Clustering time-course data using P-splines and mixed effects mixture models

by

Deidre Bredenkamp

(04639864)

Submitted in partial fulfillment of the requirements for the degree

Masters of Commerce in Advanced Data Analytics

in the Faculty of Natural and Agricultural Science

Department of Statistics University of Pretoria, Pretoria

UNIVERSITEIT VAN PRETORIA
UNIVERSITY OF PRETORIA
YUNIBESITHI YA PRETORIA

January, 2022

Contents

List of Tables

List of Algorithms

List of Figures

Declaration

I, Deidre Bredenkamp declare that the thesis/dissertation, which I hereby submit for the degree MSc Advanced Data Analytics, at the University of Pretoria, is my own work and has not previously been submitted by me for a degree at this or any other tertiary institution.

SIGNATURE: .....

DATE: January, 2022

Abstract:

In the field of biology, gene expressions are evaluated over time to study complicated biological processes and genetic supervisory networks. Because the process is continuous, time-course gene-expression data may be represented by a continuous function.

This mini dissertation addresses cluster analysis of time-course data in a mixture model framework. To take into account the time dependency of such time-course data, as well as the degree of error present in many datasets, the mixed effects model with penalized B-splines is considered. In this mini dissertation the performance of such a mixed effects model has been studied with regards to the clustering of time-course gene expression data in a mixture model system. The EM algorithm has been implemented to fit the mixture model in a mixed effects model structure. For each subject the best linear unbiased smooth estimate of its time-course trajectory has been calculated and subjects with similar mean curves have been clustered in the same cluster. Model validation statistics such has the model accuracy and the coefficient of determination (R^2) indicates that the model can cluster simulated data effectively into clusters that differ in either the form of the curves or the timing to the curves' peaks. The proposed technique is further evidenced by clustering time-course gene expression data consisting of microarray samples from lung tissue of mice exposed to different Influenza strains from 14 time-points.

1 Introduction

Gene expression is the process through which the genetic makeup in DNA is translated into a material object, such as a protein and the cell structures are formed. Microarrays are used to store gene expression data throughout time. Time-course microarray experiments, in which genes are examined repeatedly throughout time, generate massive volumes of high-dimensional data.

Clustering algorithms play an important part in reducing the dimensionality of such data. The problem that arises in most periodic genetic experiments is that the number and shape of the time-course gene expression distributions are unknown. Therefore, in order to obtain a statistically relevant set of curves and clusters, a semi-parametric clustering approach is considered. This clustering approach also takes into consideration the correlation of the time-course data between time points and should be able to deal with the issue of missing data.

Well known techniques such as self-organizing maps (SOM) [2], k-means clustering [3] and hierarchical clustering [4] don't conform to the mentioned prerequisites. A method that has shown great promise is the use of a traditional multivariate Gaussian model [5] to take the correlation framework into account, but this model disregards the time-course dependency of the gene expressions.

Since the time dependency of the real data is a fundamental factor in analyzing the grouping solutions of longitudinal data, it is necessary to explore more reliant models.

An alternative method to analyze the gene expressions series over time is to make use of an auto-regression model, cluster analysis of gene expression dynamics (CAGED) [6]. The problem with these models is that they require the Markov property to hold as well as data that is stationary, which is not necessarily the case for time-course microarray data.

Y. Luan et. al [7], established the mixed effects model using B-splines to reflect the temporal dependability of gene expressions recorded across time, as well as the unexplained variability present in the microarray data. They further investigated these mixed effects models in analysing genetics data as well as the performance of the grouping of genes in a mixture model structure. Using an expectation-maximization (EM) algorithm [8], they calculated the smoothed average gene expression curve for each cluster by fitting a mixture of the mixed-effects model. They were able to integrate data from that specific gene and other genes in the same cluster after obtaining a smooth linear unbiased estimate of each gene's expression. They showed that these techniques were efficient in clustering both noisy and time-varying data.

In the paper by JJ. Song et. al [9], they proposed clustering time-course gene expression profiles by employing a procedure based on functional data analysis (FNDA). This was done since FNDA gives solutions to take the dependence on time into consideration by employing basis function extension to describe the partial data curves. They applied this method to real as well as simulated data with two sets of basis functions, Fourier transformation and B-splines. It was found that the proposed method using a Fourier basis accurately clustered

all the sample curves of the simulated data. On the real yeast cell cycle data, it was found that the Fourier basis grouped the gene expressions more accurately than using B-spline basis functions. It was shown using BIC (Bayesian Information Criterion) [10] that the Fourier basis approach outperformed the B-spline approach.

In the paper by PD. McNicholas et. al [11] the combinations of multivariate t distributions, with a linear mean model and an updated Cholesky-decomposed covariance framework, have been used to cluster longitudinal data. By putting restrictions on the covariance structure, the model consists of a collection of mixture models that have been used for model-based clustering and classification. These models have been used for data that has been simulated, as well as for real time-course gene expression data sets. The models fitted to the simulated data were compared to their Gaussian analogues. It was found that the mixture of multivariate t-distributions outperformed it's Gaussian counterpart, in cases where the underlying distributions are heavy-tailed. Complications regarding parameter estimation in terms of the degrees of freedom is reported. They suggest that non-parametric approaches might be a solution to more accurate parameter estimation.

L. Wang et. al [12] observed data regarding gene-expression levels in the course of a certain biological process to determine the transcriptional factors (TF) in gene regulation. In this paper, they proposed a practical reaction model with coefficients that vary, to show the transcriptional outcomes on genetic levels and in doing so created a cluster smoothly clipped absolute deviation (SCAD) regression technique to select the TFs. They fitted the model on simulated data as well as on actual yeast cell cycle data. It was found that the SCAD regression procedure was accurate in obtaining the variables that are relevant to time-varying coefficients, as well as the estimation of those coefficients. The authors determined that with the actual data, the SCAD procedure identified more known cell cycle-related TF's at each time-point, compared to that identified with simple linear regression (SLR). They concluded that the SCAD procedure is indeed very efficient for the identification of variables with time-varying coefficients.

In the paper by Y. Zeng et. al, [13] to analyze periodic gene expression data, a novel hidden Markov model (HMM) is proposed. This model takes the time-related genetic descriptions into consideration, which is disregarded by many other clustering approaches. The clustering results are determined by clustering time series that are most probable to conform to the same design. The model is 'novel' since the data is displayed by a single HMM serving as a SOM. They displayed the performance on the model by fitting it to simulated data, as well as real gene expression time-course data. A comparison study was done on the simulated data where they compared the performance of the HMM model to that of general grouping approaches (k-means, self-organizing maps (SOM), complete-linkage and mixture model of Gaussians (MOG)). It was found that the HMM outperformed the other models for data that has been simulated, as well as for real time-course gene expression data, in terms of the BIC index.

The possible dependence of genetic levels on a single gene over a given length of time is an important element of this time-course gene expression data. As the level of gene expression evolves over a time period,

time can be a fundamental component that affects the level of gene expression.

Algorithms capable of retaining the time sequence and time dependence of the data, such as clustering methods, are necessary in the investigation of time-course gene expression data.

It should be noted, that although most studies in the literature review have analysed time-course gene expression data, these researched methods are applicable to a variety of longitudinal data.

To account for the time dependability of periodic observations and the random distribution of microarray data in this study, we consider the clustering of longitudinal profiles using a mixture model in a mixed-effects model with penalized B-splines (PB-splines) framework. Furthermore, this analysis examines a similar mixed-effects model in the analysis of repeated genetics data, as well as gene clustering in a mixture model structure.

The reason for considering PB-splines is that it is faster and easier to create a model with random components, in which a specific fixed component does not exist, because the fixed component will be automatically taken care of by the model.

In this mini dissertation we aim to obtain a mixed-effects model to model longitudinal data. We propose this specific model, by reason that the model allows one to take the time factor of the data into account. The main objective of this research is to cluster the longitudinal data, such that similar observations are clustered together. Clustering of the data will be using a mixture model basis and making use of the EM algorithm as well as the BIC and Restricted Maximum likelihood (REML) to determine the amount of clusters in the model and the model parameters. In addition we aim to estimate the gene expression trajectory, this will enable us to estimate the gene expression at any specific time (t) as well as to determine the time to maximum gene expression levels. To show the proposed methods, we will first simulate relevant data and fit the proposed model to it. The model is then evaluated using model validation statistics such as model accuracy and the coefficient of determination $R2$. Finally the proposed model is fitted to real-world time-course gene expression data and the results are displayed graphically.

2 Penalized Splines

2.1 Introduction

The word spline describes a sculptor's tool that is a pliable narrow piece of wood or metal that is used to design smooth curves. Multiple weights would be introduced in multiple locations, causing the strip to bend in accordance with the quantity and position of the weights. This would be pushed to pass through a series of fixed locations, such as metal pins, boat ribs and so on. The bent material's shape would naturally adopt the shape of a spline curve. In statistics a spline refers to a continuous function that represents a flexible curve. Knots are added to the function at various points throughout the data range to mark the regions where continuous functional parts connect together. This means that in the place of wooden or metal pieces, as in the sculptor's tool, the statistician's spline consists of smooth functional parts that fit the data between successive knots.

Splines are commonly used for interpolation and extrapolation of data collected at distinct points. The usage of penalized splines is a prominent method for simplifying the selection of knot locations in spline modelling.

In this section the mini dissertation explores the concepts and methods that will be utilised in this study. In particular B-splines and following on this penalized splines, as well as a mixed effects model with P-splines. A toy example will be discussed to illustrate the above mentioned methods and techniques.

2.2 B-Splines

A spline is a piece-wise polynomial function. This function, denoted as $f(x)$, is produced by partitioning the domain of $\boldsymbol{X}$ into continuous segments. We will represent $f(x)$ by a distinct polynomial in each segment, these will be referred to as basis functions. The knots κ_i indicate where the intervals start or finish .

B-spline basis functions allow for effective computing even for a large K, where K denotes the number of knots, as discussed by Hastie et al. [14]. For B-splines, we let $\xi_0 < \xi_1$ and $\xi_K < \xi_{K+1}$be the two boundary knots. B-splines also have an augmented knot sequence such that,

$$\kappa_1 \leq \kappa_2 \leq ... \leq \kappa_M \leq \xi_0,$$
$$\text{where } \kappa_{j+M} = \xi_j \text{for } j = 1, ..., K,$$
$$\text{and } \xi_{K+1} \leq \kappa_{K+M+1} \leq ... \leq \kappa_{K+2M}.$$

These knots' values beyond the boundary points are not set to fixed values and can take on any value beyond the boundary point, therefore we set them equal to ξ_0 and ξ_{K+1} respectively. We set $B_{i,m}(x)$ to be the i^{th} m-order B-spline basis function for the knot sequence $\kappa_m, m \leq M$, where we have the Haar basis function [15]

$$B_{i,1}(x) = \begin{cases} 1 & if\ \kappa_i \le x \le \kappa_{i+1} \\ 0 & ow. \end{cases} \quad \text{where } i = 1, ..., K + 2M - 1$$

De Boor's algorithm [16], which calculates the basis functions recursively, is given as,

$$B_{i,m}(x) = \frac{x - \kappa_i}{\kappa_{i+m-1} - \kappa_i} B_{i,m-1} + \frac{\kappa_{i+m} - x}{\kappa_{i+m} - \kappa_{i+1}} B_{i+1,m-1}. \tag{1}$$

Suppose $M = 4$ then we have $B_{i,4}$, where $i = 1, \ldots, K+4$ are the $K+4$ cubic B-spline basis functions for the knot sequence ξ. Since additional knots have been added, one needs to avoid dividing by zero.

For B-spline basis functions of order $m \le M$ with knots at ξ, only the subset $B_{i,m}$ for $i = M - m + 1, ..., M + K$ will be required. When we include the inner knot ξ_j, $1 \le r_j \le M$ times, the lowest order derivative is discontinuous at $x = \xi_j$ and will be of order $M - r\,j$.

For cubic splines with no repeated knot values, for example, $r_j = 1$ and $j = 1, ..., K$, the 3^{rd} order derivative will be discontinuous. If the j^{th} knot is repeated 3 times we would have $M - r_j = 1$,implying that the 1^{st} derivative will be discontinuous.

Least squares solutions for N observations and $K + M$ variables take

$$O(N(K+M)^2 + (K+M)^3)$$

floating point operations for a large number of knots, K. O denotes the asymptotic rate of growth [17].

If the N observations are ordered, the $N \times (K+M)$ regression matrix with $K+M$ basis functions analyzed at N points will include many zeros, resulting in a low computation power with just $O(N)$ floating point operations.

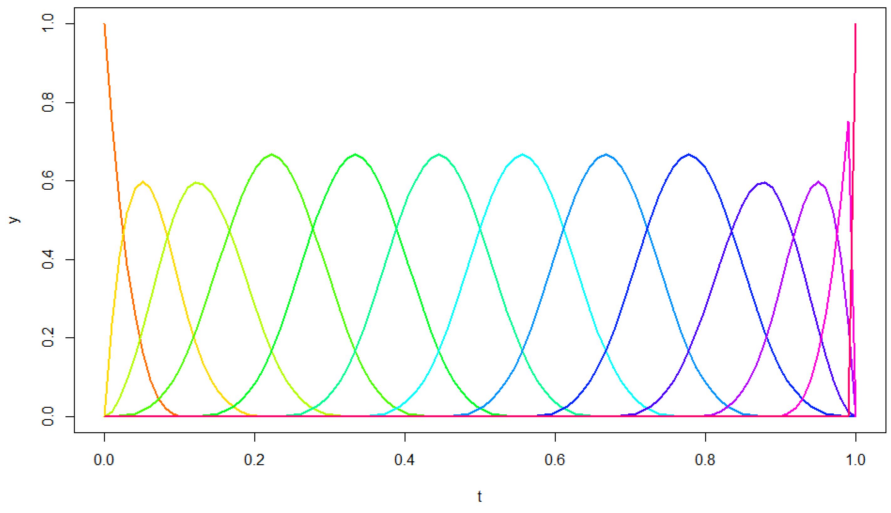

Figure 1: The cubic B-spline basis functions derived using De Boor's algorithm are plotted for $K = 10$ internal knots. In this figure each basis function is represented by a different colour.

Suppose we have a dataset with N subjects with n_i observations measured at time points $\boldsymbol{t}_i = (t_{i1}, \ldots, t_{in_i})$. Let $y_{ij} = \boldsymbol{y}_i(t_{ij})$ be the outcome measured on the i^{th} subject at time t_{ij}. For the sake of simplicity, suppose time is the only covariate.

Consider the regression model

$$\boldsymbol{y}_i = f(\boldsymbol{t}_i) + \varepsilon_i. \tag{2}$$

For B-splines we have

$$f(\boldsymbol{t}_i) = \sum_{j=-m}^{K} \boldsymbol{\beta}_j \boldsymbol{B}_j(\boldsymbol{t}_i), \tag{3}$$

where $\boldsymbol{\beta}_j$ represents the coefficients to the spline functions denoted by B_j, which are cubic B-splines, i.e. of order $m = 3$.

To estimate the basis function coefficients the function $f(t)$ is denoted as a linear regression model

$$\begin{aligned} \boldsymbol{Y} &= f(\boldsymbol{t}) + \varepsilon \\ &= \boldsymbol{B}\boldsymbol{\beta} + \varepsilon, \end{aligned} \tag{4}$$

where $\boldsymbol{Y}$ contains the raw expression values, which can be represented on a $N \times n_i$ matrix, $\boldsymbol{Y} = (\boldsymbol{y}_1, \ldots, \boldsymbol{y}_N)^T$ measured at time points $\boldsymbol{t} = (\boldsymbol{t}_1, \ldots, \boldsymbol{t}_N)^T$. The model for subject i at time t_{ij} can be expressed as

$$\boldsymbol{y}_i(t_{ij}) = \sum_{\ell=-m}^{K} \boldsymbol{\beta}_\ell \boldsymbol{B}_\ell(t_{ij}) + \varepsilon_{ij}.$$

The design matrix $\boldsymbol{B}$ in (4) has the form

$$\boldsymbol{B} = \begin{bmatrix} B_{1,-m} & B_{1,-m+1} & \ldots & B_{1,K} \\ B_{2,-m} & B_{2,-m+1} & \ldots & B_{2,K} \\ \vdots & \vdots & \ddots & \vdots \\ B_{N,-m} & \ldots & \ldots & B_{N,K} \end{bmatrix},$$

$\boldsymbol{\beta} = (\beta_{-m}, \ldots, \beta_K)^T$ is a vector of coefficients and we assume the residual error component $\boldsymbol{\varepsilon} \sim N(0, \sigma_\varepsilon^2 \boldsymbol{V})$. The columns of B are the assayed B-spline basis functions evaluated at the sorted $\boldsymbol{X}$ values, in order from left to right. The B-spline coefficients can be estimated via least-squares, such that the residual sum of squares is given by

$$RSS = ||\boldsymbol{y} - f(\boldsymbol{t})||_2^2,$$

where $||.||_2$ is an L_2 norm. Unfortunately the curve obtained by minimising $||y - B\beta||^2$ with respect to $\boldsymbol{\beta}$ shows a greater variation than what is present in the data if a dense set of spline functions are used.

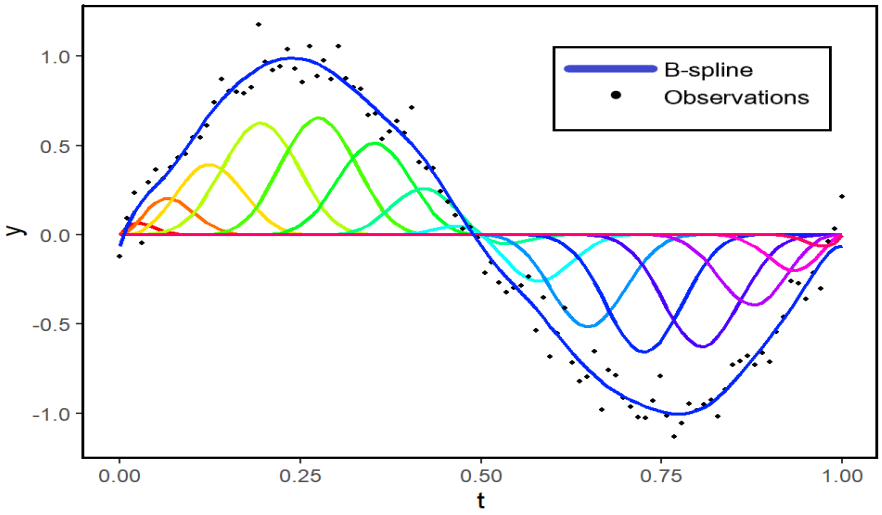

Figure 2: The B-splines model in (4) is fitted on a simulated dataset. The data was simulated from a $sin(2\pi t)$ function with a Gaussian residual added to it. This is a cubic ($m = 3$) B-spline basis with $K = 10$ knots, resulting in a total of $q = K + m + 1 = 14$ basis functions. In this figure each spline is represented by a different colour curve and the B-spline regression line(blue) is plotted onto the data points.

In Figure 2, it can be seen that the B-splines regression line fits the data quite well, but it can tend to overfit the data when the number of knots are increased. In order to avoid overfitting, we will estimate β in a penalized regression environment.

2.3 Smoothing Splines (P-splines)

In this section we introduce smoothing splines, also known as penalized splines (P-splines). P-splines performed on a B-spline basis was first introduced by Eilers et al. [18]. To model and approximate the smooth function f_i, penalized regression splines are used. The P-spline method implies that a polynomial spline in terms of a linear combination of B-spline basis functions will approximate the function f for a given a covariate x. The challenge here is to determine the number and placement of the knots; if we have too few knots, the model will not be flexible enough to encompass the data's volatility and too many knots will result in extreme overfitting, meaning the model will fit the data too well.

The use of P-splines is justified due to the many advantages of this modelling method. One advantage of using P-splines is that the challenge of deciding on the location and size of knots is essentially solved; as long as we select a sufficiently large number of knots, the penalty function guarantees that the resulting fits are very close. The difference penalty on adjacent B-splines coefficients provides the necessary smoothness to the fitted curve[19]. The difference penalty is depicted as a difference matrix $\mathbf{\Delta}$. When P-splines are used, weights are assigned to each of the splines to penalize overfitting the data while also encouraging the splines to match the data well.

According to [20], utilizing P-splines hold many other advantages: The discrete smoother of P-splines is simple and powerful to use. The P-splines method can be implemented on normal as well as non-normal data. Penalties applied on the model pushes the solution in the desired direction, in addition by implementing P-splines, the model can handle data with missing observations. Penalties can be combined to established models such as Bayes and Mixed models. P-splines aim to minimise the penalized sum of squares (PSS) denoted by,

$$\begin{aligned} PSS &= ||\boldsymbol{y} - f(\boldsymbol{t})||_2^2 + \lambda ||\boldsymbol{\Delta}_d \boldsymbol{\beta}||_2^2 \\ &= ||\boldsymbol{y} - \boldsymbol{B}\boldsymbol{\beta}||_2^2 + \lambda ||\boldsymbol{\Delta}_d \boldsymbol{\beta}||_2^2, \end{aligned} \tag{5}$$

where λ is the penalty and $\boldsymbol{\Delta}_d$ is the d^{th} order penalty matrix such that $\boldsymbol{\Delta}_d\boldsymbol{\beta}$ is a vector of the d^{th} order differences of the coefficients. So therefore, the first order difference matrix will be denoted $\boldsymbol{\Delta}\boldsymbol{\beta} = \beta_i - \beta_{i-1}$, thus we will have

$$\boldsymbol{\Delta} = \begin{bmatrix} -1 & 1 & 0 & 0 & \dots & 0 \\ 0 & -1 & 1 & 0 & \dots & 0 \\ 0 & 0 & \ddots & \ddots & & 0 \\ \vdots & & & & & \vdots \\ 0 & 0 & \dots & & -1 & 1 \end{bmatrix}.$$

Rewriting Equation (5) in matrix notation we have

$$PSS = (\boldsymbol{y} - f(\boldsymbol{t}))^T \boldsymbol{V}^{-1} (\boldsymbol{y} - f(\boldsymbol{t})) + \lambda \boldsymbol{\beta}^T \boldsymbol{\Delta}^T \boldsymbol{\Delta} \beta \tag{6}$$

$$= (\boldsymbol{y} - \boldsymbol{B}\boldsymbol{\beta})^T \boldsymbol{V}^{-1} (\boldsymbol{y} - \boldsymbol{B}\boldsymbol{\beta}) + \lambda \boldsymbol{\beta}^T \boldsymbol{\Delta}^T \boldsymbol{\Delta}\boldsymbol{\beta}. \tag{7}$$

The estimate of $\hat{\beta}$ is obtained by minimising the penalized sum of squares (PSS) with respect to β

$$\frac{\delta PSS}{\delta \beta} = -2\boldsymbol{B}^T \boldsymbol{V}^{-1} (\boldsymbol{y} - \hat{\boldsymbol{\beta}}\boldsymbol{B}) + 2\lambda \boldsymbol{\Delta}^T \boldsymbol{\Delta}\hat{\boldsymbol{\beta}}. \tag{8}$$

Setting Equation (8) equal to zero yields

$$2\boldsymbol{B}^T \boldsymbol{V}^{-1} (\boldsymbol{y} - \hat{\boldsymbol{\beta}}\boldsymbol{B}) = 2\lambda \boldsymbol{\Delta}^T \boldsymbol{\Delta}\hat{\beta}..$$

For a given λ the solution to the optimisation problem satisfies

$$\boldsymbol{B}^T \boldsymbol{V}^{-1} y = (\boldsymbol{B}^T \boldsymbol{V}^{-\mathbf{1}} \boldsymbol{B} + \lambda \boldsymbol{\Delta}^T \boldsymbol{\Delta})\hat{\boldsymbol{\beta}},$$

yielding

$$\hat{\boldsymbol{\beta}} = (\boldsymbol{B}^T\boldsymbol{V}^{-1}\boldsymbol{B} + \lambda\boldsymbol{\Omega}_B)^{-1}\boldsymbol{B}^T\boldsymbol{V}^{-1}\boldsymbol{y}, \tag{9}$$

with $\boldsymbol{\Omega}_B = \boldsymbol{\Delta}^T\boldsymbol{\Delta}$. Note that no boundary derivative constraints are taken into account, since the penalty term imposes these constraints, therefore there is infinite weight to non zero derivatives.

Then

$$\hat{\boldsymbol{y}} = \boldsymbol{B}\hat{\boldsymbol{\beta}} = \boldsymbol{H}\boldsymbol{y},$$

where H is the hat matrix such that

$$\boldsymbol{H} = \boldsymbol{B}(\boldsymbol{B}^T\boldsymbol{V}^{-1}\boldsymbol{B} + \lambda\boldsymbol{\Omega}_B)^{-1}\boldsymbol{B}^T\boldsymbol{V}^{-1}.$$

This term is one of the primary motivations for using P-splines for smoothing. To begin, the term $(\boldsymbol{y}-\boldsymbol{B}\boldsymbol{\beta})^T\boldsymbol{V}^{-1}(\boldsymbol{y}-\boldsymbol{B}\boldsymbol{\beta})$ relates to ordinary regression on the columns of $\boldsymbol{B}$ and $\lambda\boldsymbol{\beta}^T\boldsymbol{\Delta}^T\boldsymbol{\Delta}\boldsymbol{\beta}$ addresses over parameterisation of the regression model by penalizing the smoothness of β_j. Secondly, since $\boldsymbol{B}^T\boldsymbol{V}^{-1}\boldsymbol{B} + \lambda\boldsymbol{\Delta}^T\boldsymbol{\Delta}$ is a $q \times q$ matrix for $q = K + m + 1$, there is a computational superiority for a limited number of knots, this advantage is critical for a big population of N observations.

Cubic B-splines have local support, this means that the computational complexity of estimating $\hat{f} = \boldsymbol{B}\hat{\boldsymbol{\beta}}$ can be dramatically reduced to $O(N)$ operations.

By combining regression on B-spline basis functions with a discrete roughness penalty, P-splines smooth scatter plots. The basis is rich, implying that the number of basis functions was specified to be suitably large. In this situation, a simple regression model would provide an unduly inconsistent fit to the observations.

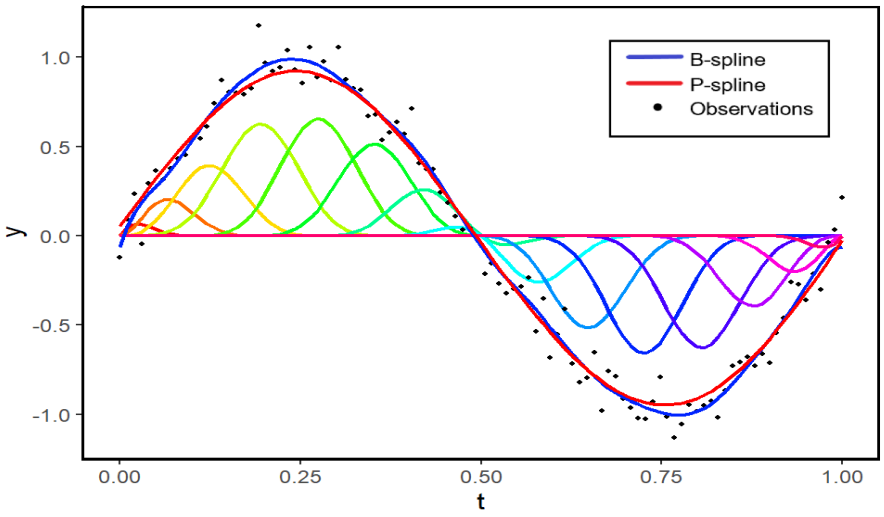

Figure 3: The B-splines model is fitted to the simulated dataset (blue line), simulated from a $sin(2\pi t)$ function with a Gaussian residual added to it (black dots). The basis functions are plotted, where each spline is represented by a different colour. The regression on the B-spline basis functions is penalized with a discrete roughness penalty resulting in a P-splines model which has a smooth fit (red line) to the data.

From Figure 3 we see that the P-spline model captures the shape of the model quite accurately and more smoothly than the B-splines regression line in Figure 2.

A penalty offers additional, continuous control over the smoothness of a model and because the penalty is discrete, it is simple to enforce, in contrast to a penalty imposed on the integral of the squared second-order derivative of the function fitted to the data. Another benefit is that the features of a P-spline model can be portrayed in a visually appealing way.

P-splines provide simple interpolation and extrapolation, allowing the power penalties to be clearly evident. The penalty works in such a manner that even in the event of incomplete data, with enough knots the automatically created interpolating curves will be smooth and consistent in shape.

When interpolating, the B-spline coefficients form a sequence of $2d-1$ elements and when extrapolating, the B-spline coefficients form a sequence of $d-1$ elements . That is, if for example $d=2$, then we get cubic interpolation and linear extrapolation.

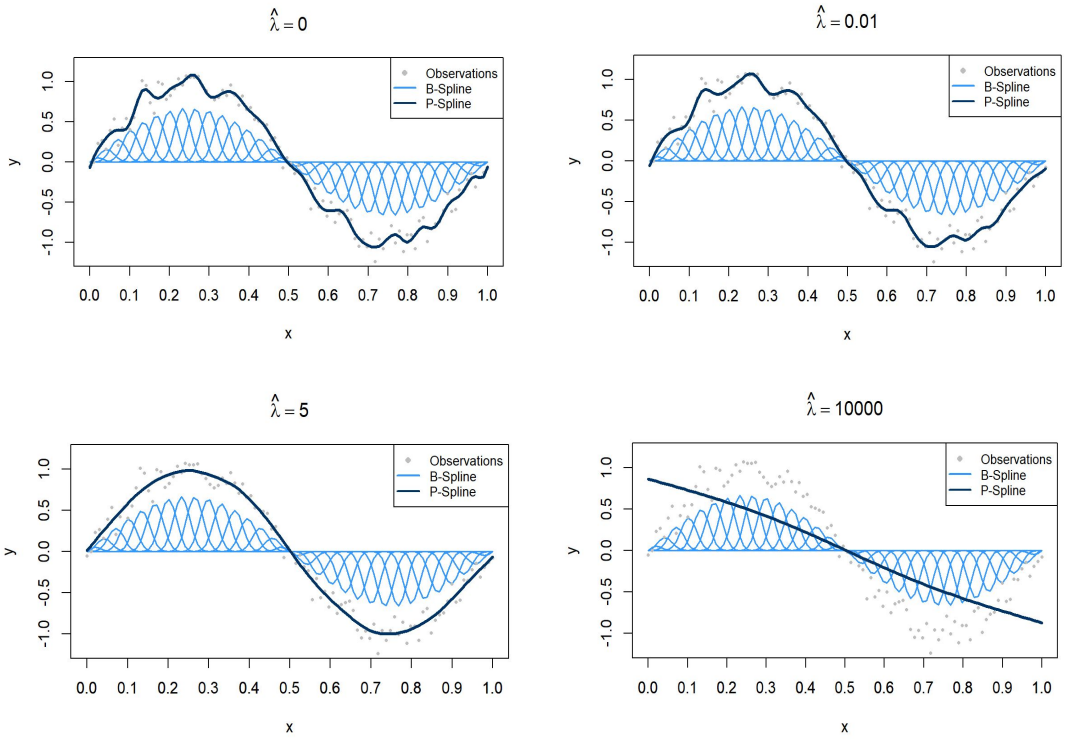

Figure 4: A depiction of the estimated P-splines with $K=30$ knots for various smoothing parameter values $\hat{\lambda}$, with the smallest value as the first plot (top left) and the largest value at the final plot (bottom right) (for brevity, only four values are examined). A Gaussian residual was added to a sine curve function of the type $sin(2\pi x)$ to simulate the data.

In Figure 4 cross-validation was used to determine the best smoothing parameter [21]. Second-order penalties have been imposed to cubic B-splines defined on equally spaced knots. From these plots we can clearly see that the curve tends to be smoother for larger values of $\hat{\lambda}$, whereas the smaller the value of $\hat{\lambda}$, the more likely it is that the model will over fit the data.

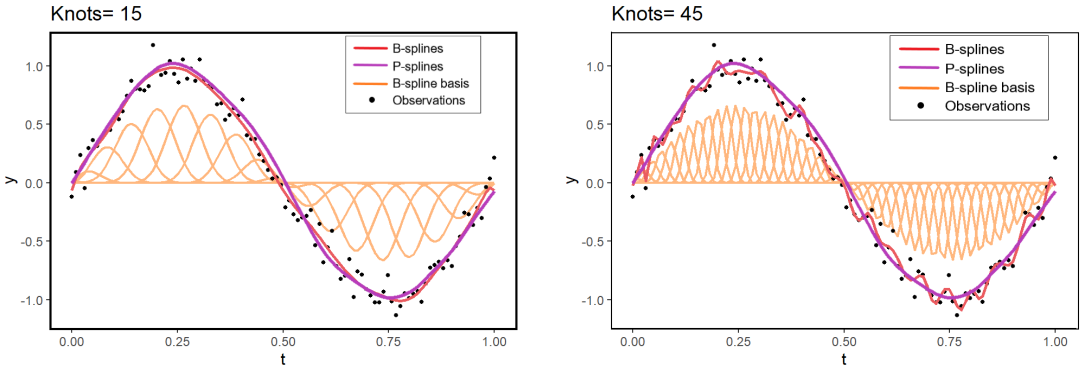

Figure 5: The figure displays the fitted B-splines regression curves(red lines) and the smoothing estimated P-splines (purple line) for $K = 15$ (on the left hand side) and $K = 45$ (on the right hand side) equally spaced knots. The B-spline basis functions (orange lines) are also present in the plots for both cases of 15 and 45 knots. The best smoothing parameters λ were chosen using generalized cross-validation (GCV).

It should be noted that for $\lambda \to 0$: The final smoother tends to interpolate the observations, thus over-fitting the model. And for $\lambda \to \infty$: the fitted function will tend to a polynomial of degree $d-1$.

2.4 Mixed models with P-splines

While the P-spline model permits for smooth patterns, there may be non-smooth patterns within the data either by design or due to unmeasured variables. In the event that there are repeated values of the latent variables, these non-smooth effects can be modelled by including an extra random component in the model. That is, the mixed effects model automatically makes provision for the irregularities that may appear in the data.

Consider the function of the model in (4), for the data $\boldsymbol{Y} = (\boldsymbol{y}_1, ...\boldsymbol{y}_N)^T$, where the $\boldsymbol{y}_i's$, $i = 1, \ldots, N$ are independent, measured at the values $\boldsymbol{t} = (\boldsymbol{t}_1, ..., \boldsymbol{t}_N)^T$ where $f(.)$ is an unknown smooth function and the vector of the errors is given by $\boldsymbol{\varepsilon} = (\boldsymbol{\varepsilon}_1, ..., \boldsymbol{\varepsilon}_N)$, where $\boldsymbol{\varepsilon} \sim N(0, R)$ where $\boldsymbol{R} = \sigma_\varepsilon^2 \boldsymbol{I}$. The variance-covariance matrix of the data takes the shape of a block diagonal matrix which will have equal diagonal blocks σ_ε^2. In other words the covariance is the same for all $\boldsymbol{\varepsilon}_i, i = 1, \ldots, N$.

According to [22], a structured additive regression (STAR) model such as in (3), can be written in the form of a generalized linear mixed model (GLMM) in the context of smoothing splines. In order to write the function $f(.)$ as a GLMM, the vector of coefficients $\boldsymbol{\beta}$ can be decomposed into a penalized and unpenalized part.

Consider the coefficient vector $\boldsymbol{\beta}_{:q\times 1}$ and the corresponding penalty matrix $\boldsymbol{\Omega}_B = \boldsymbol{\Delta}^T \boldsymbol{\Delta}$ with rank $rk = q - d$, where d is the order of the differencing matrix $\boldsymbol{\Delta}$. Then we define the decomposition as

$$\boldsymbol{\beta} = \boldsymbol{\Psi}^{(unp)} \boldsymbol{\beta}^{(unp)} + \boldsymbol{\Psi}^{(pen)} \boldsymbol{\beta}^{(pen)}, \tag{10}$$

where the columns of $\boldsymbol{\Psi}^{(unp)}_{:q\times d}$ consists of the basis of the null-space of $\boldsymbol{\Omega}_B$. The matrix $\boldsymbol{\Psi}^{(pen)}_{:q\times rk}$ is depicted as $\boldsymbol{\Psi}^{(pen)} = \boldsymbol{L}(\boldsymbol{L}^T\boldsymbol{L})^{-1}$, where the full-column rank matrix $\boldsymbol{L}_{:q\times rk}$ is determined by the decomposition of $\boldsymbol{\Omega}_B$

into $\boldsymbol{\Omega}_B = \boldsymbol{L}\boldsymbol{L}^T$. In order for this decomposition to be functional it has to meet the following requirement: $\boldsymbol{L}^T\boldsymbol{\Psi}^{(unp)} = 0$ and $\boldsymbol{\Psi}^{(unp)T}\boldsymbol{L} = 0$. That is, the vector $\boldsymbol{\beta}^{(unp)}$ represents the part of $\boldsymbol{\beta}$ that is not penalized by $\boldsymbol{\Omega}_B$. On the other hand, $\boldsymbol{\beta}^{(pen)}$ denotes the deviation of $\boldsymbol{\beta}$ from the null space of $\boldsymbol{\Omega}_B$.

In general, the decomposition $\boldsymbol{\Omega}_B = \boldsymbol{L}\boldsymbol{L}^T$ of $\boldsymbol{\Omega}_B$ can be evaluated by the spectral decomposition $\boldsymbol{\Omega}_B = \boldsymbol{\Gamma}\boldsymbol{\Lambda}\boldsymbol{\Gamma}^T$, where $\boldsymbol{\Lambda}_{:rk\times rk}$ represents the diagonal matrix of the positive eigen-values of $\boldsymbol{\Omega}_B$ in descending order, such that $\Lambda = diag(\boldsymbol{\Omega}_1,\ldots\boldsymbol{\Omega}_{rk})$. And $\boldsymbol{\Gamma}_{:q\times rk}$ denotes the orthogonal matrix of the corresponding eigen-vectors. In the event of P-splines an appropriate choice of $\boldsymbol{L}$ would be $\boldsymbol{L} = \boldsymbol{\Delta}$[22].

Substituting (10) into (4) f is the simple mixed model

$$f(\boldsymbol{t}) = \boldsymbol{X}\boldsymbol{\tau} + \boldsymbol{Z}\boldsymbol{u},$$

where $\boldsymbol{X} = \boldsymbol{B}\boldsymbol{\Psi}^{(unp)}$ denote the $N\times d$ matrix and $\boldsymbol{Z} = \boldsymbol{B}\boldsymbol{\Psi}^{(pen)}$ a $N\times rk$ matrix, such that $\boldsymbol{X}$ and $\boldsymbol{Z}$ are the known design matrices, $f = (f(\boldsymbol{t}_1),\ldots,f(\boldsymbol{t}_N))$ and $\boldsymbol{\tau} = (\tau_1,\ldots,\tau_d) = \boldsymbol{\beta}^{(unp)}$ is the unknown fixed effects coefficients and $\boldsymbol{u} = (u_1,\ldots,u_{(q-d)}) = \boldsymbol{\beta}^{(pen)} \sim N(0,G)$ is the unknown random effects coefficients and $\boldsymbol{G} = \sigma_u^2\boldsymbol{I}$ is its variance-covariance matrix. The value d denotes the number of unpenalized terms and $q-d$ is the number of penalized terms.

Thus we can write the mixed model formulation for a cubic P-spline with a d^{th} order difference penalty as

$$\begin{aligned}\boldsymbol{y} &= \boldsymbol{B}\boldsymbol{L}\boldsymbol{L}^T\boldsymbol{\beta} + \boldsymbol{B}\boldsymbol{\Delta}_d^T(\boldsymbol{\Delta}_d\boldsymbol{\Delta}_d^T)^{-1}\boldsymbol{\Delta}_d\boldsymbol{\beta} + \boldsymbol{\varepsilon}\\ &= \boldsymbol{X}\boldsymbol{\tau} + \boldsymbol{Z}\boldsymbol{u} + \boldsymbol{\varepsilon}\end{aligned} \tag{11}$$

.

We consider the matrix $\boldsymbol{B}\boldsymbol{L}$ to be a linear transformation of a polynomial model of degree $d-1$ of the vector $\boldsymbol{t}$ [23], thus a full rank matrix $\boldsymbol{A}_{d\times d}$ such that $\boldsymbol{B}\boldsymbol{L}\boldsymbol{A} = \boldsymbol{X} = [\boldsymbol{1},\boldsymbol{t}^1,...,\boldsymbol{t}^{d-1}]$. Then $\boldsymbol{\tau} = \boldsymbol{A}^{-1}\boldsymbol{L}^T\boldsymbol{\beta}$, is a vector of unknown coefficients with d elements, $\boldsymbol{Z} = \boldsymbol{B}\boldsymbol{\Delta}_d^T(\boldsymbol{\Delta}_d\boldsymbol{\Delta}_d^T)^{-1}$ and $\boldsymbol{u} = \boldsymbol{\Delta}_d\boldsymbol{\beta}$. The P-spline penalty for the smoothing parameter $\lambda = \gamma^{-1} = \sigma_\varepsilon^2/\sigma_u^2 = \gamma^{-1}\boldsymbol{u}^T\boldsymbol{u}$. The shape of the model's fixed component is determined by the order of the differencing. For d^{th} order differencing, the fixed component of the model reflects a polynomial of degree $d-1$.

Consider the PSS (6). The least squares term becomes

$$\sum_{i=1}^{N}(\boldsymbol{y}_i - f(\boldsymbol{t}_i))^2 = (\boldsymbol{Y} - \boldsymbol{X}\boldsymbol{\tau} + \boldsymbol{Z}\boldsymbol{u})^T\boldsymbol{R}^{-1}(\boldsymbol{Y} - \boldsymbol{X}\boldsymbol{\tau} + \boldsymbol{Z}\boldsymbol{u}),$$

where R denotes the variance-covariance matrix of the data noise $\boldsymbol{\varepsilon}$. For a known value of the smoothing parameter $\lambda \geq 0$ the P-spline will be the curve that minimises the penalized sum of squares (PSS)

$$(\boldsymbol{Y}-\boldsymbol{X\tau}+\boldsymbol{Zu})^T R^{-1}(\boldsymbol{Y}-\boldsymbol{X\tau}+\boldsymbol{Zu})+\boldsymbol{u}^T\boldsymbol{G}^{-1}\boldsymbol{u},$$

where $\boldsymbol{G}^{-1}$ is a known penalty matrix of the form $\boldsymbol{G}^{-1}=\lambda\boldsymbol{\Delta}_d^T\boldsymbol{\Delta}_d$. If $\lambda\rightarrow\infty$ then $f(\boldsymbol{t})=\boldsymbol{X\tau}$ will be a subset of the unpenalized terms.

Consider the model

$$\boldsymbol{Y}=\boldsymbol{X\tau}+\boldsymbol{Zu}+\boldsymbol{\varepsilon}. \tag{12}$$

Following the approach as in the study by Saputro et al.[24], the expected value of the model is given by $E(\boldsymbol{Y})=\boldsymbol{X\tau}$,with the variance-covariance matrix obtained as, $cov(\boldsymbol{Y})=\boldsymbol{ZGZ}^T+\boldsymbol{R}$.

For a non-singular $\boldsymbol{G}$ the joint density of $\boldsymbol{y}$ and $\boldsymbol{u}$ is

$$\begin{aligned} f_{y,u} &= f_{y|u}f_u \\ &= |2\pi\boldsymbol{R}|^{-1/2}\exp\left(-\frac{1}{2}(\boldsymbol{Y}-\boldsymbol{X\tau}+\boldsymbol{Zu})^T\boldsymbol{R}^{-1}(\boldsymbol{Y}-\boldsymbol{X\tau}+\boldsymbol{Zu})\right).|2\pi\boldsymbol{G}|^{-1/2}\exp(-\frac{1}{2}\boldsymbol{u^T G^{-1}u}). \end{aligned}$$

The logarithm of the joint density function is

$$l_f=\log(f_{y,u})=-\frac{1}{2}\log|2\pi\boldsymbol{R}|-\frac{1}{2}(\boldsymbol{Y}-\boldsymbol{X\tau}+\boldsymbol{Zu})^T\boldsymbol{R}^{-1}(\boldsymbol{Y}-\boldsymbol{X\tau}+\boldsymbol{Zu})-\frac{1}{2}\log|2\pi\boldsymbol{G}|-\frac{1}{2}\boldsymbol{u}^T\boldsymbol{G^{-1}u}.$$

In order to find the parameter estimates for $\boldsymbol{\tau}$ and $\boldsymbol{u}$ the partial derivative of the logarithm of the joint density is calculated with respect to each parameter respectively, such that the partial derivative with respect to $\boldsymbol{u}$ is

$$\frac{\delta l_f}{\delta u}=\boldsymbol{Z}^T\boldsymbol{R^{-1}}(\boldsymbol{Y}-\boldsymbol{X\tau}-\boldsymbol{Zu})+\boldsymbol{G^{-1}u}, \tag{13}$$

and the partial derivative with respect to $\boldsymbol{\tau}$ is

$$\frac{\delta l_f}{\delta\boldsymbol{\tau}}=\boldsymbol{X}^T\boldsymbol{R}^{-1}(\boldsymbol{Y}-\boldsymbol{X\tau}-\boldsymbol{Zu}). \tag{14}$$

To estimate the values of u and $\boldsymbol{\tau}$ that maximises equations (13) and (14) we set these equations equal to zero such that

$$\boldsymbol{Z}^T\boldsymbol{R}^{-1}\boldsymbol{Y}-\boldsymbol{Z}^T\boldsymbol{R}^{-1}\boldsymbol{X\tau}-(\boldsymbol{Z}^T\boldsymbol{R}^{-1}\boldsymbol{Z}+\boldsymbol{G}^{-1})\boldsymbol{u}=0,$$

and

$$\boldsymbol{X}^T\boldsymbol{R}^{-1}\boldsymbol{Y}-\boldsymbol{X}^T\boldsymbol{R}^{-1}\boldsymbol{X\tau}-\boldsymbol{X}^T\boldsymbol{R}^{-1}\boldsymbol{Zu}=0 \tag{15}$$

As shown in [23] the solution of this system of linear equations in matrix form is

$$\begin{bmatrix} \boldsymbol{X^T R^{-1} X} & \boldsymbol{X^T R^{-1} Z} \\ \boldsymbol{Z^T R^{-1} X} & \boldsymbol{Z^T R^{-1} Z - G^{-1}} \end{bmatrix} \begin{bmatrix} \hat{\boldsymbol{\tau}} \\ \hat{\boldsymbol{u}} \end{bmatrix} = \begin{bmatrix} \boldsymbol{X^T R^{-1}} \\ \boldsymbol{Z^T R^{-1}} \end{bmatrix} \boldsymbol{y}, \tag{16}$$

with $\boldsymbol{G}^{-1} = \lambda \boldsymbol{\Delta}_d^T \boldsymbol{\Delta}_d$.

From Equation 16 u is solved as

$$\boldsymbol{u} = (\boldsymbol{Z^T R^{-1} Z} - \boldsymbol{G}^{-1})^{-1} \boldsymbol{Z^T R^{-1}} (\boldsymbol{Y} - \boldsymbol{X\tau}). \tag{17}$$

Substituting Equation (17) into Equation (15) gives

$$\boldsymbol{X}^T \boldsymbol{R}^{-1} \boldsymbol{Y} - \boldsymbol{X}^T \boldsymbol{R}^{-1} \boldsymbol{X\tau} - \boldsymbol{X}^T \boldsymbol{R}^{-1} \boldsymbol{Z} (\boldsymbol{Z^T R^{-1} Z} - \boldsymbol{G}^{-1})^{-1} \boldsymbol{Z^T R^{-1}} (\boldsymbol{Y} - \boldsymbol{X\tau}) = 0. \tag{18}$$

Let $\boldsymbol{V} = \boldsymbol{ZGZ^T} + \boldsymbol{R}$, where $\boldsymbol{G} = \sigma_u^2 \boldsymbol{I}$ with $\sigma_u^2 = \lambda^{-1} \sigma_\varepsilon^2$ and $\boldsymbol{R} = \sigma_\varepsilon^2 \boldsymbol{I}$, such that $\boldsymbol{V} = \boldsymbol{\sigma_u^2 ZZ^T} + \boldsymbol{\sigma_\varepsilon^2 I}$.

Simplifying Equation (18) gives

$$\begin{aligned} \boldsymbol{X^T V^{-1} Y} &= \boldsymbol{X^T V^{-1} X\tau} \\ \hat{\boldsymbol{\tau}} &= (\boldsymbol{X^T V^{-1} X})^{-1} \boldsymbol{X^T V^{-1} Y}. \end{aligned} \tag{19}$$

Following this, u is estimated as

$$\begin{aligned} \hat{u} &= \boldsymbol{GZ^T V^{-1}} (\boldsymbol{Y} - \boldsymbol{X}\hat{\boldsymbol{\tau}}) \\ &= \sigma_u^2 \boldsymbol{Z^T V^{-1}} (\boldsymbol{Y} - \boldsymbol{X}\hat{\boldsymbol{\tau}}). \end{aligned} \tag{20}$$

The best linear unbiased predictor (BLUP) for the parameter $\boldsymbol{\tau}$ is exactly the same as the solution of generalised least squares (GLS) and the BLUP of $\boldsymbol{u}$ is the best linear predictor of u where one substitutes $\boldsymbol{\tau}$ with $\hat{\boldsymbol{\tau}}$. One of the easiest ways to obtain the BLUP is by applying Robinson's [25] justification, by assuming the distribution, $\boldsymbol{Y}|u \sim N(\boldsymbol{X\tau} + \boldsymbol{Zu}, \boldsymbol{R})$ and $u \sim N(0, G)$. The BLUP of $(\boldsymbol{\tau}, \boldsymbol{u})$ is

$$\begin{bmatrix} \hat{\boldsymbol{\tau}} \\ \hat{u} \end{bmatrix} = (\boldsymbol{C^T R^{-1} C} + \boldsymbol{\lambda \Omega_B})^{-1} \boldsymbol{C^T R^{-1} Y},$$

where $\boldsymbol{C} = [\boldsymbol{X}\ \boldsymbol{Z}]$ and $\boldsymbol{\Omega}_B = \boldsymbol{\Delta}^T \boldsymbol{\Delta}$, thus the BLUP of $\boldsymbol{Y}$ is

$$\hat{\boldsymbol{Y}} = \boldsymbol{X}\hat{\boldsymbol{\tau}} + \boldsymbol{Z}\hat{\boldsymbol{u}} = \boldsymbol{C^T} (\boldsymbol{C^T C} + \boldsymbol{\lambda \Omega_B})^{-1} \boldsymbol{CY}.$$

The most frequently used method of estimation is maximum likelihood (ML) and REML. The smoothing parameter λ, where $\gamma = \frac{1}{\lambda}$ is a pseudo-variable parameter which will be estimated using REML along with σ_ε^2 and ϕ. The reason for using REML instead of ML is due to the fact that it is a less biased estimate of the variable parameters given the fixed effects in a model [25]. The estimators of REML are calculated according to the conditional model $\boldsymbol{Y}|u \sim N(\boldsymbol{X\tau} + \boldsymbol{Zu}, \boldsymbol{V})$. The REML log-likelihood leads to the estimation of the variance component V and is given as

$$L_{REML} = -\frac{1}{2}\left[\log|\boldsymbol{V}| + \log|\boldsymbol{X^T V^{-1} X}| + \frac{\boldsymbol{Y}^T \boldsymbol{P Y}}{\sigma_\varepsilon^2} + (n-d)\log(\sigma_\varepsilon^2)\right], \tag{21}$$

where

$$\boldsymbol{P} = \boldsymbol{V^{-1}} - \boldsymbol{V^{-1}X}(\boldsymbol{X^T V^{-1} X})^{-1}\boldsymbol{X^T V^{-1}}.$$

We perform maximization using numeric optimisation techniques such as the Newton Rhapson method and the EM-Algorithm.

We assume that the smoothing splines take knots at K distinct time points in the data $\boldsymbol{t} = (t_1, ..., t_K)$.

For $q = K + m + 1$, $\boldsymbol{\Delta}_{d(q-d)\times(q)}$ is the d^{th} order difference matrix: For example, take a cubic P-spline ($m = 3$) with second-order differencing ($d = 2$) and K equally spaced knots. The matrix $\boldsymbol{\Delta}_2$ is

$$\boldsymbol{\Delta}_2 = \begin{bmatrix} 1 & -2 & 1 & \dots & 0 & 0 \\ 0 & 1 & -2 & 1 & \dots & 0 \\ \vdots & \ddots & \ddots & \ddots & \ddots & \vdots \\ 0 & 0 & \dots & 1 & -2 & 1 \end{bmatrix},$$

that is

$$[\boldsymbol{\Delta}_2]_{ij} = \begin{cases} 1 & j = i, i = 1, 2, ..., K-2 \\ -2 & j = i+1, i = 1, 2, ..., K-2 \\ 1 & j = i+2, i = 1, 2, ..., K-2 \\ 0 & \text{otherwise} \end{cases}.$$

2.4.1 Toy example of mixed effect models with P-splines

In this section we explore a toy example on a simulated dataset. The data is simulated from a $sin(2\pi t)$ function with a Gaussian error, with mean zero and a standard deviation of 0.1, added to it. The data has been simulated on 100 time points $\boldsymbol{t} = (t_1, \dots, t_{100})$ where $t_i \in (0, 1), i = 1, \dots, 100$.

A mixed effects model with P-splines (MMP-splines), along with a P-splines and B-splines model is fitted to the data and the results are compared in Figure 6. The MMP-splines model is fitted using the **MMBsplines** [1] function in the R software.

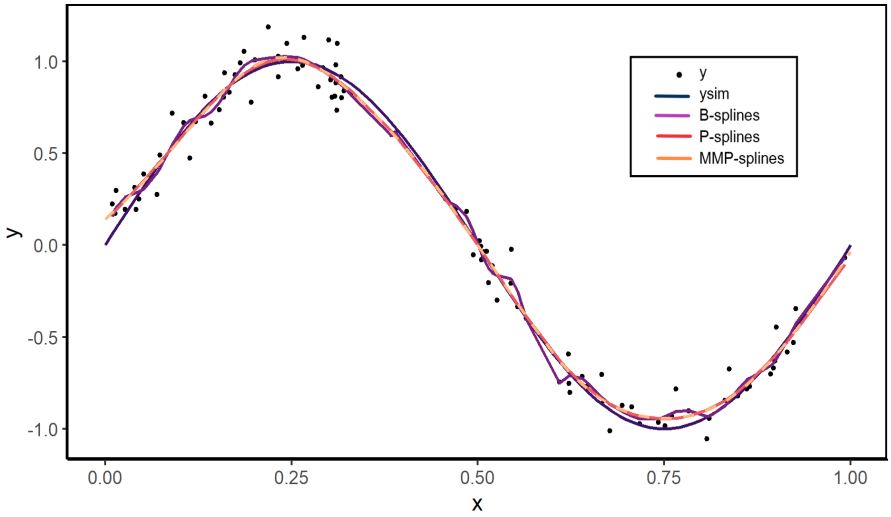

Figure 6: Plot of a mixed effects model with P-splines (MMP-splines) (orange line) plotted on the simulated data form a $sin(2\pi t)$ function (navy line) with a Gaussian residual added to it (black dots) using **MMBsplines** [1]. The plot also indicates the fitted P-splines model (red line), along with the fitted B-splines model (purple line), with $K = 20$ equally spaced knots.

From Figure 6 we can see that the mixed effects model with P-splines proves to be an accurate fit to the true simulated line (blue line). It can also be seen that the P-splines model and the MMP-splines model are very similar, such that it seems that the MMP-splines model fits the P-splines model nearly perfectly.

For this toy example, the data was simulated using the model

$$Y = sin(2\pi t) + \varepsilon,$$

where $\varepsilon \sim N(0, \sigma_\varepsilon)$, having $\sigma_\varepsilon = 0.1$. In the plot in Figure 6 the blue line in the plot is the simulated data without the error, i.e. it simply plots $sin(2\pi t)$.

As for the other models plotted, these are all spline models for which we will assume that $K = 20$ knots are used. Cubic splines will be incorporated therefore the parameter m will take the value 3.

The B-splines model (4) is represented by the purple line in the plot above, where the model has the shape

$$\boldsymbol{Y} = \boldsymbol{B}\boldsymbol{\beta} + \boldsymbol{\varepsilon},$$

such that the B-splines design matrix is obtained using De Boor's algorithm (1)

The B-splines basis function design matrix, for 100 ordered time points is obtained to be a matrix with 100 rows and $q = K + m + 1 = 24$ columns, such that

$$\boldsymbol{B}_{100\times 24} = \begin{bmatrix} 0.08552 & 0.63087 & 0.28229 & 0.00132 & 0.00000 & \dots & 0 & 0 \\ 0.05997 & 0.59533 & 0.34069 & 0.00401 & 0.00000 & \dots & 0 & 0 \\ 0.05543 & 0.58680 & 0.35294 & 0.00483 & 0.00000 & \dots & 0 & 0 \\ \vdots & & \ddots & & & \ddots & & \vdots \\ 0 & 0.00000 & \dots & 0.02707 & 0.50709 & 0.45020 & 0.01564 & 0.00000 \\ 0 & 0.00000 & \dots & 0.00000 & 0.00075 & 0.26043 & 0.64173 & 0.09709 \end{bmatrix},$$

and $\boldsymbol{\beta}$, the B-spline coefficients are estimated by means of least squares and using the *lsfit* function in R. Such that

$$\hat{\boldsymbol{\beta}} = [3.572,\ 0.162,\ 1.388,\ \dots\ 1.290,\ 0.333,\ 1.283]^T.$$

The P-splines model as discussed in Section 2.2 was fitted to the data, using the *psplines* function in Rand is represented by the red line in Figure 5 . The penalize cubic B-splines model with a second order penalty is given as

$$Y = \boldsymbol{B}\boldsymbol{\beta}^* + \boldsymbol{\varepsilon}, \tag{22}$$

where $\boldsymbol{B}$ is the same as obtained for the B-splines model and $\hat{\boldsymbol{\beta}}^*$ is estimated as shown in Equation (9), such that

$$\hat{\boldsymbol{\beta}}^* = [-0.0282, 0.1640, 0.3650, \dots -0.3632, -0.0997, 0.1668]^T.$$

The penalty matrix $\boldsymbol{\Delta}_2$ is calculated as the second order differencing matrix, with $q-d = 22$ rows and $q = 24$ columns

$$\boldsymbol{\Delta}_2 = \begin{bmatrix} 1 & -2 & 1 & \dots & 0 & 0 \\ 0 & 1 & -2 & 1 & \dots & 0 \\ \vdots & \ddots & \ddots & \ddots & \ddots & \vdots \\ 0 & 0 & \dots & 1 & -2 & 1 \end{bmatrix}.$$

The penalty parameter λ has been estimated using GCV such that

$$\hat{\lambda} = 1.833195.$$

As for the mixture model with *P*splines (12), which is represented by the orange line in the plot and is shown to be nearly identical to the red line, representing the P-splines model. This serves as an indication that

Using the **MMBsplines** function in the *R* software we obtained the parameter values for the model (12)

$$\boldsymbol{Y} = \boldsymbol{X}\boldsymbol{\tau} + \boldsymbol{Z}\boldsymbol{u} + \boldsymbol{\varepsilon}.$$

For this model, we use 20 knots and penalize cubic B-splines with a second order penalty matrix. That is,

$$K = 20,\ m = 3,\ d = 2, \text{ and } q = K + m + 1 = 24.$$

The design matrix for the fixed effects $\boldsymbol{X}$, is a $N = 100$ by $d = 2$ matrix, with the first column being a unit vector and the second column being the observed time points, such that

$$\boldsymbol{X} = \begin{bmatrix} 1 & 0.009 \\ 1 & 0.013 \\ 1 & 0.014 \\ \vdots & \vdots \\ 1 & 0.926 \\ 1 & 0.992 \end{bmatrix}.$$

The design matrix for the random effects $\boldsymbol{Z} = \boldsymbol{B}\boldsymbol{\Delta}_d^T(\boldsymbol{\Delta}_d\boldsymbol{\Delta}_d^T)^{-1}$ is a matrix with $N = 100$ rows and $q - d = 22$ columns, such that this matrix is

$$\boldsymbol{Z} = \begin{bmatrix} -0.059 & 0.377 & 0.968 & 1.443 & 1.810 & \dots & 0.175 & 0.061 \\ -0.084 & 0.293 & 0.884 & 1.362 & 1.732 & \dots & 0.172 & 0.060 \\ -0.088 & 0.276 & 0.867 & 1.346 & 1.716 & \dots & 0.172 & 0.060 \\ \vdots & \ddots & & \ddots & \ddots & \ddots & & \vdots \\ 0.048 & 0.136 & \dots & 0.642 & 0.220 & -0.275 & -0.370 & -0.131 \\ 0.062 & 0.176 & \dots & 1.840 & 1.475 & 1.001 & 0.410 & -0.048 \end{bmatrix}.$$

The respective estimated fixed and random effects are

$$\hat{\boldsymbol{\tau}} = (0.544, -1.088)^T$$

$$\hat{\boldsymbol{u}} = (-0.0148,\ -0.110,\ -0.123,\ -0.092,\ -0.092,\ -0.017,\ -0.050\ -0.106,\ -0.147,\ -0.047,$$
$$0.0467,\ 0.041,\ 0.042,\ 0.024,\ 0.042,\ 0.104,\ 0.095,\ 0.156,\ 0.075,\ -0.003,\ -0.008,\ -0.001)^T$$

The error variance $\hat{\sigma}_\varepsilon^2$ and the random effect variance $\hat{\sigma}_u^2$ are

$$\hat{\sigma_\varepsilon}^2 = 0.0082 \text{ and } \hat{\sigma}_u^2 = 0.0149,$$

hence the penalty parameter $\lambda = \gamma^{-1} = \sigma_\varepsilon^2/\sigma_s^2$ can be estimated to be $\hat{\lambda} = 0.546$ and so the model can be

estimated using the values obtained above to be

$$\hat{\boldsymbol{y}} = \boldsymbol{X}\hat{\boldsymbol{\tau}} + \boldsymbol{Z}\hat{u}.$$

From the above analysis it can clearly be seen that the models of P-splines (22) and the mixed model with P-splines (12) are completely different models with different parameter values and different design matrices, nevertheless the plot in Figure 6 indicates that these models are identical in terms of fit to the data.

2.5 Conclusion

The use of splines and in turn penalized splines are an excellent tool for describing complicated non-linear processes. The use of P-splines simplifies the approach of fitting flexible models to data without having to be concerned with calculating the number of knots, instead the penalty controls the level of smoothness of the splines function. That along with the efficiency of a mixed effects model in taking elements such as time dependency into account makes for a powerful tool in modelling complex data. This has been illustrated in this section by applying the mixed effects model with P-splines to a simulated time dependent dataset. Both the P-splines model and the proposed mixed effects model performed exactly the same in terms of fitting the data, despite the fact that the models are completely different, showing that the P-splines model can be represented by a mixed effects model.

3 P-Splines Based Mixture modelling

3.1 Introduction

Heterogeneous data are prevalent in statistics and emerge when data samples are drawn from a population consisting of multiple different groups. Data consisting of different subgroups is often analysed by means of mixture models, which are usually composed of different components that are able to model the various groups in the dataset. A mixture model is essentially a convex combination of numerous statistical distributions representing the various underlying groups in populations.

Using P-splines in a clustering framework holds many precedence, such as the fact that the use of P-splines allow us to model cluster average expression curves and subject specific data distributions in an amalgamated framework. P-spline clustering models are able to function efficiently even in the presence of missing data, we can also performs automatic smoothing parameter (λ) estimation via REML. In the event that latent variables are present in a model, the expectation-maximization algorithm is a method for executing maximum likelihood estimation. This is accomplished by first estimating the values of the latent variables, then optimizing the model and then iterating between these two steps until convergence is reached. It is a powerful and versatile method that is often used in density estimation with missing data, such as clustering algorithms.

Customary software such as SAS, R, S-plus and Python can be used to fit mixtures of mixed effects models. We can incorporate replicated curves for each subject by specifying supplementary random effects in the model. The mixed model with P-splines is flexible and can manage multiple extensions, we can add numerous covariates in the fixed effects design matrix $\boldsymbol{X}$, for the within cluster variation. For time-varying covariates we can add these via the mixtures of generalized additive mixed (GAM) models.

3.2 Mixture modelling using P-spline Mixed effects models

Using the P-splines model, we will now apply clustering. This is accomplished by extending Equation (12) to represent subject based clusters with subject-specific variance around the cluster mean.

Suppose we know that subject i falls into cluster c, then that specific subject's expression profile can be formulated as

$$y_{ij} = \mu_c(t_{ij}) + \gamma_i + \varepsilon_{ij}.$$

For $i = 1, \ldots, N_c$ and $j = 1, \ldots, n_i$, where N_C represents the number of observations in cluster c and n_i counts the set of expression values measured for gene i. The mean expression curve for cluster c is $\mu_c(t)$, $\gamma_i \sim N(0, \sigma^2_{\gamma,c})$ is an additional random effect to account for gene specific variation around the mean expression curve for gene i and $\varepsilon_{ij} \sim N(0, \sigma^2_{\varepsilon,c})$ is the remnant error for gene i expression j.

By layering the data from specific genes and using the linear mixed effects depiction of the P-splines, we

can now approximate the cluster mean expression $\mu_c(t)$ and the gene specific expression profiles for all N_c genes in cluster c, resulting in a model for the gene expression profiles in cluster c given by

$$\boldsymbol{y}_c = \boldsymbol{X}_{c,s}\boldsymbol{\tau}_{c,s} + \boldsymbol{Z}_{c,s}\boldsymbol{u}_{c,s} + \boldsymbol{Z}_{c,\gamma}\boldsymbol{\gamma}_c + \boldsymbol{\varepsilon}_c, \tag{23}$$

where

$$\begin{aligned} \boldsymbol{y}_c &= (\boldsymbol{y}_1^T, \boldsymbol{y}_2^T, ..., \boldsymbol{y}_{N_c}^T)^T & \boldsymbol{y}_i &= (y_{i,1}, ..., y_{i,n_i}) \\ \boldsymbol{X}_{c,s} &= (\boldsymbol{X}_{1,s}^T, ..., \boldsymbol{X}_{N_c,s}^T)^T & \boldsymbol{Z}_{c,s} &= (\boldsymbol{Z}_{1,s}, ..., \boldsymbol{Z}_{N_c,s})^T \\ \boldsymbol{Z}_{c,\gamma} &= diag(\boldsymbol{Z}_{1,\gamma}, ..., \boldsymbol{Z}_{N_c,\gamma}) & \boldsymbol{Z}_{i,\gamma} &= (1, ..., 1)^T. \end{aligned}$$

The subscript s is used to demonstrate the sections of the model to which smoothing is applied. The design matrix $\boldsymbol{Z}_{c,\gamma}$ provides an opportunity for the variation between expression profiles of different genes in the same cluster. In other words, it is permissible for the i^{th} gene in cluster c to have a gene-specific drift γ_i (or a random intercept) from its cluster mean curve. $\boldsymbol{Z}_{i,\gamma}$ is an $n_i \times 1$ vector for the i^{th} gene specific random intercept. We assume that

$$\begin{aligned} \boldsymbol{u}_{c,s} &\sim N(0, \sigma^2_{u,c}\mathrm{I}), \\ \boldsymbol{\gamma}_c &\sim N(0, \sigma^2_{\gamma,c}\mathrm{I}) \text{ and} \\ \boldsymbol{\varepsilon}_c &\sim N(0, \sigma^2_{\varepsilon,c}\mathrm{I}) \end{aligned}$$

are independent.

In cluster c, the model for gene c can now be written as

$$\boldsymbol{y}_i = \underbrace{\boldsymbol{X}_{i,s}\boldsymbol{\tau}_{c,s} + \boldsymbol{Z}_{i,s}\boldsymbol{u}_{c,s}}_{\mu_c(t_i)} + \boldsymbol{Z}_{i,\gamma}\gamma_i + \varepsilon_i, \tag{24}$$

where $\boldsymbol{X}_{i,s}, \boldsymbol{Z}_{i,s}$ and $\boldsymbol{Z}_{i,\gamma}$ are the design sub-matrices for gene i and $\boldsymbol{x}_i$ is the vector of times measured for gene i. For clustering, it is believed that the i^{th} gene in cluster c is distributed as a function conditional on the smoothing random effect $\boldsymbol{u}_{c,s}$

$$\boldsymbol{y}_i | \boldsymbol{u}_{c,s} \sim N(\boldsymbol{X}_{i,s}\boldsymbol{\tau}_{c,s} + \boldsymbol{Z}_{i,s}\boldsymbol{u}_{c,s}, \boldsymbol{V}_{i,c}),$$

where $\boldsymbol{V}_{i,c} = \sigma^2_{\gamma,c}\boldsymbol{Z}_{i,\gamma}\boldsymbol{Z}_{i,\gamma}^T + \sigma^2_{\varepsilon,c}\mathrm{I}_{n_i \times n_i}$ is the corresponding $n_i \times n_i$ square matrix of the square covariance matrix $\boldsymbol{V}_c = \sigma^2_{\gamma,c}\boldsymbol{Z}_{c,\gamma}\boldsymbol{Z}_{c,\gamma}^T + \sigma^2_{\varepsilon,c}\mathrm{I}$ for cluster c.

This implies that $\sigma^2_{\gamma,c}$ and $\sigma^2_{\varepsilon,c}$ are equal for all genes in the same cluster c, so $\boldsymbol{V}_{i,c}$ is only dependent on i in terms of dimension. Since the random effects in the vector $\boldsymbol{u}_{c,s}$ are only used to achieve a smooth approx-

imation of $\mu_c(t)$, this variance variable is not used in the matrix $\boldsymbol{V}_{i,c}$ when determining the $\boldsymbol{y}_i$ distribution for clustering.

In reality, the cluster affiliation is unclear, so it is presumed that $\boldsymbol{y}_i$ is derived from a combination of C components, so that we have

$$f_Y(\boldsymbol{y}_i;\Theta) = \sum_{c=1}^{C} \pi_c f_c(\boldsymbol{y}_i,\theta_c),$$

which represents a mixture model where $f_c(.)$ represents the densities of components that depend, for example, on the vector of unknown parameters $\theta_c = (\boldsymbol{\tau}_{c,s}, \sigma^2_{u,c}, \sigma^2_{\gamma,c}, \sigma^2_{\varepsilon,c})$, $\Theta = (\theta_1^T, ..., \theta_C^T, \pi_1, ..., \pi_C)^T$ and $\pi_1, ..., \pi_C$ are the mixing probabilities that need to be estimated in such a way that

$$\sum_{c=1}^{C} \pi_c = 1.$$

The smooth mean curve in each cluster c is then estimated based on the random effects $\boldsymbol{u}_{c,s}$ as

$$\begin{aligned} f_Y(\boldsymbol{y}_i|\boldsymbol{u}_{c,s};\Theta) &= \sum_{c=1}^{C} \pi_c N(\mu_c(t_i), \boldsymbol{V}_{i,c}) \\ &= \sum_{c=1}^{C} \pi_c N(\boldsymbol{X}_{i,s}\boldsymbol{\tau}_{c,s} + \boldsymbol{Z}_{i,s}\boldsymbol{u}_{c,s}, \boldsymbol{V}_{i,c}) \end{aligned} . \tag{25}$$

For a data set containing the gene expressions of $N_X = \sum_{c=1}^{C} N_c$ genes, we can obtain the log-likelihood of the mixture model as

$$\begin{aligned} \log L(\Theta|\boldsymbol{y}, \boldsymbol{u}_{c,s}) &= \log f_Y(\boldsymbol{y}_i|\boldsymbol{u}_{c,s};\Theta) \\ &= \sum_{i=1}^{N_X} \log \left[\sum_{c=1}^{C} \pi_c N(\mu_c(t_i), \boldsymbol{V}_{i,c}) \right]. \end{aligned} \tag{26}$$

3.3 Clustering using the EM-Algorithm

The EM algorithm will be employed in order to maximise the log-likelihood (26), in order to do this, we will need to add an indicator component z_{ic} defined as

$$z_{ic} = \begin{cases} 1 & \text{if subject } i \text{ belongs to cluster } c \\ 0 & \text{otherwise} \end{cases}.$$

We will be maximising the complete-data log-likelihood given by

$$L_{\mathscr{C}}(\Theta|\boldsymbol{y}, \boldsymbol{u}_{c,s}, Z) = \sum_{i=1}^{N_X} \left[\sum_{c=1}^{C} z_{ic} [\log \pi_c + \log N(\mu_c(t_i), \boldsymbol{V}_{i,c})] \right].$$

The EM algorithm is an iterative process that loops through two steps namely the E-step and the M-step.

3.3.1 The E-Step

The E-step, involves finding the expected complete log likelihood with respect to the parameters. It starts by setting the initial parameter values Θ. In the E-step, these initial parameter values are used to estimate the belonging/posterior probability that gene i falls into cluster c.

$$\begin{aligned}
\rho_{i,c}^{(r+1)} &= E(z_{ic}) \\
&= P(z_{ic}=1|\boldsymbol{y}_i,\boldsymbol{u}_{c,s},\theta_c) \\
&= P(c|\boldsymbol{y}_i,\boldsymbol{u}_{c,s},\theta_c) \\
&= \frac{P(c\ \boldsymbol{y}_i|\boldsymbol{u}_{c,s},\theta_c)}{P(\boldsymbol{y}_i|\boldsymbol{u}_{c,s},\theta_c)} \\
&= \frac{P(\boldsymbol{y}_i|c,\boldsymbol{u}_{c,s},\theta_c)P(c|\boldsymbol{u}_{c,s},\theta_c)}{P(\boldsymbol{y}_i|\boldsymbol{u}_{c,s},\theta_c)} \\
&= \frac{P(\boldsymbol{y}_i|c,\boldsymbol{u}_{c,s},\theta_c)P(c|\boldsymbol{u}_{c,s},\theta_c)}{\sum_{j=1}^{C}P(\boldsymbol{y}_i|j,\boldsymbol{u}_{c,s},\theta_c)P(j|\boldsymbol{u}_{c,s},\theta_c)} \\
&= \frac{\hat{\pi}_c^{(r)}N\left(\hat{\mu}_c^{(r)}(t_i),\hat{\boldsymbol{V}}_{i,c}^{(r)}\right)}{\sum_{j=1}^{C}\hat{\pi}_j^{(r)}N\left(\hat{\mu}_j^{(r)}(t_i),\hat{\boldsymbol{V}}_{i,j}^{(r)}\right)}
\end{aligned}, \tag{27}$$

where $\hat{\mu}_c^{(r)}(t_i)$ is the estimated average of the curve for the observations falling in cluster c evaluated at n_i times for the i^{th} subject, in addition $\hat{\boldsymbol{V}}_{i,c}^{(r)}$ is the $n_i \times n_i$ square matrix obtained from the current estimate of the covariance matrix $\hat{\boldsymbol{V}}_c^{(r)}$ for the particular cluster c.

By substituting z_{ic} with its expected value $E(z_{ic}) = \rho_{ic}$, we get,

$$\begin{aligned}
Q(\Theta,\Theta^{old}) &= E_Z L_{\mathscr{C}}(\Theta|\boldsymbol{y},Z) \\
&= \sum_{i=1}^{N_X}\sum_{c=1}^{C}\rho_{ic}[\log\pi_c + \log N(\mu_c(t_i),\boldsymbol{V}_{i,c})].
\end{aligned}$$

3.3.2 The M-Step

In the maximisation step, $Q(\Theta,\Theta^{old})$ is maximised with respect to the unknown parameters.

With the posterior probabilities obtained (27) we can estimate the cluster probabilities in the M-step, by maximising $Q(\Theta,\Theta^{old})$ with respect to π_c. Taking the constraint $\sum_{c=1}^{C}\pi_c = 1$ into consideration we need to apply the Lagrange multiplier, that means we maximise

$$Q(\Theta,\Theta^{old})^* = \sum_{i=1}^{N_X}\sum_{c=1}^{C}\rho_{ic}[\log\pi_c + \log N(\mu_c(t_i),\boldsymbol{V}_{i,c})] + \ell(\sum_{c=1}^{C}\pi_c - 1).$$

Differentiating $Q(\Theta, \Theta^{old})^*$ with respect to π_c and ℓ respectively and setting it equal to zero will result in

$$\frac{\delta Q^*}{\delta \pi_c} = \sum_{i=1}^{N_X} \frac{\rho_{ic}}{\pi_c} + \ell = 0 \tag{28}$$

$$\frac{\delta Q^*}{\delta \ell} = \sum_{i=1}^{N_X} \pi_c - 1 = 0. \tag{29}$$

Summing (28) over c and multiplying by π_c yields

$$\begin{aligned} \sum_{i=1}^{N_X} \sum_{c=1}^{C} \rho_{ic} + \ell \sum_{c=1}^{C} \pi_c &= 0 \\ n + \ell &= 0 \\ \ell &= -n, \end{aligned} \tag{30}$$

since $\sum_{i=1}^{N_X} \sum_{c=1}^{C} \rho_{ic} = n$ and $\sum_{c=1}^{C} \pi_c = 1$. By substituting (30) into (28), π_c is solved as

$$\begin{aligned} \frac{\delta Q}{\delta \pi_c} &= \sum_{i=1}^{N_X} \frac{\rho_{ic}}{\pi_c} - n = 0 \\ \hat{\pi}_c^{(r+1)} &= \frac{1}{n} \sum_{c=1}^{C} \rho_{i,c}^{(r+1)} \\ &= \frac{N_c}{n},\ c = 1, ..., C, \end{aligned} \tag{31}$$

where $n = \sum_{i=1}^{N_X} n_i$ and $N_c = \sum_{c=1}^{C} \rho_{i,c}$, in addition the M-step determines the estimates of $\hat{\mu}_c^{(r+1)}(t_i)$ and $\hat{\boldsymbol{V}}_c^{(r+1)}$ by fitting the model defined in (23) using $\hat{\boldsymbol{\rho}}_c^{(r+1)} = (\hat{\rho}_{1,c}, ..., \hat{\rho}_{N_X,c})$ as weights. These parameters are estimated using the REML in (21).

That is $\boldsymbol{\tau}$ and u are estimated using the same formulas as in equations (19) and (20), such that

$$\hat{\mu}_c^{(r+1)}(t_i) = \boldsymbol{X}_{i,s} \hat{\tau}_{c,s}^{(r+1)} + \boldsymbol{Z_{i,s}} \hat{\boldsymbol{u}}_{\boldsymbol{c,s}}^{(\boldsymbol{r+1})}. \tag{32}$$

In addition the variance components are estimated as

$$\hat{\sigma}_{\varepsilon,c}^{2(r+1)} = \frac{\boldsymbol{y_i^T y_i} - \hat{\boldsymbol{\tau}}_{\boldsymbol{c,s}}^{(\boldsymbol{r+1})} \boldsymbol{X_{i,s}^T y_i} - \hat{\boldsymbol{u}}_{\boldsymbol{c,s}}^{\boldsymbol{T(r+1)}} \boldsymbol{Z i, s^{T(r+1)} y_i}}{n - d},$$

and $\sigma_{\gamma,c}^2$ can be estimated as

$$\frac{\delta Q}{\delta \sigma^2_{\gamma,c}} = -\frac{1}{2}\left[\frac{n_i}{\sigma^2_{\gamma,c}} - \frac{\boldsymbol{X^T_{i,s}R^{-1}_i X_{i,s}} + \boldsymbol{Z^T_{i,s}R^{-1}_i Z_{i,s}} + \boldsymbol{G^{-1}_i}}{(\sigma^2_{\gamma,c})^2} - \frac{\hat{u}^T_{c,s}\hat{u}_{c,s}}{\sigma^2_\varepsilon(\sigma^2_{\gamma,c})^2}\right] = 0$$

$$\hat{\sigma}^2_{\gamma,c} = \frac{\boldsymbol{X^T_{i,s}R^{-1}_i X_{i,s}} + \boldsymbol{Z^T_{i,s}R^{-1}_i Z_{i,s}} + \boldsymbol{G^{-1}_i} + \hat{\sigma}^{-2}_\varepsilon \boldsymbol{\hat{u}^T_{c,s}\hat{u}_{c,s}}}{n_i},$$

such that

$$\hat{\boldsymbol{V}}^{(r+1)}_{i,c} = \hat{\sigma}^2_{\gamma,c}\boldsymbol{Z}_{i,\gamma}\boldsymbol{Z}^T_{i,\gamma} + \hat{\sigma}^2_{\varepsilon,c}\mathrm{I}_{n_i\times n_i}. \tag{33}$$

The process iterates between the E- and M-steps until convergence is reached. The EM algorithm is rather unpredictable for a dataset with multiple dimensions, therefore a revision of the regular EM algorithm known as rejection-controlled EM (RCEM) is used.

Ma et al. [26] define RCEM, which is used to accelerate and optimize the regular EM algorithm. In RCEM, the posterior probabilities (weights) for every gene within every cluster are calculated, as in (27). A sample of genes with posterior probability $\hat{\rho}_{i,c} \leq t$ where t is a user-specified threshold, is chosen with probability $\hat{\rho}_{i,c}/t$ for each cluster respectively. The chosen genes are then given new posterior probabilities equal to t, while the unselected genes are given new posterior probabilities equal to 0, such that we have updated belongings to be used in the M-step defined as

$$\hat{\rho}^*_{i,c} = \begin{cases} \hat{\rho}_{i,c} & \text{if } \hat{\rho}_{i,c} > t \\ t & \text{with probability } \hat{\rho}_{i,c}/t \text{ if } \hat{\rho}_{i,c} \leq t \\ 0 & \text{otherwise} \end{cases},$$

in addition we will then normalise $\hat{\rho}^*_{i,c}$ such that $\hat{\rho}^*_{i,c} = \frac{\hat{\rho}^*_{i,c}}{\sum_{c=1}^{C}\hat{\rho}^*_{i,c}}$

After obtaining these normalised belongings we continue with the M-step where Equation (3.3) is maximised with respect to the relevant parameters, using $\hat{\rho}^*_{i,c}$ as the probability of observation i belonging to cluster c. The greater the threshold value t ,the faster the EM-algorithm will reach convergence, but this will also result in the loss of precision. This algorithm will then be implemented from different starting points to ensure that the process does not converge to a local maximum, but to the global maximum for all parameters. By using the sampling method used in the RCEM algorithm, the log-likelihood does not increase uniformly as it does in regular EM. As a result, rather than iterating until convergence is reached, the RCEM algorithm iterates for a pre-defined number of times.

In cluster research, determining the number of clusters C is usually a challenge. As a result, suitable methods for determining an optimal value for C are required. The probability of an observation falling into a cluster increases as the number of parameters in the model increases. Over-fitting can occur as a result of this, in which case a model-selection criterion such as the Akaike Information Criterion (AIC) or the Bayesian Infor-

mation Criterion (BIC) can be employed to estimate the value of C.

The BIC also is known as the Schwarz Information Criterion (SIC) [27] is a criterion used to determine trade-off between model complexity and fit, it is also the criterion we will use in this study. It is in part based on the likelihood function of the respective models. The BIC evaluates the accuracy as well as the complexity of each model to identify the true model that fits the data. A lower value of BIC demonstrates that the model has a better fit to the data. The BIC is

$$BIC = -2\log(L) + d\log(n), \tag{34}$$

, where n is the sample size, $\log(L)$ is as in Equation (26) and d is the number of parameters to be estimated in the model. In our case we define d as $d = pC + 4C - 1$, where C is the number of clusters, we have $C-1$ mixing proportions $\pi_1, ...\pi_{C-1}$ and three variance components $(\sigma^2_{u,c}, \sigma^2_{\gamma,c}, \sigma^2_{\varepsilon,c})$ for each cluster.

Cluster analysis is primarily used as an exploratory tool in the analysis of gene expression data across time and evaluating the actual number of clusters is not the main objective. Even so, in some cases, this could be important and the topic is revisited in the discussion. The EM algorithm is given in Algorithm 1.

Algorithm 1 EM algorithm Clustering of mixture of penalized mixed effects model.

1: Assign genes at random to C clusters and use a penalized smoothing spline to match the data in each cluster using Equation (23).
2: Assign initial parameter values for $\hat{\mu}_c^{(0)}, \hat{\pi}_c^{(0)}$ and $\hat{\boldsymbol{V}}_c^{(0)}$.
3: ***E-Step***
4: Determine the posterior probabilities $\hat{\rho}_{i,c}$ using the current estimates of $\hat{\mu}_c^{(r)}, \hat{\pi}_c^{(r)}$ and $\hat{\boldsymbol{V}}_c^{(r)}$ for each gene $c = 1, ..., C$ using Equation (27).
5: For the genes in each cluster with posterior probability $\hat{\rho}_{i,c} \leq t$ assign to that gene the posterior probability of t with probability $\hat{\rho}_{i,c}/t$ and 0 with probability $1 - \hat{\rho}_{i,c}/t$.
6: ***M-Step***
7: We fit the penalized smoothing spline model again using the normalised posterior probabilities determined in the E-step, this time based on changes in cluster membership as shown by shifts in cluster probabilities.
8: Update parameter estimates of $\hat{\mu}_c^{(r+1)}, \hat{\pi}_c^{(r+1)}$ and $\hat{\boldsymbol{V}}_c^{(r+1)}$ for each gene $c = 1, ..., C$ using equations (19,20,31,33)
9: Set $\Theta^{old} = \Theta^{new}$, where Θ^{new} contains the values of the updated parameters.
10: Iterate between E- and M-steps for a number of steps that we specify.
11: Steps $1-9$ must be repeated with different starting points to prevent convergence to a local limit and to choose the solution with the greatest log-likelihood value.
12: Steps $1-10$ are repeated with different C values and the number of clusters with the smallest BIC value is chosen.

3.4 Conclusion

In this section a mixture model consisting of mixed effects model components is considered. The P-splines mixed effects model explored in Equation (12) has been extended to include a random component that captures subject based clusters with subject-specific variance around the cluster mean. By combining data from individual genes and by applying this model in a mixed model framework enables one to estimate the cluster mean expression and gene specific expression profiles. The EM-algorithm is a powerful and versatile method

that is often used in density estimation with missing data, such as clustering algorithms like. This algorithm has been applied to the mixture model to estimate the values of the unknown parameters that would maximise the model's likelihood. A restriction has been applied to the expectation maximisation method, such that the rate to convergence is severely decreased. By applying REML the computational expensiveness of the model is reduced and convergence of the model is reached faster. The next step in this study would be to evaluate the performance of this model on a real or simulated dataset .

4 Simulation study

4.1 Introduction

Simulation studies are computer experiments that use pseudo-random sampling to generate data with the goal of analyzing the data. The capacity to comprehend the behaviour of statistical approaches is a significant strength of simulation studies since some parameters of interest are known from the process of creating the data. This enables us to consider technique characteristics such as bias and variance. Among other scenarios, a simulation study is useful if theoretical arguments are insufficient to determine if the method of interest is valid in a specific practical applications or whether infringements of the assumptions underlying the available theory affect the accuracy of the study. Simulations, like experiments in science, play an important part in methodological research [28]. In this section data is simulated to fit the proposed model to this data, the aim of this simulation study is to test whether a mixture of penalized mixed effects models can accurately cluster and capture the trend of the data in each respective cluster. Data will be simulated from time dependent models that captures different trends over time.

4.2 Simulation Scenario

A short simulation study was conducted to evaluate the performance of the proposed technique. To demonstrate this, data from 3 different models have been simulated, such that $C = 3$. The models from which the data has been simulated are as follows,

$$
\begin{aligned}
y_{ij}^{(1)} &= 0 + \gamma_i + \varepsilon_{ij}, \\
y_{ij}^{(2)} &= 5^{t_{ij}} + 10 + \gamma_i + \varepsilon_{ij}, \\
y_{ij}^{(3)} &= -exp(t_{ij}) - 7 + \gamma_i + \varepsilon_{ij},
\end{aligned}
$$

where it is assumed that $\gamma_i \sim N(0, \sigma_\gamma^2)$ where σ_γ^2 is in the range 0.3 to 1 such that the initial parameters of σ_γ^2 for each cluster respectively is $(0.75, 0.45$ and $0.3)$, thus combining the basic model description with a variety of parameter choices to mimic numerous variation levels across genes within a cluster. For these simulations it is assumed that $\varepsilon_{ij} \sim N(0, \sigma_\varepsilon^2)$ where σ_ε ranges from 0.5 to 1.3 and has been chosen as $(1.2,\ 0.96$ and $1.25)$ for each cluster respectively, such that various degrees of error is generated in the data.

The aim of this study is to demonstrate the performance of the proposed model, a mixed model with P-splines in clustering time-course data.

For this study a hundred datasets were simulated and the model has been fitted to each of these datasets. For every dataset, the data has been simulated such that cluster sizes vary. The cluster sizes were chosen to be $(N_1 = 100, N_2 = 70\ , N_3 = 60)$, resulting in a total of $N_T = 230$ simulated expression curves, measured over

$n = n_i = 14$ equally spaced time points in $[0, 1.2]$. The dataset contains a total of 3220 observations. The mixing proportions denoted by π_c for each cluster have been initialised such that all mixing components have equal mixing probabilities, that is $\pi_c = \left(\frac{1}{3}, \frac{1}{3}, \frac{1}{3}\right)$ for $c = 1, 2, 3$.

The various simulated clusters for a single dataset (one simulation iteration) which has been simulated and which will serve as an example for this study, are plotted in Figure 7.

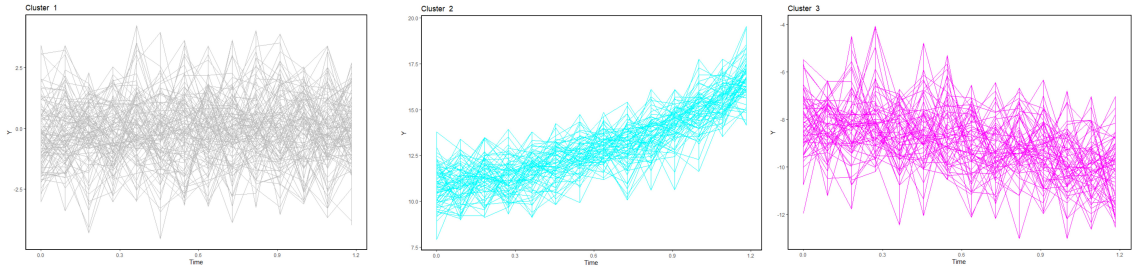

Figure 7: Plots of simulated data from the above mentioned models, for different numbers of simulated observations per cluster. The data has been simulated on 10 equally spaced time points.

The generated data is plotted in Figure 8. The proposed clustering technique has been applied to this dataset. For the EM-algorithm the initial values were chosen as follows. First a random sample of size $\frac{N_T}{C}$ has been drawn from the dataset. The initial parameter values for $\boldsymbol{\tau}$, $\boldsymbol{u}$, σ_ε^2 and σ_γ^2 were set such that $\boldsymbol{\tau}$, and u are random values obtained from a $Unif(-5, 5)$ distribution, this range has been carefully evaluated and has been established to be an accurate range for the parameters of the model for the simulated data. The variance components σ_ε^2 and σ_γ^2 has been chosen as in paragraph 1 on page 33. The design matrices, $\boldsymbol{X}$ and $\boldsymbol{Z}$ were calculated according to the formulas in Equation (11). The penalty parameter λ has been assigned a fixed value of 1.5, which has been determined using GCV, and has been rounded to the nearest decimal for simplicity. The threshold value for the RCEM algorithm has been set to be $t = 0.1$.

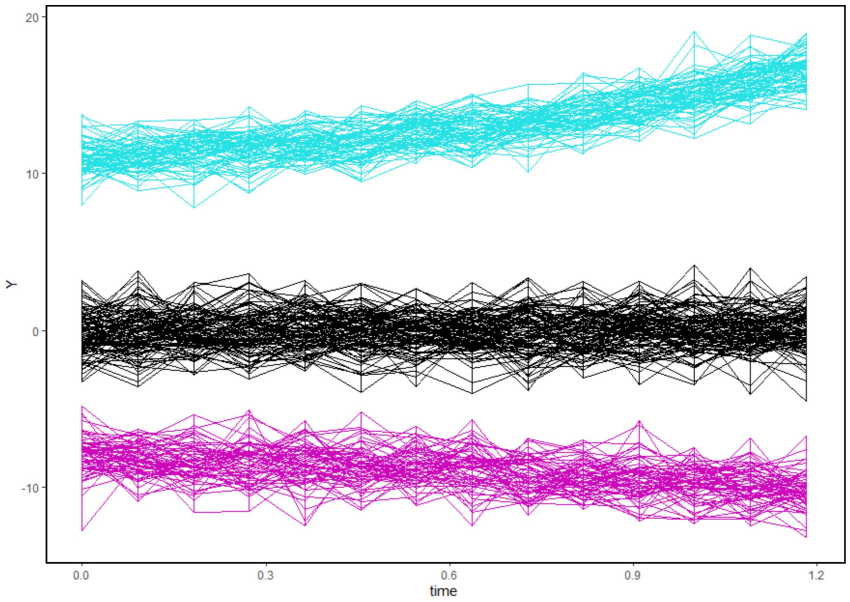

Figure 8: A plot of a full simulated dataset, simulated from the 3 model components. Where all the observations measured at 10 distinct time points are plotted on a single plot.

To reduce the number of possible combinations, certain hyper parameters are fixed such as the degree of the basis functions, which is chosen to be $m = 3$, the order of the differencing has been set to $d = 2$ and the number of knots have been chosen to be $K = 7$. Hence the model consists of $K + m + 1 = 11$ basis functions.

4.3 Goodness-of-fit

4.3.1 Accuracy

In order to evaluate the accuracy of the clustering model, a confusion matrix was used to determine how many of the observations were clustered correctly. It should be noted that since the EM-Algorithm does not assign the observations to the same cluster order as labeled in Figure 7, the predicted cluster numbers had to be refactored to match the initial cluster labeling. This has been done by obtaining the mean of each predicted cluster and measuring it against the means of the actual clusters. The predicted clusters were relabeled according to the real clusters which proved to have the smallest difference from the predicted cluster means.

The confusion matrix obtained is

Prediction \ Reference	1	2	3
1	1398	0	2
2	0	980	0
3	2	0	838

Table 1: Confusion matrix of predicted cluster belonging, obtained from the results of the EM algorithm

.

From the confusion matrix, the accuracy of the clustering model is calculated as

$$accuracy = \frac{\text{correct predictions}}{\text{all predictions}}$$

which is the sum of the diagonal values in the confusion matrix divided by all the values in the matrix. The accuracy obtained for this dataset is 0.998. The average accuracy is obtained to be 0.915. A minimum accuracy of 0.652 has been observed and a maximum accuracy of 100% has been obtained.

A box-plot of the recorded accuracies for each simulated dataset is plotted in Figure 9.

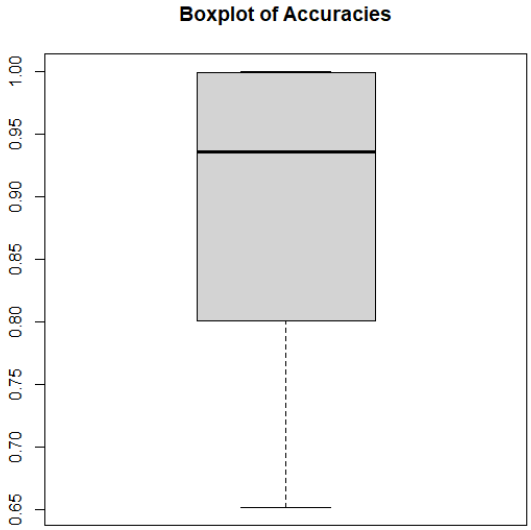

Figure 9: Box-plot of the 100 calculated model-accuracy for every simulated dataset.

4.3.2 Coefficient of Determination (R^2)

An additional measure of model performance that has been evaluated in this study is the coefficient of determination to measure how close the simulated data is to the fitted regression line.

The formula for calculating R^2 for simple linear regression is

$$R^2 = 1 - \frac{SSE}{SST} = 1 - \frac{\sum_{i=1}^{N}(\boldsymbol{y}_i - \hat{\boldsymbol{y}}_i)^2}{\sum_{i=1}^{N}(\boldsymbol{y}_i - \bar{\boldsymbol{y}})^2}$$

Where $\hat{\boldsymbol{y}}_i$ is the predicted values obtained from fitting the model to the simulated model and $\bar{\boldsymbol{y}}$ is the sample average of the simulated data.

In this research we will be evaluating the R^2 using 2 distinct methods. The first method evaluates the hard clustered results obtained from the simulation experiment. Essentially the data is divided into their distinct clusters. The estimated parameters for each cluster can be used to calculated the coefficient of determination

for each cluster respectively.

$$R_1^2 = 1 - \frac{\sum_{i=1}^{N_1}\left(\boldsymbol{y}_i - \hat{\boldsymbol{y}}_i^{(1)}\right)^2}{\sum_{i=1}^{N_1}\left(\boldsymbol{y}_i^{(1)} - \bar{\boldsymbol{y}}^{(1)}\right)^2} = 0.798,$$

$$R_2^2 = 1 - \frac{\sum_{i=1}^{N_2}\left(\boldsymbol{y}_i - \hat{\boldsymbol{y}}_i^{(2)}\right)^2}{\sum_{i=1}^{N_2}\left(\boldsymbol{y}_i^{(2)} - \bar{\boldsymbol{y}}^{(2)}\right)^2} = 0.856,$$

$$R_3^2 = 1 - \frac{\sum_{i=1}^{N_3}\left(\boldsymbol{y}_i - \hat{\boldsymbol{y}}_i^{(3)}\right)^2}{\sum_{i=1}^{N_3}\left(\boldsymbol{y}_i^{(3)} - \bar{\boldsymbol{y}}^{(3)}\right)^2} = 0.824$$

,

inferring that 79.8% of the variation in cluster 1 is explained by component 1 of the model, 85.6% of the variation in cluster 2 is explained by component 2 of the model and 82.4% of the variation in cluster 3 is explained by component 3 of the model.The average R^2 obtained across all the simulated datasets for each cluster respectively is 0.841, 0.883, and 0.862 for each cluster respectively. This value indicated that the simulated data and the fitted model is severely correlated.

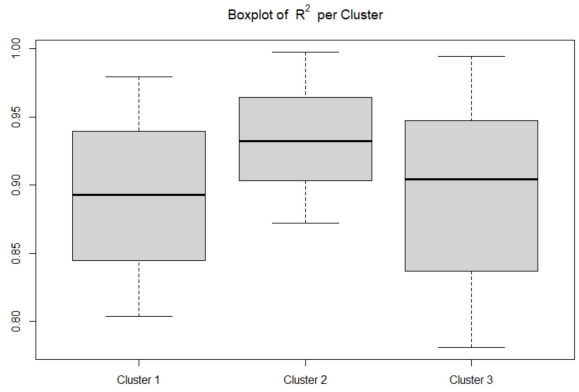

Figure 10: A box-plot indicating the spread of the coefficient of determination R^2 obtained for every estimated cluster for every data simulation .

A box-plot indicating the spread of the R^2 values, obtained from all simulated datasets is plotted in Figure 10. From this figure it is evident that the correlation between the actual and predicted clusters are high, indicating that a fair amount of the variation in the clusters is explained by each component in the model.

The second approach we will be looking at is a weighted coefficient of determination where each observation's residuals for each component of the model are weighted with the respective cluster responsibilities.

$$R^2_{weighted} = 1 - \frac{\sum_{i=1}^{N}\left[\hat{\pi}_1\left(\boldsymbol{y}_i - \hat{\boldsymbol{y}}_i^{(1)}\right)^2 + \hat{\pi}_2\left(\boldsymbol{y}_i - \hat{\boldsymbol{y}}_i^{(2)}\right)^2 + \hat{\pi}_3\left(\boldsymbol{y}_i - \hat{\boldsymbol{y}}_i^{(3)}\right)^2\right]}{\sum_{i=1}^{N}\left(\boldsymbol{y}_i - \bar{\boldsymbol{y}}_{weighted}\right)^2},$$

where $\bar{\boldsymbol{y}}_{weighted} = \frac{\sum_{i=1}^{N}(\hat{\pi}_1\boldsymbol{y}_i + \hat{\pi}_2\boldsymbol{y}_i + \hat{\pi}_3\boldsymbol{y}_i)}{N} = \frac{\sum_{i=1}^{N}\boldsymbol{y}_i}{N} = \bar{\boldsymbol{y}}$, since $\sum_c^3 \hat{\pi}_c = 1$. Using this method, this weighted R^2 indicates that 78.6% of the total variation in the data is explained by the 3 component mixture of mixed effects model. This is a relatively high value for R^2 indicating that the model is a adequate choice for modelling the data and explaining the variation in the data.

4.4 Parameter Estimates

The parameter values obtained for each cluster respectively is as follows,

Table 2: Parameter estimates for each cluster, obtained using the EM-algorithm.

Parameter \ Cluster	1	2	3
$\hat{\pi_c}$	0.434	0.304	0.261
$\hat{\tau_{1c}}$	0.061	10.415	-7.822
$\hat{\tau_{2c}}$	-0.111	4.942	-2.074
$\hat{u_{1c}}$	-0.149	-0.014	-0.260
$\hat{u_{2c}}$	0.152	0.252	0.111
$\hat{u_{3c}}$	0.031	-0.191	0.370
$\hat{u_{4c}}$	0.003	0.061	-0.222
$\hat{u_{5c}}$	-0.157	0.338	-0.308
$\hat{u_{6c}}$	0.288	0.097	0.264
$\hat{u_{7c}}$	-0.355	0.380	-0.143
$\hat{u_{8c}}$	0.029	0.189	-0.351
$\hat{u_{9c}}$	-0.023	-0.198	-0.298
$\hat{\sigma^2_\gamma}$	0.888	0.383	0.874
$\hat{\sigma^2_\varepsilon}$	1.759	0.759	1.731

From the results in Table 2 it is clear that the mixing proportions π_c coincide with the number of observations simulated in each cluster, with the majority of the observations falling into cluster 1, the second most observations in cluster 2 and the least observations in cluster 3.

To demonstrate the performance of the model,the estimated mean for each cluster has been plotted against the simulated observations.

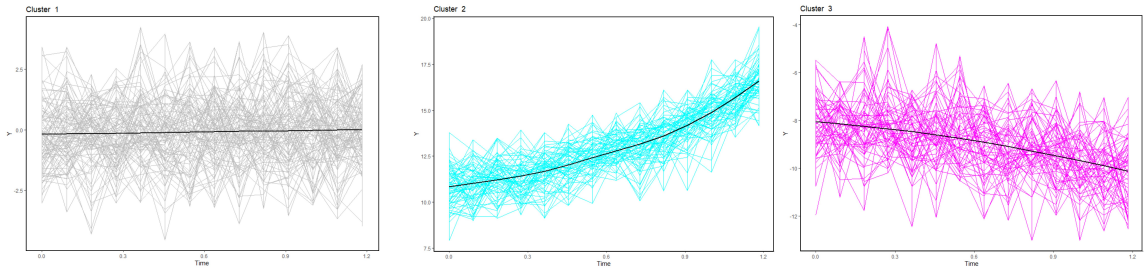

Figure 11: Plots of simulated data, for different numbers of simulated observations per cluster. The data has been simulated on 10 equally spaced time points. The estimated mean for each cluster has been plotted(black line) to demonstrate the efficiency of the model.

As can be seen from the graphs in Figure 11, the EM-algorithm performed remarkably in clustering the simulated data into the correct clusters. The performance of the model is evident due to the fact that the estimated means exhibit the same trend as that of the simulated data. It can thus be concluded that the suggested model can be applied to real data to cluster longitudinal data.

4.5 Conclusion

In this section the accuracy and the fit of the model has been evaluated on simulated data. The data in this section has been simulated from three different time dependent functions, that are able to capture various trends over a specified time interval. The data from the respective models have been added in a single data frame. The proposed model has been fitted to the data, clustering has been perform by means of the EM-algorithm as discussed in the previous section. By making use of goodness-of-fit tests, the accuracy of the model has been evaluated. It has been found that the model performed outstandingly since the model obtained 91% accuracy score. The correlation between the simulated data and the predicted values of the model has also proven to be high, inferring that the model performed adequately in capturing the trend of the simulated data.

5 Clustering of time-course influenza data using mixtures of Penalized mixed effects models.

5.1 Background

To analyse the the proposed techniques on real life data, the suggested models are applied to data from a micro-array time-course experiment, where mice have been exposed to different strains of influenza. Lung tissue has been collected from these specimen during 14 time points after infection (0, 3, 6, 9, 12, 18, 24, 30 ,36, 48, 60, 72, 120 and 168 hours) [29].

The strains of influenza that the mice have been exposed to and which is analysed in the study are a low pathogenic seasonal H1N1 influenza virus (A/Kawasaki/UTK4/2009 [H1N1]), a mildly pathogenic virus from the 2009 pandemic season (A/California/04/2009 [H1N1]) and a highly pathogenic H5N1 avian influenza virus (A/Vietnam/1203/2004 [H5N1]). Mice were injected with 105 PFU of each virus. An additional 42 mice were injected with a lower dose of the Vietnam avian influenza virus (103 PFU).

The authors show that the inflammatory response of lung tissue is controlled until a viral concentration threshold is exceeded in the lung by integrating gene expression time-course data with pathogen growth data. When this threshold is reached, a significant inflammatory and cytokine response follows. These findings show that viral concentration has a non-linear effect on the pathological response.

5.2 Application of the model onto real influenza mouse data

Before commencing the analysis, the data has been normalized and quality control checks were performed using the techniques as proposed in [30][31], for which the code is available on GitHub [32]. The dataset contains the information of 39544 observed genes, the times which these observations were taken,

For the sake of reducing the computational expensive nature of this large dataset, only the first replica of each observation has been taken into account, as well as only specimen that has been exposed to the mildly pathogenic virus from the highly pathogenic H5N1 avian influenza virus (A/Vietnam/1203/2004 [H5N1]) has been considered as the data for this study.

Therefore taking the lung tissue of 1000 random specimen collected at 14 time points, the model is applied to the data in order to determine the number of distinct clusters, as well as which samples fall into the same gene cluster.

Similar to the simulation study, parameters have been initialized in order to reduce the computational power of this model on the data. The number of knots has been chosen to be equal to half that of the number of time points, that is $K = 7$. The degree of the basis functions has been fixed to be $m = 3$, the order of differencing has been set $d = 2$. This results in a total of $K + m + 1 = 11$ basis functions. Additionally, the value of the smoothing parameter λ is calculated as 2.37 using GCV. The initial mixing proportions has been set to be

$\pi_c = 1/C$ for $c = 1, ..., C$, where C is the number of clusters in the model. Random initial values were chosen for the parameters τ and u, as well as for the variance components σ_ε^2 and σ_u^2. The design matrices, $\boldsymbol{X}$ and $\boldsymbol{Z}$ were calculated according to the formulas in Equation (11). The threshold value for the RCEM algorithm has again been set to be $t = 0.1$.

In order to determine the correct number of distinct clusters present within the data, the BIC is used and measured for different numbers of clusters. The number of clusters tested on the data ranges between 2 and 25. The BIC obtained for the different number of clusters tested are given in the Table 3.

Clusters	Parameters	Log-Likelihood	BIC
2	14	-3837.997	7791.093
5	35	-3067.407	6435.840
8	56	-2923.425	6333.802
9	63	-3096.366	6741.660
10	70	-2832.083	6275.069
11	77	-3096.740	6866.359
12	84	-2760.655	6256.165
13	91	-2765.440	6327.710
14	98	-2808.413	6475.633
15	105	-2774.836	6470.454
20	140	-2653.924	6538.507
25	175	-2642.357	6825.252

Table 3: Table of BIC values obtained for different number of clusters tested on real data.

From the BIC values obtained in Table 3, it is evident that the number of clusters providing the minimum BIC value is for 12 clusters. This indicates that the optimal number of clusters for the real dataset would be 12 clusters.

In order to demonstrate the clustering technique with the proposed model, the real data observations have been clustered into 12 distinct clusters, by means of the EM algorithm as proposed in Section 3. The performance of the model has been displayed by plotting the real data into their respective clusters, as allocated by the EM algorithm. The observed gene expressions Y has been plotted against the covariate $Time$.

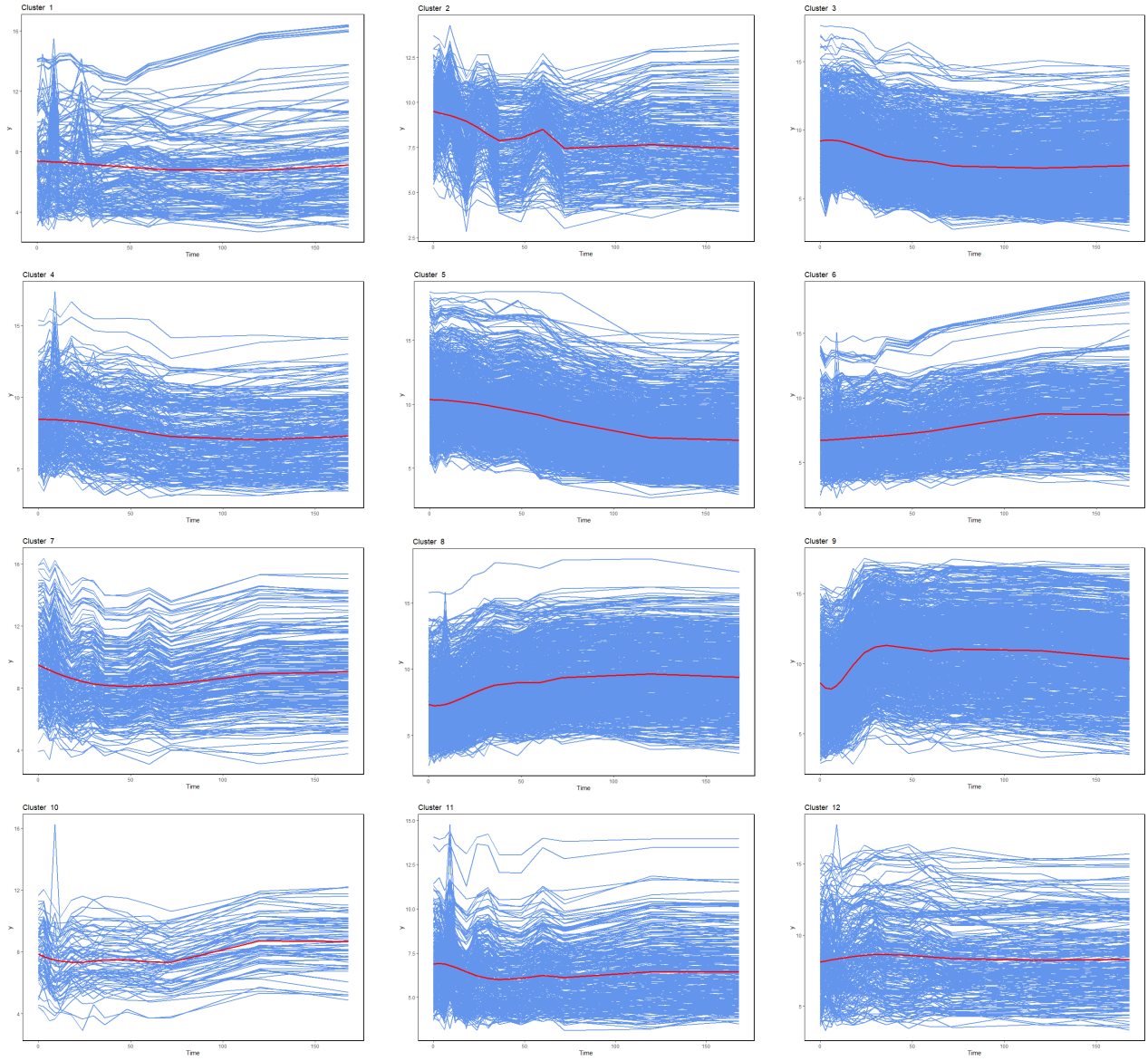

Figure 12: Clustering results on real influenza micro-array data, obtained by fitting the proposed model to the data and using the EM-algorithm to perform clustering. Each gene (blue lines) is plotted in it's respective cluster. The cluster mean(red line) is included in the plot of each cluster's data.

According to the plots displayed in Figure 12 it can be seen that the EM-algorithm clearly distinguished between the different clusters present in the data and that it estimated the cluster mean quite accurately. As can be seen from these plots, a larger amount of variation is evident in the earlier time points than in the later time points, this is due to the fact that more observations were taken at the earlier time points. It is clear that the clusters display a clear trend over time, especially for earlier time points where samples have been recorded in closer consecutive times than for later time points. For later time points where samples have been taken at larger time intervals the mean seems to even out and appear more constant over time. But it should be noted that despite the fact that the means for each distinct cluster seem to be converge over time, the means are different for each cluster indicating a clear variation between clusters. Therefore the model has the ability to

distinguish between within and between cluster variation.

Firstly, the parameters of the model which provided the best fit to the data has been obtained and the parameters for each cluster is displayed in the Table 4.

Parameter \ Cluster	1	2	3	4	5	6	7	8	9	10	11	12
$\hat{\pi}_c$	0.033	0.052	0.153	0.052	0.151	0.095	0.041	0.144	0.153	0.014	0.061	0.049
$\hat{\tau}_{1c}$	6.874	9.240	-7.357	7.905	10.255	6.753	10.698	25.175	31.620	7.343	7.685	7.968
$\hat{\tau}_{2c}$	0.005	-0.007	0.906	-0.001	-0.015	0.010	-0.011	-0.998	-1.420	0.007	-0.006	0.002
$\hat{u}_{1c}$	-2.979	-1.795	11.747	-4.266	-3.847	1.101	9.610	-17.067	-42.002	0.181	5.748	-0.996
$\hat{u}_{2c}$	1.058	4.181	-4.134	2.638	4.103	-1.742	-0.567	5.840	11.940	-0.417	-0.532	-1.058
$\hat{u}_{3c}$	-0.748	-6.510	10.556	-2.844	-4.796	2.611	-1.628	-12.271	-19.434	-0.011	-1.277	0.366
$\hat{u}_{4c}$	4.252	11.197	-55.908	5.586	7.947	-4.566	5.923	62.557	92.102	0.557	6.397	0.103
$\hat{u}_{5c}$	-5.587	-9.246	1300.796	-5.445	-7.391	4.682	-5.858	-1440.727	-2072.201	0.501	-7.252	0.181
$\hat{u}_{6c}$	-0.737	-3.098	-3066.846	-1.036	-2.398	0.728	-1.276	3395.585	4879.649	-0.178	-1.503	0.010
$\hat{u}_{7c}$	2.491	4.547	1575.283	2.507	4.174	-2.321	2.318	-1744.071	-2505.743	-0.538	3.111	-0.093
$\hat{u}_{8c}$	1.462	2.754	1050.601	1.486	2.507	-1.364	1.389	-1163.175	-1671.205	-0.300	1.854	-0.054
$\hat{u}_{9c}$	0.244	0.459	175.094	0.248	0.418	-0.227	0.232	-193.852	-278.516	-0.050	0.309	-0.009
$\hat{\sigma}^2_\gamma$	4.827	2.291	3.669	3.472	4.351	3.594	3.489	3.916	5.228	1.866	2.104	4.784
$\hat{\sigma}^2_\varepsilon$	2.411	1.144	1.833	1.734	2.173	1.795	1.742	1.956	2.611	0.932	1.051	2.390

Table 4: Parameter estimates for each cluster, obtained using the EM-algorithm. The parameters are obtained for the model fitted onto real influenza data. For which 12 clusters have been identified. The parameter estimates of the 12 clusters are displayed in this table.

From the results portrayed in Table 4 it can be seen that the parameter values for clusters 3,8 and 9 are severely larger in scale for parameters u_{ic} than for any of the other clusters.

For this clustering algorithm, the i^{th} gene expression in cluster c is distributed as a function conditional on the smoothing random effect $\boldsymbol{u}_{c,s}$

$$\boldsymbol{y}_i|\boldsymbol{u}_{c,s} \sim N(\boldsymbol{X}_{i,s}\boldsymbol{\tau}_{c,s} + \boldsymbol{Z}_{i,s}\boldsymbol{u}_{c,s}, \boldsymbol{V}_{i,c}),$$

where $\boldsymbol{V}_{i,c} = \sigma^2_{\gamma,c}\boldsymbol{Z}_{i,\gamma}\boldsymbol{Z}^T_{i,\gamma} + \sigma^2_{\varepsilon,c}\mathrm{I}_{n_i \times n_i}$ is the corresponding $n_i \times n_i$ square matrix of the square covariance matrix $\boldsymbol{V}_c = \sigma^2_{\gamma,c}\boldsymbol{Z}_{c,\gamma}\boldsymbol{Z}^T_{c,\gamma} + \sigma^2_{\varepsilon,c}\mathrm{I}$ for cluster c.

Where the design matrices $\boldsymbol{X}_{i,s}$ and $\boldsymbol{Z}_{i,s}$ are calculated for each gene respectively, using the methods as discussed in section 2.4. By replacing the τ_c, u_c, σ^2_γ and σ^2_ε parameters by the estimated $\hat{\tau}_c$ and u_c in Table 4 into the formula one can estimate the distribution of each gene in each cluster respectively.

The overall model to estimate the gene expression curve for the i^{th} gene in cluster c is given as

$$f_Y(\boldsymbol{y}_i; \hat{\Theta}) = \sum_{c=1}^{12} \hat{\pi}_c f_c(\boldsymbol{y}_i, \hat{\theta}_c),$$

which represents a mixture model where $f_c(.)$ represents the densities of components that depend, on the vector of estimated parameters $\hat{\theta}_c = (\hat{\boldsymbol{\tau}}_{c,s}, \hat{\sigma}^2_{u,c}, \hat{\sigma}^2_{\gamma,c}, \hat{\sigma}^2_{\varepsilon,c})$, $\hat{\Theta} = (\hat{\theta}^T_1, ..., \hat{\theta}^T_{12}, \hat{\pi}_1, ..., \hat{\pi_c})^T$ and $\hat{\pi}_1, ..., \hat{\pi_{12}}$ are the mixing ratios that were estimated.

The smooth mean curve in each cluster c is then estimated based on the random effects $\boldsymbol{u}_{c,s}$

$$\begin{aligned} f_Y(\boldsymbol{y}_i|\boldsymbol{u}_{c,s};\hat{\Theta}) &= \sum_{c=1}^{12} \hat{\pi}_c N(\hat{\mu}_c(t_i), \hat{\boldsymbol{V}}_{i,c}) \\ &= \sum_{c=1}^{12} \pi_c N(\boldsymbol{X}_{i,s}\hat{\boldsymbol{\tau}}_{c,s} + \boldsymbol{Z}_{i,s}\hat{\boldsymbol{u}}_{c,s}, \hat{\boldsymbol{V}}_{i,c}) \end{aligned}.$$

The smooth mean curve of each cluster is represented by the red lines in Figure 12. In this model the dependent parameter $\boldsymbol{y}_{i,c}$ represents the i^{th} gene expression in cluster c.

5.3 Conclusion

In this section the application of mixtures of penalized mixed effects models on the clustering of time-course influenza data has been explored. In this analysis lung tissue gene samples collected over time, from mice exposed to strains of influenza has been used. By using the proposed model and implementing the EM-algorithm as in Section 3, clustering has been performed on the data. The clustering results for the data has been depicted in the form of graphs, where genes that have been clustered together were plotted on the same plot, the clustering mean of each cluster has been plotted along with the genes in their respective cluster plots. The parameter estimates for each cluster has also been explored. The results from this study serves as proof of the model's ability.

6 Discussion

Clustering techniques are fundamental in reducing the dimensionality of high-dimensional micro-array data. A challenge that arises in many studies is that the number and structure of the time-course gene expression distributions are unknown in most longitudinal genetic studies. As a result, a semi-parametric clustering technique has been explored in order to generate a statistically meaningful collection of curves and clusters.

This mini dissertation has researched an alternative method for clustering time-series data that employs the linear mixed effects model representation of penalized B-splines to cluster gene-expression patterns . By implementing smoothing spline estimates with P-spline smoothing the computational load involved with clustering is severely reduced. Furthermore, it makes it easier to describe the problem as a linear mixed effects model and, as a result, to incorporate mixtures of mixed effects models to smooth and cluster the data at the same time. In each cluster, mixed effects models allow for the introduction of gene-specific variation around the mean curve. This suggests that different genes from the same cluster are used to estimate model parameters to determine the cluster means as well as the cluster standard deviation. In this study clustering was done using the RCEM algorithm, which also reduces the computational time required to fit the model.

The suggested model has been implemented on 100 simulated datasets, for every dataset the model has been fitted and the accuracy as well as the coefficient of determination has been calculated. In this study it was found that, the model performed efficiently in fitting and clustering longitudinal data with high accuracy. The average accuracy obtained from the model for fitting and clustering the data has been calculated to be approximately 92%. Along with the accuracy measure the coefficient of determination has been used as a measure of model performance. The average coefficient of determination has been calculated to be 86.28%. From this value it is shown that there is high correlation between the predicted model and the actual data, indicating that the model is efficient in determining the trend of the longitudinal data.

The model has been implemented on real gene expression data of mice that have been exposed to strains of the influenza virus, for which lung tissue data has been observed over 14 time points. Clustering has been performed on the model and the model has accurately predicted that 12 distinct clusters are present in the data sample.

Although this model is an alteration and extension of the model discussed by [26], it is believed that this model has the ability to perform more adequately, due to the extension of the model by including a subject-specific variance around the cluster mean to represent subject based clusters .

An area in this research that might require further investigation is in choosing the number of clustering components. The BIC is a typical measure in determining the number of clusters, but according to [33], the BIC has a tendency to overestimate the number of clusters in a dataset. Multiple components, for example, might reflect non-normality of errors inside clusters or other characteristics of model inadequacy, causing BIC to choose a greater estimated number of clusters than the real number of clusters, thus using other model

selection approaches, such as the Integrated Completed Likelihood (ICL) proposed by [34], may lead to more accurate results.

Acknowledgement

This study was conducted at the University of Pretoria. This work is based upon research supported by the National Research Foundation, South Africa (Research chair: Computational and Methodological Statistics, Grant number 71199)(SARChI). Opinions expressed and conclusions arrived at are those of the author and are not necessarily to be attributed to the NRF. The author would like to thank their supervisors Dr. S. Millard and Dr. F. Kanfer for their continuous support comments and suggestions during the conducting of this research.

Data for this study was obtained from the *R* package *moanin* which contains the normalized data and metadata of [29] NAS124/2019.

References

[1] Martin P. Boer. A fast Mixed Model B-splines algorithm. *Biometris WUR, Wageningen, The Netherlands*, February 2015.

[2] Teuvo Kohonen. *Self-Organizing Maps*. Springer Science & Business Media, December 2012.

[3] J. A. Hartigan and M. A. Wong. Algorithm AS 136: A K-Means Clustering Algorithm. *Journal of the Royal Statistical Society. Series C (Applied Statistics)*, 28(1):100–108, 1979. Publisher: [Wiley, Royal Statistical Society].

[4] M. B. Eisen, P. T. Spellman, P. O. Brown, and D. Botstein. Cluster analysis and display of genome-wide expression patterns. *Proceedings of the National Academy of Sciences of the United States of America*, 95(25):14863–14868, December 1998.

[5] Chris Fraley and Adrian E Raftery. Model-Based Clustering, Discriminant Analysis, and Density Estimation. *Journal of the American Statistical Association*, 97(458):611–631, June 2002. Publisher: Taylor & Francis.

[6] Marco F. Ramoni, Paola Sebastiani, and Isaac S. Kohane. Cluster analysis of gene expression dynamics. *Proceedings of the National Academy of Sciences of the United States of America*, 99(14):9121–9126, July 2002.

[7] Y. Luan and H. Li. Clustering of time-course gene expression data using a mixed-effects model with B-splines. *Bioinformatics*, 19(4):474–482, March 2003.

[8] 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*, 39(1):1–38, 1977.

[9] Joon Jin Song, Ho-Jin Lee, Jeffrey S. Morris, and Sanghoon Kang. Clustering of time-course gene expression data using functional data analysis. *Computational Biology and Chemistry*, 31(4):265–274, August 2007.

[10] Gideon Schwarz. Estimating the Dimension of a Model. *Annals of Statistics*, 6(2):461–464, March 1978. Publisher: Institute of Mathematical Statistics.

[11] Paul D. McNicholas and Sanjeena Subedi. Clustering gene expression time course data using mixtures of multivariate t-distributions. *Journal of Statistical Planning and Inference*, 142(5):1114–1127, May 2012.

[12] L. Wang, G. Chen, and H. Li. Group SCAD regression analysis for microarray time course gene expression data. *Bioinformatics*, 23(12):1486–1494, June 2007.

[13] Yujing Zeng and Javier Garcia-Frias. A novel HMM-based clustering algorithm for the analysis of gene expression time-course data. *Computational Statistics & Data Analysis*, 50(9):2472–2494, May 2006.

[14] Trevor Hastie, Robert Tibshirani, Jerome Friedman, and James Franklin. The Elements of Statistical Learning: Data Mining, Inference, and Prediction. *Math. Intell.*, 27:83–85, November 2004.

[15] Alfred Haar. Zur Theorie der orthogonalen Funktionensysteme. *Mathematische Annalen*, 69(3):331–371, 1910.

[16] Carl R. de Boor. Subroutine package for calculating with b-splines. January 1971.

[17] Paul Gustav Heinrich Bachmann. *Die analytische Zahlentheorie.: Dargestellt von Paul Bachmann.* B.G. Teubner, Leipzig, 1894.

[18] Paul H. C. Eilers and Brian D. Marx. Flexible smoothing with B-splines and penalties. *Statistical Science*, 11(2):89–121, May 1996. Publisher: Institute of Mathematical Statistics.

[19] David Ruppert. Selecting the number of knots for penalized splines. *Journal of Computational and Graphical Statistics*, 11(4):735–757, 2002.

[20] Paul H. C. Eilers and Brian D. Marx. *Splines, Knots, and Penalties.* 2004.

[21] Lou Xiao, Cai Li, Will Checkley, and Ciprian Crainiceanu. Fast covariance estimation for sparse functional data. *Statistics and Computing*, 28, 05 2018.

[22] Ludwig Fahrmeir, Thomas Kneib, and Stefan Lang. Penalized structured additive regression for spacetime data: a bayesian perspective. *Statistica Sinica*, pages 731–761, 2004.

[23] I. D. Currie and M. Durban. Flexible smoothing with p-splines: a unified approach. *Statistical Modelling*, 2(4):333–349, 2002.

[24] D R S Saputro, H Lukmawati, and P Widyaningsih. Smoothing parameters of penalized spline nonparametric regression model using linear mixed model. *IOP Conference Series: Earth and Environmental Science*, 243:012040, April 2019.

[25] G. K. Robinson. That BLUP is a Good Thing: The Estimation of Random Effects. *Statistical Science*, 6(1):15 – 32, 1991.

[26] Ping Ma, Cristian I. Castillo-Davis, Wenxuan Zhong, and Jun S. Liu. A data-driven clustering method for time course gene expression data. *Nucleic Acids Research*, 34(4):1261–1269, 2006.

[27] Gideon Schwarz. Estimating the dimension of a model. *Ann. Statist.*, 6(2):461–464, 03 1978.

[28] Kathleen E. Lotterhos, Jason H. Moore, and Ann E. Stapleton. Analysis validation has been neglected in the Age of Reproducibility. *PLOS Biology*, 16(12):e3000070, December 2018. Publisher: Public Library of Science.

[29] Jason E. Shoemaker, Satoshi Fukuyama, Amie J. Eisfeld, Dongming Zhao, Eiryo Kawakami, Saori Sakabe, Tadashi Maemura, Takeo Gorai, Hiroaki Katsura, Yukiko Muramoto, Shinji Watanabe, Tokiko Watanabe, Ken Fuji, Yukiko Matsuoka, Hiroaki Kitano, and Yoshihiro Kawaoka. An Ultrasensitive Mechanism Regulates Influenza Virus-Induced Inflammation. *PLoS Pathogens*, 11(6):e1004856, June 2015.

[30] Zachary B. Abrams, Travis S. Johnson, Kun Huang, Philip R. O. Payne, and Kevin Coombes. A protocol to evaluate RNA sequencing normalization methods. *BMC Bioinformatics*, 20(Suppl 24):679, December 2019.

[31] Taesung Park, Sung-Gon Yi, Sung-Hyun Kang, SeungYeoun Lee, Yong-Sung Lee, and Richard Simon. Evaluation of normalization methods for microarray data. *BMC Bioinformatics*, 4:33, September 2003.

[32] Nelle Varoquaux. A pipeline to analyse transcriptomic time-course data, June 2021. original-date: 2019-01-08T16:24:04Z.

[33] N. Coffey, J. Hinde, and E. Holian. Clustering longitudinal profiles using P-splines and mixed effects models applied to time-course gene expression data. *Computational Statistics & Data Analysis*, 71:14–29, March 2014.

[34] Christophe Biernacki, Gilles Celeux, and Gérard Govaert. Assessing a mixture model for clustering with the integrated completed likelihood. *IEEE Trans. Pattern Anal. Mach. Intell.*, 22:719–725, 2000.

Appendix

6.1 R-Code

6.1.1 MMBsplines Function

```
#' Mixed Model B-splines main function
#'
#' @param x explanatory variable
#' @param y response variable
#' @param xmin minimum value of x
#' @param xmax maximum value of x
#' @param nseg number of segments
#' @param degree degree of B-splines, default degree=3
#' @param sparse sparse model
#' @param lambda penalty parameter
#' @param optimize boolean, default TRUE
#' @param Psplines boolean, default TRUE
#' @export

MMBsplines = function(x, y,
                      xmin, xmax, nseg, degree = 3, sparse=TRUE,
                      lambda = 1.0, optimize=TRUE, Psplines=TRUE,
                      clustering=FALSE, weights=NULL)
{
  t0 = proc.time()[1]
  p = 2
  N = length(y)
  dx = (xmax - xmin) / (nseg+1)
  knots = seq(xmin - degree * dx, xmax + degree * dx, by = dx)
  B = splineDesign.sparse(knots, x, derivs=rep(0,N), ord = degree + 1)
  m = ncol(B)
  X = matrix(outer(x,c(0:(p-1)),"^"),ncol=p)
  D = diff(diag.spam(m), diff=2)
  if (sparse==FALSE) {
    Z = B %*% t(D) %*% solve(D%*%t(D))
  } else
    Z = B %*% t(D) }
  U = cbind(X,Z)
```

```
UtU = t(U) %*% U
Uty = t(U) %*% y
################ Code to implement when performing clustering
if(clustering==TRUE){
  if(is.null(weights)){
    print("Values for wieghts cannot be NULL when performing clustering,
          please provide values for weights.")
    exit()
  }
  W=diag(weights+rep(0.00001,length(weights)))
  UtU = t(U) %*% W %*% U
  Uty = t(U) %*% W %*%y
}
#############################################################
if (sparse == FALSE)
{
  Q = diag(m-2)
  log_det_Q = 0.0
}
else
{
  A = diag.spam(m-2)
  if (degree == 3 && !Psplines) {
    A = (1/6)*(abs(row(A)-col(A))==1) + diag(4/6,m-2)
  }
  DDt = D %*% t(D)
  Q = DDt %*% A %*% DDt
  log_det_DDt = as.double(determinant(DDt)$modulus)
  log_det_A = as.double(determinant(A)$modulus)
  log_det_Q = 2*log_det_DDt + log_det_A
  #if (nseg > 500) {
  #  Q = Q + 1.0e-12*diag.spam(m-2)
  #}
}
#log_det_Q = 2.0*as.double(determinant(D %*%t(D))$modulus)
Q_all = bdiag.spam(diag.spam(0,2),Q)
penaltylog10 = seq(-6,6, by=0.1)
```

```
q = ncol(U)-2 # number of penalized components
ssy = sum(y^2)
# in the for-loop we use update of Cholesky:
C = UtU + lambda * Q_all
cholC = chol(C)
L = list(C, cholC=cholC,UtU=UtU,Uty=Uty,Q_all = Q_all,
             ssy=ssy, p=p, q=q, N=N, log_det_Q = log_det_Q,xmin=xmin,xmax=xmax,
                 nseg=nseg,
          deg=degree,knots=knots,X=X)
if (optimize)
{
  if (sparse==FALSE)
  {
    result = optimize(REMLlogprofile.dense,c(-6,6),tol=1.0e-12,obj=L,maximum=
        TRUE)
  } else
  {
    result = optimize(REMLlogprofile.sparse,c(-6,6),tol=1.0e-12,obj=L,maximum=
        TRUE)
  }
  lambda_opt = 10^(result$maximum)
  logL_opt = result$objective
} else {
  if (sparse==FALSE)
  {
    logL_opt = REMLlogprofile.dense(log10(lambda),L)
  } else
  {
    logL_opt = REMLlogprofile.sparse(log10(lambda),L)
  }
  lambda_opt = lambda
}
if (sparse==FALSE) {
  C = UtU + lambda_opt * Q_all
  cholC = chol(C)
  a = backsolve(cholC, forwardsolve(t(cholC),Uty))
}
```

```
else {
  cholC = update(cholC,UtU + lambda_opt * Q_all)
  a = backsolve.spam(cholC, forwardsolve.spam(cholC,Uty))
}
yPy = ssy - sum(a*Uty)
sigma2 = yPy/(N-p)

if(clustering==TRUE){
  if(is.null(weights)){
    print("Values for wieghts cannot be NULL when performing clustering,
          please provide values for weights.")
    exit()
  }
  yhi=U%*%a
  sigma2=sum(weights %*% (y-yhi)**2) / (sum(weights)-p)
}
# calc ED dimension, not optimized:
C = UtU + lambda_opt * Q_all
ed = sum(diag(solve(as.matrix(C)) %*% as.matrix(UtU)))
sig_u=sigma2/lambda_opt
t1 = proc.time()[1]
L$time = as.double(t1-t0)
L$lambda_opt = lambda_opt
L$logL_opt = logL_opt
L$cholC = cholC
L$a = a
L$sigma2 = sigma2
L$ed = ed
L$method = ifelse(sparse,"SPARSE","DENSE")
L$sig2_u = sig_u
L$B=B
L$Z=Z
L$D=D
L$X=X
L$U=U
class(L) = "MMBsplines"
L
```

```
}
```

6.1.2 predict.MMBsplines Function

```
#
# Martin Boer, Biometris, WUR, Wageningen, The Netherlands
#
# E-mail: martin.boer@wur.nl
#

#' @export
predict = function(obj, x, linear = FALSE)
{
  if (linear) {
    mu = obj$a[1]
    beta = obj$a[2]
    ylin = mu + beta*x
    return(ylin)
  }
  derivs = rep(0,length(x))
  if (obj$method == "SPARSE")
  {
    B = splineDesign.sparse(obj$knots, x, derivs=rep(0,length(x)), ord = obj$deg
        + 1,outer.ok = TRUE)
    m = ncol(B)
    X = matrix(outer(x,c(0:(obj$p-1)),"^"),ncol=obj$p)
    D = diff(diag.spam(m), diff=2)
    Z = B %*% t(D)
    U = cbind(X,Z)
    yhat = U %*% obj$a
    yhat
  } else {
    B = splineDesign(obj$knots, x, derivs=rep(0,length(x)), ord = obj$deg + 1,
       outer.ok = TRUE)
    m = ncol(B)
    X = matrix(outer(x,c(0:(obj$p-1)),"^"),ncol=obj$p)
    D = diff(diag(m), diff=2)
    Z = B %*% t(D) %*% solve(D%*%t(D))
```

```
    U = cbind(X,Z)
    yhat = U %*% obj$a
    yhat
  }
}
```

6.1.3 Simulate data Functions

```
####### Simulate data

sim1<- function(tij,N , sigE, sigB,clust){
  ni=length(tij)
  sig_e=sigE
  tij=rep(tij,N)
  tij=matrix(tij,nrow = ni,ncol = N)
  yij=matrix(nrow = ni, ncol = N)
  for (i in c(1:N)){
    sig_u=sigB
    bi=rnorm(1,mean = 0,sd=sig_u)
    for (j in c(1:ni)){
      Eij=rnorm(1,0,sig_e)
      yij[j,i]=bi+Eij
    }
  }
  p=ggplot()
  for (i in c(1:N)) {
    df=as.data.frame(cbind(tij[,i],yij[,i]))
    p=p+geom_line(data = df,aes(x=V1,y=V2))
  }
  p=p+labs(x="Time",y="Y")+ theme_classic()+ggtitle(paste("Cluster",clust))

  L=list(yij,tij,p)
  L$yij=yij
  L$tij=tij
  L$p=p
  return(L)
}
```

```
sim5<- function(tij,N , sigE, sigB,clust){
  ni=length(tij)
  sig_e=sigE
  tij=rep(tij,N)
  tij=matrix(tij,nrow = ni,ncol = N)
  yij=matrix(nrow = ni, ncol = N)
  for (i in c(1:N)){
    sig_u=sigB
    bi=rnorm(1,mean = 0,sd=sig_u)
    for (j in c(1:ni)){
      Eij=rnorm(1,0,sig_e)
      yij[j,i]=5**(tij[j,i])+10+bi+Eij
    }
  }
  p=ggplot()
  for (i in c(1:N)) {
    df=as.data.frame(cbind(tij[,i],yij[,i]))
    p=p+geom_line(data = df,aes(x=V1,y=V2))
  }
  p=p+labs(x="Time",y="Y")+ theme_classic()+ggtitle(paste("Cluster",clust))

  L=list(yij,tij,p)
  L$yij=yij
  L$tij=tij
  L$p=p
  return(L)
}


sim6<- function(tij,N , sigE, sigB,clust){
  ni=length(tij)
  sig_e=sigE
  tij=rep(tij,N)
  tij=matrix(tij,nrow = ni,ncol = N)
  yij=matrix(nrow = ni, ncol = N)
  for (i in c(1:N)){
    sig_u=sigB
```

```
    bi=rnorm(1,mean = 0,sd=sig_u)
    for (j in c(1:ni)){
      Eij=rnorm(1,0,sig_e)
      yij[j,i]=-exp(tij[j,i])-7+bi+Eij
    }
  }
  p=ggplot()
  for (i in c(1:N)) {
    df=as.data.frame(cbind(tij[,i],yij[,i]))
    p=p+geom_line(data = df,aes(x=V1,y=V2))
  }
  p=p+labs(x="Time",y="Y")+ theme_classic()+ggtitle(paste("Cluster",clust))

  L=list(yij,tij,p)
  L$yij=yij
  L$tij=tij
  L$p=p
  return(L)
}
```

6.1.4 Simulation Study

```
library(ggplot2)
library(dplyr)
library(caret)
library(splines)
library(spam)
library(pracma)
library(mclust)

source("C:/Users/User/Documents/UP-Stuff/Masters/Research/ToyExample/MMBsplines/
    R/sim_data.R")
source("C:/Users/User/Documents/UP-Stuff/Masters/Research/ToyExample/MMBsplines/
    R/MMBsplines.R")
source("C:/Users/User/Documents/UP-Stuff/Masters/Research/ToyExample/MMBsplines/
    R/aux_functions.R")
source("C:/Users/User/Documents/UP-Stuff/Masters/Research/ToyExample/MMBsplines/
    R/predict.MMBsplines.R")
```

```
#source("recalc.weights.R")

################ Simulate Data ###################################
#set.seed(123)
ni=12
tij=seq(0, 1.2, by = 1/(ni-1))
nii=length(tij)
sets=500
data.sets=list()
for(s in 1:sets){
Y1=sim1(tij,N=100,sigE=1.2,sigB = 0.75,clust=1)
t1=t(Y1$tij)
#print(Y1$p)
Y1=t(Y1$yij)
c1=rep(1,100*nii)
pi=3.1415926534
Y5=sim5(tij,N=70,sigE=0.96,sigB = 0.45,clust=2)#5)
t5=t(Y5$tij)
#print(Y5$p)
Y5=t(Y5$yij)
c5=rep(5,70*nii)
Y6=sim6(tij,N=60,sigE=1.25,sigB = 0.3,clust=6)
t6=t(Y6$tij)
#print(Y6$p)
Y6=t(Y6$yij)
c6=rep(6,60*nii)

real_clusts=c(c1,c5,c6)#c(c1,c2,c3,c4,c5,c6)

Y_full=do.call("rbind", list(Y1,Y5,Y6))
t_full=do.call("rbind", list(t1,t5,t6))

Nn=ncol(Y_full)*nrow(Y_full)

t_long=matrix(t(t_full),nrow=Nn,ncol = 1,byrow = F)
Y_long=matrix(t(Y_full),nrow=Nn,ncol = 1,byrow = F)
```

```
data=as.data.frame(do.call("cbind",list(Y_long,t_long,real_clusts)))
colnames(data)=c("Y","t","Clusters")
#file="C:/Users/User/Documents/UP-Stuff/Masters/Research/ToyExample/MMBsplines"
#write.csv(data,file =paste(file,"data_out2.csv",sep = "") )
data.sets[[s]]=data
}
################## Initialise parms for EM
    ##############################################################################
results=list()
accuracies=c()
ari=c()
R_squared=list()
ex=list()
a_out=list()
custers_out=list()
ji=0
g=0
while (ji<100) {
  g=g+1
  data=data.sets[[g]]
  Y=data[,1]
  x=data[,2]
  degree=3
  xmin=min(x)
  xmax=max(x)
  nseg=nii/2 # nr knots
  p = 2
  N = length(Y)
  dx = (xmax - xmin) / (nseg+1)
  knots = seq(xmin - degree * dx, xmax + degree * dx, by = dx)
  B = splineDesign.sparse(knots, x, derivs=rep(0,N), ord = degree + 1,outer.ok =
      TRUE)
  m=ncol(B)
  uniq.clusts=unique(real_clusts)
  orig.means=list()
  times=unique(x)
  for (j in uniq.clusts) {
```

```
  x1=x[real_clusts==j]
  Y1=Y[real_clusts==j]
  obj=MMBsplines(x1,Y1,xmin = xmin,xmax = xmax,nseg = nseg,degree=degree)
  U1=obj$U
  a1=obj$a
  mean.obj=U1%*%a1
  mean.obj.times=c()
  for (l in times) {
    ave=mean(mean.obj[x1==l])
    mean.obj.times=c(mean.obj.times,ave)
  }
  orig.means[[paste(j)]]=mean.obj.times
}
X=cbind(rep(1,length(Y)),x)
D = diff(diag.spam(m), diff=2)
Z = B %*% t(D) %*% solve(D%*%t(D))
U = cbind(X,Z)
c=3 # nclust
a_init=rep(runif(n=m*c,min=-5,max=5))
a=matrix(a_init,nrow = m,ncol = c)
sig=matrix(runif(n=c,min=0.3,max=1.8),nrow = 1,ncol = c)
pi_c=rep(1/c,c)
llb=-99999999999999999999999999999999999999
res=c()
gamma=matrix(nrow = length(Y), ncol = c)
lli=matrix(nrow = length(Y), ncol = c)
Q = diag(m-2)
log_det_Q = 0.0
lambda=1.5
Q_all = bdiag.spam(diag.spam(0,2),Q)
penaltylog10 = seq(-6,6, by=0.1)
q = ncol(U)-2 # number of penalized components
i=1
it=0
check=500
################ Perform EM alg ################################\
while (it<50 & check>0.0001) {
```

```
  for(i in c(1:c)){
    yhi=U%*%a[,i]
    gamma[,i]= pi_c[i]*dnorm(Y,yhi,sig[,i])
  }
  gamma=gamma/rowSums(gamma)
  gamma=recalc.weights(gamma,c=0.01)
  pi_c=colMeans(gamma)
  nk=colSums(gamma)
  for (i in c(1:c)) {
    yhi=U%*%a[,i]
    wi=diag(gamma[,i])
    UtU=t(U)%*%wi%*%U
    Uty=t(U)%*%wi%*%Y
    C = UtU + lambda * Q_all
    cholC = chol(C)
    a[,i] = backsolve(cholC, forwardsolve(t(cholC),Uty))
   if(nk[i]>p){sig[,i] = sum(gamma[,i] %*% (Y-yhi)**2) / (nk[i]-p)
    }else{sig[,i] = sum(gamma[,i] %*% (Y-yhi)**2) / (nk[i])}
    sig[,i] = sqrt(sig[,i])
    lli[,i] = gamma[,i]%*%(log(pi_c[i]) + log(dnorm(Y,yhi,sig[,i])+0.000000000
        1))
 }
  ll=sum(lli)
  check=abs(llb-ll)
  llb=ll
  it=it+1
  res=rbind(res,t(as.vector(c(it,pi_c,sig,nk,ll))))
}
if(it<50){
  ji=ji+1
results[[ji]]=res
a_out[[ji]]=a
clusts=max.col(gamma)
#custers_out[[ji]][1]=clusts
tau=a[1:2,]
u=a[3:m,]
library(caret)
```

```
est_clust=as.factor(clusts)
l=cbind(est_clust,real_clusts)
############ Create mean plots ##############################
means=matrix(nrow=nrow(data),ncol=c)
df_clusts=as.data.frame(do.call("cbind", list(Y, x, clusts)))
df_true=as.data.frame(do.call("cbind", list(Y, x, real_clusts)))
p1=ggplot(data=df_true)+geom_point(aes(x=x,y=Y,col=real_clusts))+scale_color_
    gradientn(colours = rainbow(7))+
  ggtitle(paste("simulation",g))
print(p1)
p2=ggplot(data=df_clusts)+geom_point(aes(x=x,y=Y,col=clusts))+scale_color_
    gradientn(colours = rainbow(6))+
  ggtitle(paste("simulation",g))
print(p2)
est.clust.means=c()
estimates_clusters=clusts
est.means=list()
for (i in c(1:c)) {
  means[,i]=U%*%a[,i]
  pli=cbind(means[,i],x)
  pli=pli[order(pli[,2]),]

  avg=mean(means[,i])
  est.clust.means=c()
  for (l in times) {
    ave=mean(means[x==l,i])
    est.clust.means=c(est.clust.means,ave)
  }
  est.means[[i]]=est.clust.means
}
#### Determine cluster refactoring ####
means.mat=matrix(nrow = c,ncol=c)

for (i in c(1:c)) {
  #l=org.clusts[i]
  means0=orig.means[[i]]
  for(j in c(1:c)){
```

```
    meansE=est.means[[j]]
    error=abs(mean(means0)-mean(meansE))
    means.mat[i,j]=error
  }
}
min.col <- function(m, ...) max.col(-m, ...)
pred=min.col(means.mat)
real_clusts=recode(real_clusts,'1'=1,'5'=2,'6'=3)

pred.unique=length(unique(pred))
if(pred.unique==3){
  pred=pred
}else if(pred.unique==2){
  cs=c(1,2,3)
  for (i in c(1:3)) {
    pred_i=which(pred==i)
    uni_pred=unique(pred)
    if(length(pred_i)>1){
      r=cs[-which(cs %in% uni_pred)]
      pred[pred_i[2]]=r
    }
  }
}else{
  for (i in c(1:3)){
    uni_pred=unique(pred)
    cols_remaining=cs[-which(cs %in% uni_pred)]
    pred2=min.col(means.mat[-1,cols_remaining])

    if(length(unique(pred2))==2){
      pred[c(2,3)]=cols_remaining[pred2]
    }else{
      pred[2]=cols_remaining[pred2[1]]
      pred[3]=cols_remaining[-pred2[1]]
    }
  }
}
est.clust.recoded=recode(clusts,'1'=which(pred==1),'2'=which(pred==2),'3'=
```

```
      which(pred==3))
  custers_out[[ji]]=c(clusts,as.vector(est.clust.recoded))
  ########## Determine accuracy measures ##############
  ex2<- confusionMatrix(data=as.factor(est.clust.recoded), reference = as.factor
      (real_clusts))
  ex[[ji]]=ex2
  #max.col(t(ex2$table))
  accuracy=ex2$overall[1]
  if(accuracy>0.9){
    print(ex2$table)
    print(accuracy)
    print(p1)
    print(p2)
  }
  ari=c(ari,adjustedRandIndex(as.factor(est.clust.recoded),as.factor(real_clusts
      )))
  accuracies=c(accuracies,accuracy)
  }
  ###############  Calculate R2
  r2=c()
  for (r in c(1:c)) {
    Y_r2=Y[real_clusts==r]
    X_r2=x[real_clusts==r]
    X_est=x[est.clust.recoded==r]
    Y_est=Y[est.clust.recoded==r]
    obj_r2=MMBsplines(x=X_r2,y=Y_r2,xmin,xmax,nseg=nseg,degree=degree,sparse=F)
    y_hat=predict(obj_r2, X_est, linear = FALSE)
    Y_bar=obj_r2$U%*%obj_r2$a
    obj_est=MMBsplines(X_est,y=Y_est,xmin,xmax,nseg=nseg,degree=degree,sparse=F)
    Y_bar_est=obj_est$U%*%obj_r2$a
    RSS=sum((Y_bar_est-y_hat)**2)
    TSS=sum((Y_r2-Y_bar)**2)
    r2=c(r2,1-RSS/TSS)
  }
  R_squared[[ji]]=r2
}
```

6.1.5 Real Data Analysis

```
library (BiocManager)
library(splines)
library(spam)
library(Matrix)
library(face)
library(ggplot2)
library(RColorBrewer)
library(reshape2)
library(viridis)
library(pracma)
# Now load in the metadata
data(shoemaker2015)
meta = shoemaker2015$meta
data_real = shoemaker2015$data
tpts=rep(meta$Timepoint,nrow(data_real))
grp=rep(meta$Group,nrow(data_real))
reps=rep(meta$Replicate,nrow(data_real))
cols=which(meta$Replicate==1)
Nt=nrow(data_real)
ni=nrow(meta)

time.points=t(matrix(tpts,nrow = ni,ncol = Nt))
groups=t(matrix(grp,nrow = ni,ncol = Nt))
replicas=t(matrix(reps,nrow = ni,ncol = Nt))
reps1=replicas[1,]
grs=groups[1,(which(reps1==1))]
Nc=1000
len_data=nrow(data_real)
data_list=list()
time_list=list()
for (j in c(1:50)) {
  samp=sample(c(1:len_data),Nc)
  data_use=data_real[samp,(which(reps1==1))]
  times=time.points[samp,(which(reps1==1))]#& which(grs=="K"))]
  data_use=data_use[,which(grs=="M")]
  times=times[,which(grs=="M")]
```

```
  data_list[[j]]=data_use
  time_list[[j]]=times
}

res_list=list()
a_list=list()
clust_list=list()
Ks=c(2,5,8,9,10,11,12,13,14,15,20,25)

g=0
ji=0
c=Ks[3]
while (ji <1) {
  g=g+1
  data_use=data_list[[g]]
  times=time_list[[g]]
  ni=ncol(data_use)
  Y=matrix(t(data_use),nrow=Nc*ni,ncol = 1,byrow = F)
  x=matrix(t(times),nrow=Nc*ni,ncol = 1,byrow = F)
  degree=3
  xmin=min(x)
  xmax=max(x)

  nseg=length(unique(as.vector(x)))/2 # nr knots
  p = 2
  N = length(Y)
  dx = (xmax - xmin) / (nseg+1)
  knots = seq(xmin - degree * dx, xmax + degree * dx, by = dx)
  B = splineDesign.sparse(knots, x, derivs=rep(0,N), ord = degree + 1,outer.ok =
      T)
  m=ncol(B)
  X=cbind(rep(1,length(Y)),x)
  D = diff(diag.spam(m), diff=2)
  Z = B %*% t(D) %*% solve(D%*%t(D))
  U = cbind(X,Z)
  a_init=rep(runif(n=m*c,min=-20,max=20))
  a=matrix(a_init,nrow = m,ncol = c)
```

```
sig=matrix(runif(n=c,min=0.3,max=3),nrow = 1,ncol = c)
pi=rep(1/c,c)#rand_vect_cont(N=10,M=1)
llb=-99999999999999999999999999999999999999
res=c()
gamma=matrix(nrow = length(Y), ncol = c)
Q = diag(m-2)
log_det_Q = 0.0
lambda=1.75
Q_all = bdiag.spam(diag.spam(0,2),Q)
penaltylog10 = seq(-6,6, by=0.1)
q = ncol(U)-2 # number of penalized components
lli=matrix(nrow=1,ncol=c)
i=1
it=0
check=500
 ###################### EM Algorithm #############
while (it<30 & check>0.005) {
  for(i in c(1:c)){
    yhi=U%*%a[,i]
    gamma[,i]= pi[i]*dnorm(Y,yhi,sig[,i])
  }
  gamma=gamma/(rowSums(gamma)+0.00000001)
  gamma=recalc.weights(gamma,c=0.01)
  pi=colMeans(gamma)
  nk=colSums(gamma)
  for (i in c(1:c)) {
    yhi=U%*%a[,i]
    wi=diag(gamma[,i]+rep(0.00001,nrow(gamma)))
    if(sum(is.na(wi))>1){print(paste(it,i))
      break()}
    UtU=t(U)%*%wi%*%U
    Uty=t(U)%*%wi%*%Y
    C = UtU + lambda * Q_all
    cholC = chol(C)
    a[,i] = backsolve(cholC, forwardsolve(t(cholC),Uty))
    sig[,i] = sum(gamma[,i] %*% (Y-yhi)**2) / (nk[i])
    sig[,i] = sqrt(sig[,i])
```

```
      lli[,i] = gamma[,i]%*%(log(pi[i]) + log(dnorm(Y,yhi,sig[,i])+0.000001))
    }
    ll=sum(lli)
    check=abs(llb-ll)
    llb=ll
    it=it+1
    res=rbind(res,t(as.vector(c(c,it,pi_c,sig,nk,ll))))
  }
  if (it<30){
    res_list[[paste(c)]]=res
    a_list[[paste(c)]]=a
    clusts=max.col(gamma)
    clust_list[[paste(c)]]=clusts
    ji=1
  }
 }

clusters=clust_list$`12`
data_plot=data.frame(Y=Y,time=x)
data_plot$clusters=clusters
uniq_clusts_real=unique(clusters)
for (i in c(1:length(uniq_clusts_real))) {
  c=uniq_clusts_real[i]
  dat_plt=data_plot[data_plot$clusters==c,]
  ni=length(unique(dat_plt$time))
  N=nrow(dat_plt)/ni
  p=ggplot()+theme_classic()
  +scale_color_gradientn(colours = magma(50),limits=c(-30,20))
  for(j in c(1:N-1)){
    start=(j-1)*ni+1
    fin=j*ni
    df_mini=dat_plt[c(start:fin),]
    p=p+geom_line(data=df_mini,aes(x=time,y=Y))

  }
  p=p+theme(legend.position = "none")+
  +geom_line(data = df_meansPlot,aes(x=x1,y=mean1),lwd=1)
```

```
    ggtitle(paste("Cluster ",i))

  if(NROW(dat_plt>=100)){
   # p=plot(x=dat_plt$time,y=dat_plt$Y)+lines()
  }
  print(p)

}
```