J|$ objects) is

$$SW_j = \frac{b(j) - a(j)}{\max\{a(j), b(j)\}} \tag{5}$$

where

$$a(j) = \frac{1}{|C_J| - 1} \sum_{j^* \in C_J, j^* \neq j} d(j, j^*) \tag{6}$$

and

$$b(j) = \min_{J' \neq J} \frac{1}{|C_{J'}|} \sum_{j' \in C_{J'}} d(j, j'). \tag{7}$$

Thus $a(j)$ is the average distance of object j to other objects $j^* \in C_J$ in its own cluster which is denoted by C_J. And $b(j)$ is the average distance of object j to objects $j' \in C_{J'}$ in the nearest cluster to that object, which is denoted by $C_{J'}$. Intuitively, $a(j)$ measures how close the j-th object is to the other objects in its own cluster. And $b(j)$ measures how close the j-th object is to the objects in the cluster nearest to the j-th object's own cluster. If $b(j)$ is much larger than $a(j)$, then the silhouette width of the j-th object is highly positive and this is evidence that the j-th object has been placed in the correct cluster. The average silhouette width is necessarily between -1 and 1, and the value of the silhouette widths averaged within some generic cluster C (where N_C is the number of objects in cluster C) is

$$\overline{SW}(C) = \frac{1}{N_C} \sum_{j=1}^{N_C} SW_j \tag{8}$$

where the average is taken over all the N_C objects in cluster C, $j = 1, \ldots, N_C$. Then the overall average silhouette width for the clustering partition is the average of $\overline{SW}(C)$ values over the K clusters $C = 1, \ldots, K$. If the data are homogeneous and well separated, a larger average silhouette width value means a better quality of clustering. Compared with other measures of cluster accuracy such as the F-measure, the adjusted Rand Index, connectivity, the Dunn Index and a stability-based method, the average silhouette width has a good performance when the number of clusters is smaller than or equal to 5 (Handl et al, 2005) and is typically considered a reliable measure of the quality of a clustering partition.

In our procedure, at each step of the algorithm we calculate the silhouette width corresponding to each curve. When an object's silhouette width is low, it is desirable to move the curve to another cluster with high probability. Therefore, we base the decision of whether to move a curve to a different cluster on its silhouette width:

1. Compute the silhouette width SW_j for curve j, as defined above, within its particular current cluster.
2. Compute probability $p_s = \max\left\{ \dfrac{SW_j - minSW}{maxSW - minSW}, \dfrac{SW_j}{maxSW} \right\}$.
3. Generate a uniform number u_s from Unif[0,1].
   - If $u_s \leq p_s$, then curve j stays in its current cluster,
   - else curve j changes to its nearest cluster.
4. Return to step 1.

where $minSW$ is the lowest value over every silhouette width value in each cluster, and $maxSW$ is the largest value over every silhouette width value in each cluster. The proportion $(SW_j)/(maxSW)$ is high when SW_j is close to the maximum silhouette width, meaning the j-th curve is nearly optimally clustered already, and so $(SW_j)/(maxSW)$ produces a sensible probability of a curve staying in its cluster. The role of $(SW_j - minSW)/(maxSW - minSW)$ in p_s is for the special case when $SW_j < 0$, to avoid a negative probability.

3.4 Random Pick Technique

Recall that clustering is a process to separate a set of data into multiple homogeneous groups (Jacques and Preda, 2014). During a partitioning clustering algorithm, observations are switched from one cluster to another until some equilibrium is reached. Often, a clustering algorithm is determined in large part by the mechanism for observation switching. In the previous subsection, we have presented the silhouette width method for potentially moving objects from one cluster to another. We now describe another technique to transfer curves from one cluster to another cluster, which we call the "random pick" technique. The reason for using this random pick approach is to obtain an irreducible Markov chain as the chain of our partitions over the steps of the algorithm. As we describe below, every partition in the state space has a nonzero probability of being proposed at some point, so the chain has a chance to reach every state. Eventually, our algorithm will alternate between the silhouette width method and the random pick method during its steps.

During the process, we assign a probability weight to each curve under consideration for switching. The cluster of each curve may be randomly changed according to that curve's specified probability. This generates a new cluster partition for the data set. It allows the Markov chain to be ergodic, meaning that every possible partition has a non-zero chance of appearing in the chain. In the actual classification, we prefer to control the number of clusters generated or eliminated, so we create a new parameter δ. Also, we create another parameter α to control the probability of a curve staying in its current cluster or changing to another cluster.

In our algorithm, K is the current number of clusters, $K \in \{1, 2, 3, ...\}$. We set parameters $\delta = 0.4^{\log(N)}$ and $\alpha = (K-1)/[K(K-1)-1]$ for $k \geq 3$ (and $\alpha = 0.5$ for $K \leq 2$), so that $\alpha \in (0, 0.5]$ and N is the number of curves in the entire functional data set (across all clusters). While these choices seem heuristic, they were carefully chosen after extensive experimentation to produce a robust rule for adding clusters and moving objects to different clusters, with the goal of giving high probability to keeping the number of clusters stable and keeping a curve in its present cluster, but still allowing a nonzero chance for a curve to switch clusters or to be placed into a newly created cluster. We use δ to control the rate of generating new clusters and α to control the probability of a curve changing clusters in this algorithm. For each observation: We keep the current number of clusters and keep the curve in the same cluster with probability $1 - \delta$. We move the observation from the current cluster to a different existing cluster with probability $(K-1)\alpha\delta$, i.e. with probability $\alpha\delta$ of moving to any one of the other $K-1$ clusters besides the cluster it is currently in. We add a new cluster and place the curve in this new cluster with probability $(1-(K-1)\alpha)\delta$. For example, if we have $N = 20$ or $N = 40$ curves to cluster, Table 1 displays the probabilities associated with our random pick technique for various values of α, where α depends on the current number of clusters k. Table 1 illustrates how the α value and the probability of creating a new cluster decreases as the number of clusters increases, thereby favoring a solution with a smaller number of clusters. The number of clusters depends on δ, which is a function of the number of curves in the data set, and α, a function of the current number of clusters. But the eventual number of clusters in the final solution is guided by the data, since the partitions are accepted or not based on the predictive likelihood value. Furthermore, since the primary goal of using the "random pick" method is to produce an ergodic chain of proposed partitions, the overall clustering solution will be quite insensitive to small changes in the formulas for λ or α, as long as these choices yield stable adjustments to the proposed clustering partitions.

3.5 Choosing an appropriate basis

In functional data analysis, a basis (i.e., a collection of basis functions) is a mathematical technique to represent functional data with an approximate mathematical form. The choice of basis is a crucial decision in various applications. The selection of appropriate basis functions depends on the nature of the data. Some common basis functions that are widely used include: Polynomial basis functions (simple and effective for capturing regular patterns in curves); Fourier basis functions (to represent periodic

Table 1 Tuning parameter example for a functional data set with $N = 20$ or $N = 40$ curves

$N = 20$				
K clusters	α	$1-\delta$	$(K-1)\alpha\delta$	$(1-(K-1)\alpha)\delta$
3	0.400	0.936	0.051	0.013
4	0.273	0.936	0.052	0.012
5	0.211	0.936	0.054	0.010
6	0.172	0.936	0.055	0.009
$N = 40$				
3	0.400	0.966	0.027	0.007
4	0.273	0.966	0.028	0.006
5	0.211	0.966	0.029	0.005
6	0.172	0.966	0.029	0.005

phenomena); B-spline basis functions (for representing smooth curves). Researchers and analysts should exercise caution when selecting the appropriate basis functions for their datasets. For example, when one chooses a Fourier basis to represent the data, the functional data must exhibit periodic behavior. In our simulation study, we illustrate the impact of different choices of basis function.

3.6 Our Clustering Algorithm

In this section, we explain in detail how we apply our new method to a set of functional data to be clustered. We begin by choosing a basis to represent the functional data and estimating the basis coefficients via least squares, separately for each curve. We select the model for the basis representation of the curves based on the shape of the curves. For example, we can choose a Fourier series basis if the data is periodic. The specific approach we take is to consider a number of potential bases; for example, in our real data analysis in Section 5, we consider a Fourier basis, a quadratic regression basis, and a basis that is a mix of different types of basis functions.

The initial partition in the algorithm must be chosen in our algorithm, and we may specify an arbitrary initial set of cluster labels, although performance is improved when the initial cluster labels are well chosen, for example by doing a quick naïve clustering method on the data. As a reasonable approach, a commonly-cited rule of thumb (Boehmke and Greenwell, 2019) for small to moderate data sets is letting the initial number of clusters $K \approx \sqrt{N/2}$, where N is the number of functional objects in the whole data set. In our algorithm, the number of clusters will be dynamically adjusted in a data-driven way to find the optimal number of clusters in the functional data. This is because in both of the partition-generation approaches we use, the number of clusters is not necessarily fixed. In the "silhouette width" approach described in Section 3.3, it is possible for all the objects in a cluster to move to another existing cluster, thereby reducing the overall number of clusters in the partition. In the "random pick" approach described in Section 3.4, it is possible both that an additional new cluster could be created in a move, or all the objects in an existing cluster could exit that cluster, so that the total number of clusters could either increase or decrease in a step. Of course, any newly proposed partition is more likely to be accepted in the algorithm if it produces a fairly high predictive likelihood.

From the starting partition, we seek to improve the predictive likelihood to reach the best or nearly best partition. Following the approach outlined in Sections 3.3 and 3.4, at each step we generate a new partition for the data using the random pick technique or silhouette width method. Experimentation revealed that a good strategy is to alternate between the two different methods of generating a partition, using one of them M times in succession, and then switching to the other one for M steps, and so forth. We experimented with various values of M and found $M = 3$ works well, and a sensitivity analysis described in Section 4.1 confirms that the results were not especially sensitive to the choice of M. Once we have generated a new partition, we compare the predictive likelihood values of the new partition and the current partition; if the value of the new partition is higher than the current one, then we replace the current partition; if the value of new partition is lower, then we derive an acceptance probability to replace the current partition using the formula in Equation 3. The algorithm is described in Algorithm 1:

Algorithm 1 Predictive likelihood clustering algorithm

Require: Functional data, each observed at n discrete measurement points
Ensure: Final clustering partition

1: Randomly assign the curves into K clusters to obtain an initial partition.
2: Compute the initial predictive likelihood value $value_{old}$.
3: Choose the basis function model according to the characteristics of the curves, construct the design matrix $\mathbf{T}$, and estimate the model parameters by least squares.
4: **repeat**
5: Generate a new partition of K clusters using either the random pick or silhouette width techniques (doing each for $M = 3$ iterations in a row)
6: Compute the predictive likelihood value $value_{new}$ for the new partition.
7: Compute the temperature parameter $T_c = \frac{100}{\log(\log(1+c))}$ at the c-th iteration.
8: Compute the acceptance probability

$$p_c = \min\left\{\exp\left(\frac{value_{new} - value_{old}}{T_c}\right), 1\right\}.$$

9: Generate $u_c \sim \text{Unif}(0, 1)$.
10: **if** $u_c \leq p_c$ **then**
11: Accept the new partition and set $value_{old} \leftarrow value_{new}$.
12: If the new partition has added a new cluster or if a previous cluster has been made empty, increment or decrement the value of K accordingly.
13: **else**
14: Retain the current partition
15: **end if**
16: **until** overall maximum predictive likelihood value found has not changed for S steps (we recommend using $S = 3000$)
17: Among partitions considered, select the partition that yields the highest predictive likelihood value.

To improve the efficiency of clustering, we break our algorithm into stages consisting of 50 iterations each. After a given stage, one approach is to use the best (in terms of predictive likelihood value) partition across these 50 iterations as the initial input for the next stage. We slightly alter this for flexibility: We randomly select, as the initial input for the next stage, one of the top 20 (out of 50) partitions from the stage, with the probability of selection being proportional to the relative magnitude of each partition's predictive likelihood value. This balances computational time with good quality of clustering results. While the number of the iterations per stage can be arbitrary, we have found that our clustering accuracy and computation time is best balanced by using 50 iterations in each stage. This stagewise approach is often used in optimization, machine learning, and simulation problems to find the best solution in a complex search space.

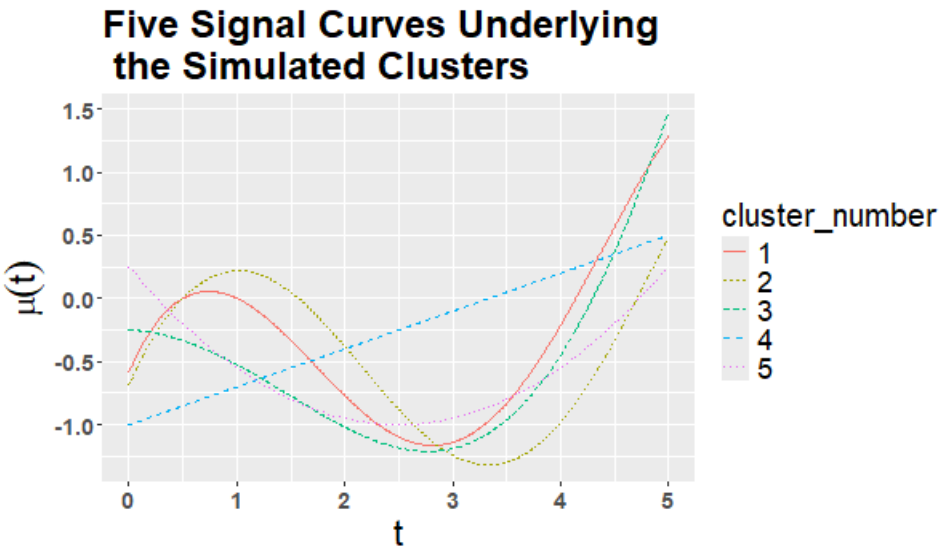

Fig. 1 The five signal curves, representing five distinct clusters of the simulated data.

4 Simulation Study

In this section, we describe an experiment on simulated curves to demonstrate the efficiency and accuracy of our algorithm in clustering curves. In each set of simulated curves, we generate N curves coming from several distinct groups with signal curves $(\mu_1(t), \mu_2(t), \mu_3(t), \mu_4(t)\ , \mu_5(t))$, so that each group has an equal number of curves. The groups of simulated curves have these signal functions (plotted in Figure 1):

$$
\begin{aligned}
\mu_1(t) &= -\sin(t-1)\ln(t+0.5) \\
\mu_2(t) &= \cos(t)\ln(t+0.5) \\
\mu_3(t) &= -0.25 - 0.1\cos(0.5(t-1))t^{1.5}\sqrt{5(\sqrt{t}+0.5)} \\
\mu_4(t) &= -1 + 0.3t \\
\mu_5(t) &= 0.2(t-2.5)^2 - 1
\end{aligned}
$$

In the discretized form of the functional observations, our generated values of t form an equally spaced grid of points where $t_1, t_2, \ldots, t_n \in [0, 5]$, $n = 50$. When simulating the observed curves, in order to obtain different curves within each group, we create a simulated functional datum by adding noise following a Ornstein-Uhlenbeck process to the signal function. The Ornstein-Uhlenbeck process is a common mean-reverting

process to model continuous noise. The stochastic differential equation of the Ornstein-Uhlenbeck process $\{X_t\}$ is:

$$dX_t = -\theta X_t dt + \sigma dW_t$$

in which $\theta > 0$ is called the drift parameter, which measures the speed of the regression to the mean, σ is the variability, and W_t is Brownian motion. A positive drift parameter will cause the process to drift back towards the signal component. Each simulated data set consists of several clusters each containing an equal number of curves as described below, where the each observed curve is composed of one of the signal curves plus random Ornstein-Uhlenbeck noise; we set $\theta = 1$ and vary σ for the simulated data. Note that our functional regression model underlying the predictive likelihood actually assumed an independent noise structure, while the Ornstein-Uhlenbeck process yields a dependent noise structure, which is likely more realistic for actual functional data. Because the dependent noise process is most realistic, we carry out our main simulation study with Ornstein-Uhlenbeck noise, but in Section 4.1, we employ several other independent noise settings to examine the sensitivity of the clustering results to the noise structure.

To assess the performance of our method, we consider 18 simulation settings, and for each setting, we simulate 100 data sets and perform cluster analysis on each data set. Across the 18 settings, we allow the number of clusters, the number of curves per cluster, and the size of the noise variance (measured by the Ornstein-Uhlenbeck parameter σ) to vary. Our six basic settings are: I: $N = 20$ curves, $K = 4$ clusters; II: $N = 40$ curves, $K = 4$ clusters; III: $N = 20$ curves, $K = 2$ clusters; II: $N = 40$ curves, $K = 2$ clusters; V: $N = 20$ curves, $K = 5$ clusters; VI: $N = 40$ curves, $K = 5$ clusters. For each basic setting, we consider $\sigma = 0.5$, $\sigma = 0.8$, and $\sigma = 1.2$, making 18 settings overall. The larger values of σ correspond to less separated clusters. Note that for settings with $K = 5$ clusters, all five signal curves in Figure 1 were used to define the clusters. For settings with $K = 4$ clusters, curves $\mu_1(t)$, $\mu_2(t)$, $\mu_3(t)$, and $\mu_4(t)$ were used. For settings with with $K = 2$ clusters, curves $\mu_1(t)$ and $\mu_3(t)$ were used.

Before clustering, we must decide which basis function representation to use for our functional data. For the simulated curves, we considered four different choices for basis functions: a cubic basis, a Fourier basis including two sine terms and two cosine terms, a mixed basis combining cubic basis functions and Fourier basis functions, and a cubic B-spline basis with 5 distinct knots. To investigate the effect of the choice of basis on the results, we tried each basis under consideration for the setting with $N = 20$, $K = 4$, and $\sigma = 0.5$. For each of the bases, we generated 100 sets of simulated data, estimated the basis coefficients, used our clustering method to find a clustering partition, and calculated the mean and Monte Carlo standard error (MCSE) of the Rand index (Rand, 1971) across 100 simulated data sets (see Table 2). From the result, using the B-spline basis produced the highest mean Rand index compared to other basis functions, and the mixed basis also performed well. It suggests the B-spline basis functions and mixed basis functions are well suited for capturing the underlying structure of these simulated data. In summary, the choice of basis can significantly impact clustering results, although Table 2 shows several choices of basis perform nearly equally well in this example. Our predictive likelihood clustering method may

Table 2 Mean Rand index values (and MCSE) across 100 simulated data sets for several choices of basis.

	Cubic basis	Fourier basis	Mixed basis	B spline
Mean	0.8528	0.9217	0.9301	0.9368
MCSE	0.0037	0.0022	0.0037	0.0047

perform differently with different bases, and we choose the cubic B-spline basis for our simulation study. Therefore, it is important to consider the structure of the data and how the characteristics of each basis mirror that structure when selecting an appropriate basis for the specific clustering task. Note that with a real data set, we do not have knowledge of a gold-standard clustering partition, so we propose in Section 5 an approach for choosing the basis before clustering a real data set.

We define the true clustering structure as corresponding to the groups of signal curves that the simulated curves are derived from. Once our final clustering partition is computed for each data set, we compare the achieved partition with the "true" clustering partition, via the Rand index (Rand, 1971), a measure of the similarity between two data clusterings, to get an accuracy rate. The value of the Rand index ranges from 0 to 1, which 0 indicates no similarity between the clusterings and 1 indicates identical clusterings. Therefore, if one of the clusterings being compared is the gold standard ("true" clustering), then the Rand index serves as the proportion of pairs of curves that are "correctly matched" (either together or apart) by the other clustering (the one yielded by the algorithm).

We clustered the simulated functional data with three competing algorithms: our "predictive likelihood" method; the functional K-means algorithm, implemented with the `kmeans.fd` function in the `fda.usc` package (Febrero-Bande and de la Fuente, 2012) in R (R Core Team, 2024); and the funFEM algorithm (Bouveyron et al, 2015), implemented with the `funFEM` function in the `funFEM` package in R, with model specification `model="AkjBk"` and initialization `init="kmeans"`. The accuracy rate for each clustering method was measured by the average Rand index across the 100 simulated data sets. The comparative results (mean Rand index values and Monte Carlo standard errors (MCSE)) are presented in Table 3. Our predictive likelihood method consistently has higher mean Rand index values than the approaches using the `kmeans.fd` and `funFEM` functions. All methods naturally worsen in performance when the within-curve noise rises to $\sigma = 1.2$, implying less separation among the clusters, but our method holds up fairly well. For all methods, Rand index values tend to be lower in simulated data sets with more curves, but the decline in performance as the number of curves increases is modest for our method. The functional k-means method struggles notably when there are $K = 2$ clusters, probably because that method is driven primarily by distance between curves rather than shapes of curves, and the two signal curves $\mu_1(t)$ and $\mu_3(t)$ are fairly close in distance. Our method, which smooths the observed curves and clusters based on the basis representations, holds up better in this more difficult clustering setting.

Table 3 Mean Rand index values (and MCSE) for our method and two competing methods for several simulation settings, varying the total number of curves, the number of clusters, and the separation between clusters. Settings: I: $N = 20$ curves, $K = 4$ clusters; II: $N = 40$ curves, $K = 4$ clusters; III: $N = 20$ curves, $K = 2$ clusters; IV: $N = 40$ curves, $K = 2$ clusters; V: $N = 20$ curves, $K = 5$ clusters; VI: $N = 40$ curves, $K = 5$ clusters.

		$\sigma = 0.5$	$\sigma = 0.8$	$\sigma = 1.2$
Setting	Method	Mean (MCSE)	Mean (MCSE)	Mean (MCSE)
I	Predictive Likelihood	0.9368 (0.0047)	0.9009 (0.0032)	0.8384 (0.0022)
	`kmeans.fd`	0.8301 (0.0052)	0.8171 (0.0058)	0.8031 (0.0054)
	`funFEM`	0.7326 (0.0016)	0.7392 (0.0026)	0.7044 (0.0045)
II	Predictive Likelihood	0.8656 (0.0025)	0.8334 (0.0023)	0.7816 (0.0021)
	`kmeans.fd`	0.7577 (0.0035)	0.7527 (0.0031)	0.7539 (0.0042)
	`funFEM`	0.6810 (0.0012)	0.6755 (0.0016)	0.6582 (0.0031)
III	Predictive Likelihood	1 (0)	0.9952 (0.0020)	0.8704 (0.0066)
	`kmeans.fd`	0.4774 (0.0004)	0.4783 (0.0005)	0.4832 (0.0011)
	`funFEM`	0.5994 (0.0021)	0.5923 (0.0037)	0.5818 (0.0058)
IV	Predictive Likelihood	0.9990 (0.0005)	0.9183 (0.0041)	0.7430 (0.0046)
	`kmeans.fd`	0.4885 (0.0001)	0.4887 (0.0002)	0.4889 (0.0003)
	`funFEM`	0.6130 (0.0015)	0.6077 (0.0028)	0.5915 (0.0044)
V	Predictive Likelihood	0.8764 (0.0019)	0.8545 (0.0019)	0.8198 (0.0018)
	`kmeans.fd`	0.7774 (0.0062)	0.7724 (0.0062)	0.7678 (0.0056)
	`funFEM`	0.8316 (0.0053)	0.8104 (0.0038)	0.7656 (0.0028)
VI	Predictive Likelihood	0.8277 (0.0014)	0.8090 (0.0016)	0.7673 (0.0014)
	`kmeans.fd`	0.7705 (0.0043)	0.7711 (0.0040)	0.7505 (0.0047)
	`funFEM`	0.8073 (0.0043)	0.7847 (0.0036)	0.7432 (0.0028)

Table 4 Mean Rand index values (with MCSE) for our predictive likelihood clustering method for $M = 2, 3$, and 7 (for one particular simulation setting).

	$M = 2$	$M = 3$	$M = 7$
Mean	0.9365	0.9368	0.9245
MCSE	0.0043	0.0047	0.0040

4.1 Some Sensitivity Analyses and Discussion of Performance

In this section, we investigate the sensitivity of our results to the choices of several tuning parameters in our algorithm and examine subtle questions related to the algorithm's performance.

Recall that in our algorithm, new partitions are generated by alternating between two stochastic updating methods (the"silhouette-width" and "random pick" updates). These two methods are switched every M iterations. Table 4, which varies M for the simulation setting with $N = 20$ curves, $K = 4$ clusters, and Ornstein-Uhlenbeck noise with $\sigma = 0.5$, shows that the mean Rand index results are not significantly different for $M = 2, 3$, and 7, and our recommended $M = 3$ appears to be a fine choice of M.

While our predictive likelihood approach technically assumes an independent normal noise structure, in Section 4 our simulated data had time-dependent noise, which more realistically represents real functional data. To assess the sensitivity of the proposed method to different error structures (both in terms of independence

Table 5 Mean Rand index values (with MCSE) for our method and two competing methods for independent noise structures with normal, heavy-tailed (t), and light-tailed (Uniform) noise.

	Normal	t	Uniform
Predictive Likelihood	0.8662 (0.0023)	0.8679 (0.0025)	0.8524 (0.0021)
`kmeans.fd`	0.8125 (0.0066)	0.8129 (0.0052)	0.8047 (0.0061)
`funFEM`	0.7176 (0.0029)	0.7231 (0.0033)	0.7211 (0.0036)

and distribution), we conducted additional simulation studies by generating three independent error distributions: Normal, t-distributed (heavy-tailed), and Uniform (light-tailed). The parameters of each distribution were chosen so that the error variance is comparable across settings. The simulated noise followed these respective distributions:

- $\epsilon_{ij} \sim N(0, \sigma^2 = 0.125)$
- $\epsilon_{ij} = W/\sqrt{10}$, where $W \sim t(df = 10)$
- $\epsilon_{ij} \sim \text{Unif}(-0.6, 0.6)$

All these settings yield an error variance of 0.125, which matches the variance $\sigma^2/2(1)$ of the Ornstein-Uhlenbeck noise structure with parameter $\sigma = 0.5$. In these extra simulation studies, we kept the B-spline basis representation and all components of the simulation design the same as in simulation runs in Section 4 with $K = 4$ clusters and 5 curves per cluster, except for altering the error structure. For each error distribution, we generated 100 simulated datasets and evaluated clustering performance using the Rand index. For all these noise settings, we also clustering the simulated data with the functional K-means algorithm (`kmeans.fd` function in the `fda.usc` package) and the funFEM algorithm (`funFEM` function in the `funFEM` package).

As seen in Table 5, our predictive likelihood method is quite robust to changes in the error structure, obtaining only slightly lower Rand index values as it did with the Ornstein-Uhlenbeck errors in Table 3. It is unsurprising that the Rand index values are slightly higher for all methods with the dependent noise structure, since the positively autocorrelated noise tends to produce smoother observed curves (whose structure is more easily captured with basis representations) than independent noise does.

An intriguing and fundamental question is whether a higher predictive likelihood value necessarily corresponds to a better partition. And can predictive likelihood values meaningfully compare two clustering partitions having different numbers of clusters? To investigate this, we simulated 100 functional data sets (that truly came from 4 clusters) and used our algorithm to cluster each simulated data set, allowing the number of clusters to vary as usual. The Rand index is, in a sense, a measure of the goodness of the choice of partition. We kept track of *both* the best Rand index value and the maximum predictive likelihood value for several choices of k during the algorithm ($k = 3$, 4, and 5 in this example). On average, a larger predictive likelihood value (across values of k) usually corresponds to a better partition (as measured by the Rand index), but it is not a guarantee: In 84 of the 100 simulated data sets, the choice of k with the higher predictive likelihood value corresponded to the choice of k with the highest Rand index. In the 16 cases where these did not coincide, it was typically the choice of $k = 3$ that had a *slightly* higher predictive likelihood value than

the "correct" choice of $k = 4$. This is likely due to the penalty terms in the predictive likelihood rewarding a smaller number of clusters having more objects, so that the method overall, when it errs, tends to err on the side of a more parsimonious clustering structure.

5 Real Data Analysis

In this section, we apply our algorithm to two real data sets, one containing density values for a sample of manufactured boards and the other containing expression ratios for a sample of yeast genes.

5.1 Vertical Density Profile Data

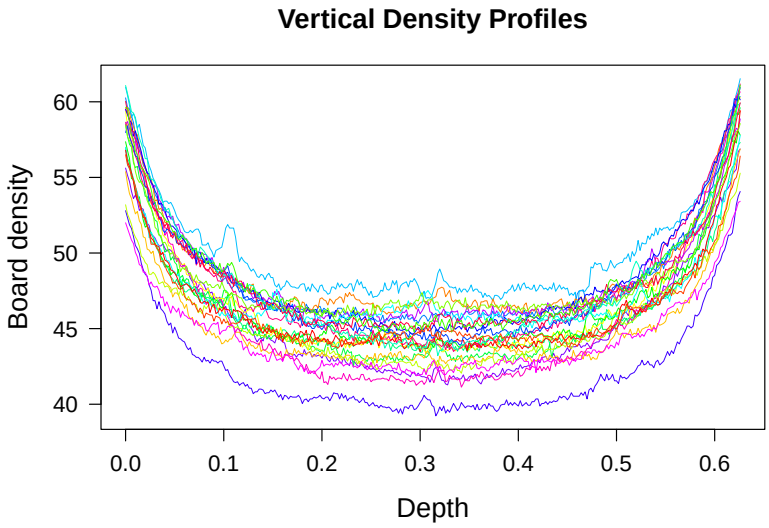

Fig. 2 Observed Vertical Density Profiles for 24 boards, separated by color.

Properties of engineered wood boards are studied by Walker and Wright (2002). In particular, the measurements of interest are a series of density values taken across the thickness of the board, which can in combination form a functional datum. This function, known as the vertical density profile (VDP) of the board, is a measure of the density of thickness of a sample board, measured by a profilometer (Walker and Wright, 2002). The entire set of functional data that we study in this example contains 24 such profiles, each with 314 measurement points. We will use our algorithm to group similar profiles based on these measurements, which can be treated as approximately continuous functions. Based on a visual inspection of the profile curves, their basic trend shapes are U-shaped, concave-up curves. We initially considered a quadratic

Table 6 Average AIC values across the fitted curves for the VDP profiles, for three choices of basis.

Basis	Fourier basis	Quadratic basis	Mixed quadratic and Fourier basis
Average AIC	590.929	974.801	228.114

Table 7 The list of cluster memberships for the 24 profiles in the VDP data, based on the predictive likelihood clustering solution.

Cluster	Profile Numbers in VDP Data
1	1,2,3,4,5,8,14,15,20
2	6,16
3	9,11,17,18,24
4	7,12,19
5	10
6	13
7	21

basis for the design matrix because there are no periodic changes in the functions, and the plots of the profile curves display parabolic shapes.

While the initial appearance of the curves suggested a quadratic basis, we chose the appropriate basis in a more objective way. We considered several candidates for basis, used each set of basis functions to approximate a fit to each curve in the data set, calculated the AIC of each regression fit, and then, for each possible choice of basis considered, we averaged the AIC values across the curves. We computed the average AIC values for the following choices of basis: a Fourier basis including two sine terms and two cosine terms, a quadratic basis, and a mixed basis that included two sine terms, two cosine terms, and a set of polynomial terms up to the quadratic. A lower average AIC value indicates a more favorable balance between model fit and complexity. From the average AIC values in Table 6, it is evident that the mixed basis that combines the Fourier terms and the quadratic basis is more appropriate for the VDP data.

We then carried out our clustering algorithm using the predictive likelihood as our objective function. The clustering result shown in Table 7 reveals the formation of seven clusters among the 24 curves. This suggests a meaningful segmentation of the data, with each cluster likely representing a unique pattern or behavior within the dataset. The identification of such clusters can provide valuable insights into the underlying structure of the data, facilitating a more nuanced understanding of the relationships between the curves.

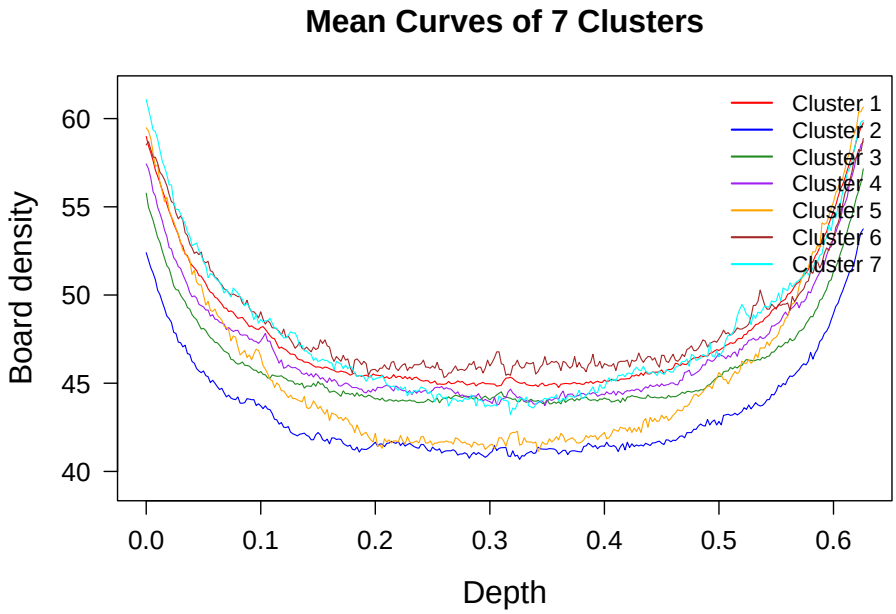

Fig. 3 Pointwise mean curves, separate for each of the seven clusters in the clustering solution for the analysis of the vertical profile density data.

Figure 3 displays the mean curves for each of the seven clusters in the clustering of the VDP data. The distinct patterns observed in the mean curves indicates that our clustering algorithm's solution has identified different thickness characteristics of the boards and separated them by these characteristics. From these seven clusters, some clusters exhibit greater density at all levels of depth and others clusters show less density. For example, Cluster 2 (shown in blue color) has the lowest density over the entire depth domain. Cluster 6 (shown in brown) has a relatively large density over the entire domain. Also, we can see some mean curves exhibit varying relative magnitudes of density across the domain. For example, the mean curve of Cluster 5 (gold) has one of the lowest density level in the middle values of depth, but its mean curve is quite high at the low and high regions on the depth domain. The mean curve of Cluster 1 (orange) shows a similar pattern to Cluster 5 at the extremes of the domain, but is higher in the middle of the domain than that of Cluster 5. These different density patterns could, for example, help manufacturers to separate the boards into meaningful groups, possibly allowing the different groups to have different price points when selling the boards. Finally, it is of critical interest that three of the seven clusters are formed of singleton objects which be thought of as outliers that have their own distinct density patterns: The aforementioned Cluster 5 and the high-magnitude Cluster 6 are each singleton curves. Also, the low-magnitude Cluster 2 consists merely of a pair of curves. On the other hand, Clusters 1 and 3, whose magnitudes tend to be in the medium range, have a larger number of curves in them.

5.2 Yeast Genes

The data we analyze in this section is a well-known set of yeast gene data, originally studied by (Eisen et al, 1998) and (Alter et al, 2000). The expression ratio is the response variable and is calculated as the ratio of gene expression levels in one condition relative to another. We used a log transformation on the yeast gene data, as is commonly done with expression ratios. In the data set studied here, which is a subset of the original data of (Eisen et al, 1998), we have measurements of expression ratios for 78 yeast genes at 18 different time points, with each time point being 7 minutes apart. Thus we treat the data as 78 functions, whose clustering creates groups for the 78 genes.

For these 78 yeast gene curves, biologists believe that there is a classification of genes based on their functions during different phases of the cell cycle (G1, S, G2 and M). Genes (1-13) in this group are believed to be active during both the gap phase 1 (G1) and mitosis (M) of the cell cycle. Genes (14-52) are thought to be specific to the G1 phase which is the gap phase that occurs before DNA synthesis. Gene (53-60) are related to the synthesis (S) phase during which DNA replication takes place. Genes (61-67) in this group are associated with both the synthesis (S) phase and the gap phase 2 (G2) of the cell cycle. Genes (68-78) are believed to belong the gap phase 2 (G2) and mitosis (M) of the cell cycle.

We selected the basis that provides the best fit to the yeast data in a similar way as described in the previous section. We calculated average AIC values across fits to the 78 genes, considering several candidate choices of basis: a Fourier basis with two sine terms and two cosine terms, a cubic basis, and a mixed basis that combined the Fourier terms and the polynomial terms up to the cubic. A lower average AIC value suggests a better trade-off between model fit and complexity. From Table 8, we see that the best model is the mixed basis model which includes polynomial terms and periodic terms.

Table 8 Average AIC values across the fitted curves for the yeast gene data, for three choices of basis.

Basis	Fourier basis	Cubic basis	Mixed basis
Average AIC	12.539	22.417	2.453

Our clustering algorithm stopped at 5 clusters after 30000 iterations. The highest predictive likelihood value achieved was 680.55, corresponding to the optimal group identified by our algorithm. The details of the best group at this peak predictive likelihood are presented in the following Table 9, which gives the confusion matrix that compares the achieved clustering partition (represented by the rows of the table) to the groups of the genes based on the biologists' beliefs (represented by the columns of the table).

From the table, we see that the clustering partition is for the most part accurate. Among the 78 genes, 40 are classified in the cell cycle phase hypothesized by the biologists. In addition, for the yeast gene data, genes in adjacent phases should behave similarly due to the circular nature of the phases. So a misclassification into an adjacent

Table 9 Confusion matrix giving "correct" classifications and misclassifications of the 78 observations in the yeast gene data set.

Cluster	M/G1	G1	S	S/G2	G2/M
I (11)	**9**	1	0	0	1
II (24)	2	**22**	0	0	0
III (24)	1	14	**1**	6	2
IV (8)	0	2	6	**0**	0
V (11)	1	0	1	1	**8**
Total	13	39	8	7	11

phase is less problematic than a misclassification into a non-adjacent phase. The results indicate that out of the 78 curves, only 6 are not assigned to their own or adjacent phases. This showcases the effectiveness of our clustering algorithm in accurately and precisely clustering these functional data.

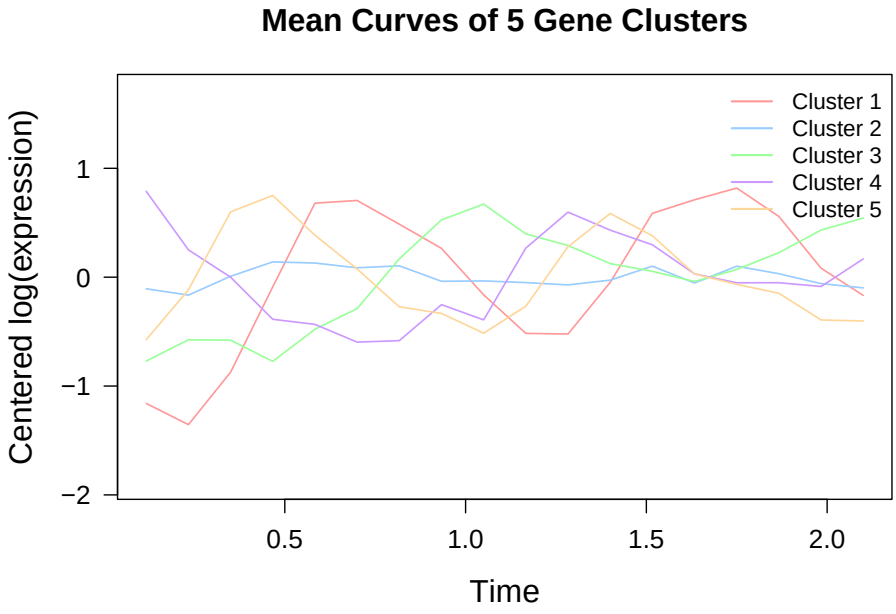

Fig. 4 Mean curves for the five-cluster predictive likelihood clustering solution for the yeast gene data.

Figure 4 shows the mean curves of the five clusters yielded by the predictive likelihood clustering solution. Such a plot can aid in making substantive distinctions between the various clusters. We see that the mean of Cluster 2 (associated with the G1 phase) has a relatively constant log expression ratio over time. Clusters 5 and 1 (which are associated with the adjacent G2/M and M/G1 phases) have similar mean curves, which display quite volatile log expression ratios which rise high and fall steeply

over time. It makes sense that the mean curves of these two clusters appear similar, since we expect phases which are adjacent in the cell cycle to behave similarly. Furthermore, Clusters 3 and 4 (associated with the adjacent S and S/G2 phases) have mean curves which are similar over the middle portion of the time domain, although clear differences in behavior of these mean curves which distinguish and separate these two clusters become apparent at early and late regions of the time domain. In short, the clustering solution appears sensible and reflects what we know about the cell-cycle-influenced behavior of these genes.

6 Conclusion

There are many clustering algorithms to classify data into groups. In our method, the predictive likelihood is a measure of how well a statistical model predicts the clustering, given the model's parameters and the data that was used to fit the model. We use a predictive likelihood as an objective function to cluster curves, allowing the number of clusters to vary during the algorithm. Our method has the advantage that we can allow the data to determine the correct number of clusters. As an example of the data-driven nature of our algorithm, in the vertical density profile data, the initial number of clusters we used was 3 and the final number of clusters as determined by the data was 7.

There are some limitations to our predictive likelihood method. For example, the algorithm is sensitive to the initial partition, although this was alleviated by using our approach, described in Section 3.6, of running the algorithm for an initial 50 iterations to get an intelligent starting partition and subsequently employing stages of 50 iterations at a time. Another issue is that the computational time required for the predictive likelihood method is considerable when the number of functional observations is large.

Overall, our method using a predictive likelihood for functional data to cluster curves is a useful method for clustering functions that can be represented with a common basis.

A Derivation of the Predictive Likelihood Objective Function

Taking the classification likelihood approach, assuming $K < N$ clusters, we have the joint density

$$f(y, z) = \prod_{i=1}^{N} \prod_{j=1}^{K} [p_j f(y_i | z_{ij})]^{z_{ij}} \tag{9}$$

If $\mathbf{Z}_1, ..., \mathbf{Z}_N$ are i.i.d. unit multinomial vectors with unknown probability vector $\mathbf{p} = (p_1, \ldots, p_K)$, and

$$Y_i | z_{ij} = 1 \sim N(\mathbf{T}\beta_j, \sigma_j^2 \mathbf{I}) \tag{10}$$

for $i = 1, ..., N$. Then marginally, the vector $\mathbf{m} = (m_1, \ldots, m_K)$ is multinomial$(K, \mathbf{p})$; conditional on $\mathbf{m}$, $\hat{\boldsymbol{\beta}}_j$ is multivariate normal and independent of $(\frac{m_j n - p^*}{\sigma_j^2})\hat{\sigma}_j^2 \sim$

$\chi^2_{m_j n-p^*}$. Then the predictive likelihood objective function, up to a proportionality constant, can be written as:

$$g(\mathbf{Z}) = \frac{f(\mathbf{Y},\mathbf{Z})}{\prod_{j\in J} f(m_j,\hat{\boldsymbol{\beta}}_j,\hat{\sigma}_j^2)}$$
$$\propto \prod_{j\in J} m_j!(m_j)^{-1/2}(\hat{\sigma}_j^2)^{-\frac{m_j n-p^*}{2}+1}\Gamma\left(\frac{m_j n-p^*}{2}\right)\left(\frac{m_j n-p^*}{2}\right)^{-\frac{m_j n-p^*}{2}}.$$

Proof:

$$f(y,z) = \prod_{i=1}^{N}\prod_{j=1}^{K}\left\{p_j\left(\frac{1}{\sqrt{2\pi}\sigma_j}\right)^n \exp\left[-\frac{1}{2\sigma_j^2}(y_i-\mathbf{T}\beta_j)'(y_i-\mathbf{T}\beta_j)\right]\right\}^{z_{ij}}$$
$$\log f(y,z) = \sum_{i=1}^{N}\sum_{j=1}^{K} z_{ij}\left\{\log p_j + \log\left(\frac{1}{\sqrt{2\pi}\sigma_j}\right)^n - \frac{1}{2\sigma_j^2}\left(y_i-\mathbf{T}\beta_j\right)'\left(y_i-\mathbf{T}\beta_j\right)\right\}$$
$$= \sum_{j=1}^{K}\left\{\log p_j^{m_j} + m_j\log\left(\frac{1}{\sqrt{2\pi}\sigma_j}\right)^n - \frac{1}{2\sigma_j^2}z_{ij}\left(y_i-\mathbf{T}\beta_j\right)'\left(y_i-\mathbf{T}\beta_j\right)\right\}$$
$$= \sum_{j=1}^{K}\left\{\log p_j^{m_j} + m_j\log\left(\frac{1}{\sqrt{2\pi}\sigma_j}\right)^n \right.$$
$$\left. - \frac{1}{2\sigma_j^2}\left[\sum_{i=1}^{N} z_{ij}(y_i-\mathbf{T}\hat{\boldsymbol{\beta}}_j)'(y_i-\mathbf{T}\hat{\boldsymbol{\beta}}_j) + m_j(\mathbf{T}\hat{\boldsymbol{\beta}}_j-\mathbf{T}\beta_j)'(\mathbf{T}\hat{\boldsymbol{\beta}}_j-\mathbf{T}\beta_j)\right]\right\} \tag{11}$$

Then the $f(y,z)$ is

$$f(y,z) = \prod_{j\epsilon J} p_j^{m_j}(\frac{1}{\sqrt{2\pi}\sigma_j})^{nm_j}\exp\left\{-\sum_{i=1}^{N} z_{ij}\frac{(y_i-\mathbf{T}\hat{\boldsymbol{\beta}}_j)'(y_i-\mathbf{T}\hat{\boldsymbol{\beta}}_j)}{2\sigma_j^2} - \right.$$
$$\left. m_j\frac{(\mathbf{T}\hat{\boldsymbol{\beta}}_j-\mathbf{T}\beta_j)'(\mathbf{T}\hat{\boldsymbol{\beta}}_j-\mathbf{T}\beta_j)}{2\sigma_j^2}\right\}$$
$$= \prod_{j\epsilon J} p_j^{m_j}(2\pi)^{-\frac{m_j n}{2}}\sigma_j^{-m_j n}\exp\left\{-m_j\frac{(\mathbf{T}\hat{\boldsymbol{\beta}}_j-\mathbf{T}\beta_j)'(\mathbf{T}\hat{\boldsymbol{\beta}}_j-\mathbf{T}\beta_j)}{2\sigma_j^2} - \frac{(m_j n-p^*)\hat{\sigma}_j^2}{2\sigma_j^2}\right\} \tag{12}$$

The denominator

$$\prod_{j\in J} f(m_j,\hat{\beta}_j,\hat{\sigma}_j^2) = \prod_{j\in J} N!\frac{p_j^{m_j}}{m_j!}\frac{1}{(2\pi)^{\frac{p^*}{2}}|cov(\hat{\beta}_j)|^{\frac{1}{2}}}\exp\left\{-\frac{1}{2}(\hat{\beta}_j-\beta_j)'[cov(\hat{\beta}_j)]^{-1}(\hat{\beta}_j-\beta_j)\right\}$$
$$\times\frac{(\frac{1}{2})^{\frac{m_jn-p^*}{2}}}{\Gamma(\frac{m_jn-p^*}{2})}(\frac{m_jn-p^*}{\sigma_j^2}\hat{\sigma}_j^2)^{\frac{m_jn-p^*}{2}-1}\exp\left\{-\frac{1}{2}(\frac{m_jn-p^*}{\sigma_j^2}\hat{\sigma}_j^2)\right\}(\frac{m_jn-p^*}{\sigma_j^2})$$
$$= N!\prod_{j\in J}\frac{p_j^{m_j}}{m_j!}\frac{1}{(2\pi)^{\frac{p^*}{2}}(\frac{\sigma_j^2}{m_j}|(\mathbf{T}'\mathbf{T})^{-1}|)^{\frac{1}{2}}}\exp\left\{-\frac{m_j}{2\sigma_j^2}(\hat{\beta}_j-\beta_j)'(\mathbf{T}'\mathbf{T})(\hat{\beta}_j-\beta_j)\right\}$$
$$\times(\hat{\sigma}_j^2)^{\frac{m_jn-p^*}{2}-1}\frac{(\frac{m_jn-p^*}{2\sigma_j^2})^{\frac{m_jn-p^*}{2}}}{\Gamma(\frac{m_jn-p^*}{2})}\exp\left\{-\frac{1}{2}(\frac{m_jn-p^*}{\sigma_j^2}\hat{\sigma}_j^2)\right\} \tag{13}$$

Hence, since

$$\exp\left\{-\frac{m_j}{2\sigma_j^2}(\hat{\beta}_j-\beta_j)'(\mathbf{T}'\mathbf{T})(\hat{\beta}_j-\beta_j)\right\} = \exp\left\{-\frac{m_j}{2\sigma_j^2}(\mathbf{T}\hat{\boldsymbol{\beta}}_j-\mathbf{T}\beta_j)'(\mathbf{T}\hat{\boldsymbol{\beta}}_j-\mathbf{T}\beta_j)\right\},$$

the predictive likelihood is

$$g(z) = \frac{f(y,z)}{\prod_{j\in J} f(m_j,\hat{\beta}_j,\hat{\sigma}_j^2)}$$
$$\propto \prod_{j\in J} m_j!(m_j)^{-\frac{1}{2}}(\hat{\sigma}_j^2)^{-\frac{m_jn-p^*}{2}+1}\Gamma\left(\frac{m_jn-p^*}{2}\right)\left(\frac{m_jn-p^*}{2}\right)^{-\frac{m_jn-p^*}{2}}$$

A.1 Definition and Derivation of $\hat{\boldsymbol{\beta}}_j$ and $\hat{\boldsymbol{\sigma}}_j^2$

Recall that $\mathbf{y_i}$ is the response vector for object i, and we denote the design matrix by $\mathbf{T}$. Then $\hat{\boldsymbol{\beta}}_j$, the estimator of the coefficients of the regression of all the objects in cluster j, is:

$$\begin{aligned}\hat{\boldsymbol{\beta}}_j &= \left[(\mathbf{T}',\ldots,\mathbf{T}')\begin{pmatrix}\mathbf{T}\\ \vdots\\ \mathbf{T}\end{pmatrix}\right]^{-1}(\mathbf{T}',\ldots,\mathbf{T}')\begin{pmatrix}\mathbf{y}_1\\ \vdots\\ \mathbf{y_{m_j}}\end{pmatrix}\\ &= [m_j\mathbf{T}'\mathbf{T}]^{-1}\left(\mathbf{T}'\mathbf{y}_1+\cdots+\mathbf{T}'\mathbf{y_{m_j}}\right)\\ &= \frac{1}{m_j}\sum_{i=1}^{m_j}(\mathbf{T}'\mathbf{T})^{-1}\mathbf{T}'\mathbf{y_i}\\ &= \frac{1}{m_j}\sum_{i=1}^{m_j}\hat{\boldsymbol{\beta}}_{ij}.\end{aligned}$$

SSE_j, the sum of squared errors of the regression of all the objects in cluster j, is:

$$
\begin{aligned}
SSE_j &= \left(\mathbf{y_1}', \ldots, \mathbf{y_{m_j}}'\right) \begin{pmatrix} \mathbf{y_1} \\ \vdots \\ \mathbf{y_{m_j}} \end{pmatrix} \\
&- \left(\mathbf{y_1}', \ldots, \mathbf{y_{m_j}}'\right) \begin{pmatrix} \mathbf{T} \\ \vdots \\ \mathbf{T} \end{pmatrix} \left[\left(\mathbf{T}', \ldots, \mathbf{T}'\right) \begin{pmatrix} \mathbf{T} \\ \vdots \\ \mathbf{T} \end{pmatrix} \right]^{-1} \left(\mathbf{T}', \ldots, \mathbf{T}'\right) \begin{pmatrix} \mathbf{y}_1 \\ \vdots \\ \mathbf{y_{m_j}} \end{pmatrix} \\
&= \mathbf{y}_1{}'\mathbf{y}_1 + \cdots + \mathbf{y}'_{\mathbf{m}_j}\mathbf{y_{m_j}} \\
&- \left(\mathbf{y}_1{}'\mathbf{T} + \cdots + \mathbf{y_{m_j}}'\mathbf{T}\right) \left[m_j \mathbf{T}'\mathbf{T}\right]^{-1} \left(\mathbf{T}'\mathbf{y}_1 + \cdots + \mathbf{T}'\mathbf{y}_{\mathbf{m}_j}\right) \\
&= \mathbf{y}_1{}'\mathbf{y_1} + \cdots + \mathbf{y_{m_j}}'\mathbf{y_{m_j}} \\
&- \frac{1}{m_j} \left(\mathbf{y_1}'\mathbf{T} + \cdots + \mathbf{y_{m_j}}'\mathbf{T}\right) \left[\left(\mathbf{T}'\mathbf{T}\right)^{-1} \mathbf{T}'\mathbf{y_1} + \cdots + \left(\mathbf{T}'\mathbf{T}\right)^{-1} \mathbf{T}'\mathbf{y_{m_j}}\right] \\
&= \frac{1}{m_j} \left[\left(\mathbf{y}_1\mathbf{y_1} - \mathbf{y_1}\mathbf{T}\left(\mathbf{T}'\mathbf{T}\right)^{-1}\mathbf{T}'\mathbf{y_1}\right) + \cdots + \left(\mathbf{y}'_{\mathbf{m}_j}\mathbf{y_{m_j}} - \mathbf{y_{m_j}}\mathbf{T}\left(\mathbf{T}'\mathbf{T}\right)^{-1}\mathbf{T}^j\mathbf{y_{m_j}}\right)\right] \\
&+ \frac{m_j - 1}{m_j} \left[\left(\mathbf{y}_1{}', \ldots, \mathbf{y_{m_j}}'\right) \mathbf{S} \begin{pmatrix} \mathbf{y}_1 \\ \vdots \\ \mathbf{y_{m_j}} \end{pmatrix}\right]
\end{aligned}
$$

(where $\mathbf{S}$ is the $m_j n \times m_j n$ block matrix with $\mathbf{I}_n$ in each diagonal block and $-\frac{1}{m_j - 1}\mathbf{T}\left(\mathbf{T}'\mathbf{T}\right)^{-1}\mathbf{T}'$ in each off-diagonal block)

$$= \left(\frac{1}{m_j}\sum_{i=1}^{m_j} SSE_i\right) + \frac{m_j - 1}{m_j}\left[\left(\mathbf{y_1}', \ldots, \mathbf{y_{m_j}}'\right)\mathbf{S}\left(\mathbf{y_1}', \ldots, \mathbf{y_{m_j}}'\right)'\right]$$

Hence, the mean squared error of the regression of all the objects in cluster j, is:

$$\hat{\sigma}_j^2 = \frac{1}{m_j}\sum_{i=1}^{m_j}\hat{\sigma}_{ij}^2 + \frac{1}{m_j n - p^*}\left(\frac{m_j - 1}{m_j}\left[\left(\mathbf{y_1}', \ldots, \mathbf{y_{m_j}}'\right)\mathbf{S}\left(\mathbf{y_1}', \ldots, \mathbf{y_{m_j}}'\right)'\right]\right).$$

Funding and Conflicts of Interests/Competing Interests

No funding was received to assist with the preparation of this manuscript. The authors have no competing interests to declare that are relevant to the content of this article.

References

Alter O, Brown PO, Botstein D (2000) Singular value decomposition for genome-wide expression data processing and modeling. Proceedings of the National Academy of

Sciences 97(18):10101–10106

Banfield JD, Raftery AE (1993) Model-based Gaussian and non-Gaussian clustering. Biometrics pp 803–821

Bjørnstad JF (1990) Predictive likelihood: A review. Statistical Science pp 242–254

Boehmke B, Greenwell BM (2019) Hands-on Machine Learning with R. Chapman and Hall/CRC

Bouveyron C, Côme E, Jacques J (2015) The discriminative functional mixture model for a comparative analysis of bike sharing systems. Annals of Applied Statistics 9(4):1726–1760

Butler RW (1986) Predictive likelihood inference with applications. Journal of the Royal Statistical Society Series B: Statistical Methodology 48(1):1–23

Černỳ V (1984) Minimization of continuous functions by simulated annealing. University of Helsinki. Research Institute for Theoretical Physics

Chamroukhi F (2016) Piecewise regression mixture for simultaneous functional data clustering and optimal segmentation. Journal of Classification 33(3):374–411

Chamroukhi F, Nguyen HD (2019) Model-based clustering and classification of functional data. Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery 9(4):e1298

Chen H, Reiss PT, Tarpey T (2014) Optimally weighted l2 distance for functional data. Biometrics 70(3):516–525

Delaigle A, Hall P, Pham T (2019) Clustering functional data into groups by using projections. Journal of the Royal Statistical Society: Series B (Statistical Methodology) 81(2):271–304

Eisen MB, Spellman PT, Brown PO, et al (1998) Cluster analysis and display of genome-wide expression patterns. Proceedings of the National Academy of Sciences 95(25):14863–14868

Everitt BS, Landau S, Leese M, et al (2011) Cluster Analysis. Wiley-Blackwell, Chichester, UK

Febrero-Bande M, de la Fuente MO (2012) Statistical computing in functional data analysis: The R package fda.usc. Journal of Statistical Software 51(4):1–28. https://doi.org/10.18637/jss.v051.i04, URL https://www.jstatsoft.org/index.php/jss/article/view/v051i04

Ferraty F, Vieu P (2006) Nonparametric Functional Data Analysis. Springer, New York

Handl J, Knowles J, Kell DB (2005) Computational cluster validation in post-genomic data analysis. Bioinformatics 21(15):3201–3212

Hébrail G, Hugueney B, Lechevallier Y, et al (2010) Exploratory analysis of functional data via clustering and optimal segmentation. Neurocomputing 73(7-9):1125–1141

Hitchcock DB, Greenwood MC (2015) Clustering functional data. In: Handbook of Cluster Analysis. Chapman and Hall/CRC, Boca Raton, FL, pp 265–288

Jacques J, Preda C (2014) Functional data clustering: a survey. Advances in Data Analysis and Classification 8:231–255

Kirkpatrick S, Gelatt Jr CD, Vecchi MP (1983) Optimization by simulated annealing. Science 220(4598):671–680

Metropolis N, Rosenbluth A, Rosenbluth M, et al (1953) Simulated annealing. Journal of Chemical Physics 21(161-162):1087–1092

Nguyen X, Gelfand AE (2011) The Dirichlet labeling process for clustering functional data. Statistica Sinica pp 1249–1289

Nourani Y, Andresen B (1998) A comparison of simulated annealing cooling strategies. Journal of Physics A: Mathematical and General 31(41):8373

R Core Team (2024) R: A Language and Environment for Statistical Computing. R Foundation for Statistical Computing, Vienna, Austria, URL https://www.R-project.org/

Ramsay JO, Silverman BW (2002) Applied Functional Data Analysis: Methods and Case Studies. Springer, New York

Rand WM (1971) Objective criteria for the evaluation of clustering methods. Journal of the American Statistical Association pp 846–850

Robert CP, Casella G (1999) Monte Carlo Statistical Methods. Springer, New York

Rossi F, Conan-Guez B, El Golli A (2004) Clustering functional data with the SOM algorithm. In: ESANN, Citeseer, pp 305–312

Rousseeuw PJ (1987) Silhouettes: a graphical aid to the interpretation and validation of cluster analysis. Journal of Computational and Applied Mathematics 20:53–65

Tarpey T, Kinateder KK (2003) Clustering functional data. Journal of Classification 20(1):93–114

Walker E, Wright SP (2002) Comparing curves using additive models. Journal of Quality Technology 34(1):118–129

Wu R, Wang B, Xu A (2022) Functional data clustering using principal curve methods. Communications in Statistics - Theory and Methods 51(20):7264–7283

Zhang M, Parnell A (2023) Review of clustering methods for functional data. ACM Transactions on Knowledge Discovery from Data 17(7):1–34