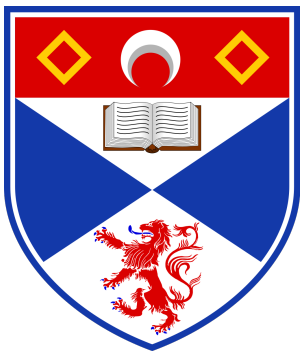


Robust Inference of Semi-Functional Linear Regression Models

Samraj Singh 180005888

School of Mathematics and Statistics

University of St Andrews

August 2023

supervised by Elham Mirfarah

I hereby certify that this dissertation, which is approximately 9033 words in length, has been composed by me, that it is the record of work carried out by me and that it has not been submitted in any previous application for a degree. This project was conducted by me at the University of St Andrews from May 2023 to August 2023 towards fulfilment of the requirements of the University of St Andrews for the degree of MSc Applied Statistics and Datamining under the supervision of Elham Mirfarah.

Abstract

In this paper, we explore the field of Functional Data Analysis (FDA). Although FDA provides a valuable way to analyse functional data, it is still vulnerable to issues that persist in traditional statistical analysis. One such issue is the ability of Functional Linear Regression techniques to handle the presence of outliers in data. As the field of FDA is still emerging, relatively little work has been done on this subject. We reviewed the literature on the subject and chose to focus on the process of generating robust estimations of Semi-Functional Linear Regressions using a Bayesian approach. We chose to focus on the work of Shan et al. (2020), noticing oversights in their proposal and aiming to remedy these. This refined approach was then implemented in R, expanding the code base for this robust estimation technique. We applied the proposed models to both simulated and real-world datasets to gauge the efficacy and robustness of the models, utilising a broader array of selection criteria compared to the work of Shan et al. (2020). The code related to the analysis carried out in this work can be found on GitHub.

Contents

Chapter 1

Introduction

The origins of Functional Data Analysis (FDA) can be found in the late 20th Century. The term 'Functional Data Analysis' was originally authored by Ramsay (1982) [25]. Both Ramsey and Dalzell (1991) [25] made significant contributions to the foundation of FDA. They built on the work of Grenander (1950) and Rao (1958) [25]. As the popularity of FDA has grown with time, so too have the analysis techniques within FDA. This includes techniques such as: functional principal component analysis (FPCA); functional linear regression models; and functional clustering.

A functional data point $X(t)$ can be thought of as a realisation of a real-valued function, where $X : T \to \mathbb{R}$, where T is a subset of the real line [25]. We can think of these data points as a realisation of a one-dimensional stochastic process [25]. The ability to view data as functions rather than scalars [25] is what makes FDA so fascinating. This unique perspective on data enables a better understanding of underlying trends and a more effective analysis of datasets that vary over a continuous domain.

As time goes on, the importance of Functional Data Analysis has grown. It gives us a unique approach to analyse data that varies continuously over a domain, such as time or space. Put simply, FDA is the study of curves [25]. Although FDA is unique in its ability to analyse curves (real-valued functions), it can still be viewed as an extension of multivariate data analysis. Due to this, we see various parallels between the two approaches [20].

As society advances, so too does our data's complexity and diversity. After the technological revolution, our ability to generate and collect data has grown tremendously. Data has gone from simple numerals to vast and complex databases. For instance, in medicine, we can now use ECG data, curves that represent a patient's heart rhythm, and diagnose heart diseases we could not previously. Similarly, in physics, researchers can employ mass spectrometer techniques to study the light absorption of materials and generate insights. Both of these sources of data can be represented as curves, making FDA the ideal analysis approach.

FDA provides a good framework by which we can analyse functional data, yet it still faces challenges seen in traditional methods. A key challenge is the sensitivity of estimation techniques towards outliers in the data. These outliers skew results and lead to misleading conclusions.

In this paper, we delve into the estimation of functional and semi-functional linear regressions. Later explained in further detail, methods used to estimate these regressions, such as least-squares estimates and maximum likelihood calculations, are not robust to the presence of outliers in the data. Therefore, using these techniques in cases where the data is not distributed normally results in incorrect inferences and estimates. We aim to explore methods to make these estimates robust to outliers. Ultimately we chose to adapt the method proposed by Shan et al. (2020) [23]. With this methodology, we use Bayesian techniques. This allows us to utilise a class of distributions known as Scale-Mixture of Normal [11], explained in detail later in the text.

Despite the growing body of literature on FDA, the volume of literature related to the robust estimation of Semi-Functional Linear Regressions is relatively sparse. Due to how new the field of FDA is, this topic has only recently begun to be explored. As such, there needs to be more evaluations of the current proposed methodologies to tackle this issue. Consequently, our aim is to examine the existing methods, with a focus on the work of Shan et al. (2020) and try to expand upon this work. Our goal is to enhance the robustness of their methodology by utilising new datasets, simulation studies, and a broader range of model evaluation techniques. We also plan to expand the existing code base for this methodology, coding our work in R. This allows for more flexibility and transparency in our

approach, which may act as a foundation for future research.

The rest of the dissertation is structured as follows:

- **Chapter 2** offers an in-depth literature review on the robustness of semi-functional linear regression models, and an overview of the applications related to the paper.
- **Chapter 3** explains the methodology, and the proposed enhancements.
- **Chapter 4** shows and evaluates the performance of the models, using simulated data.
- **Chapter 5** shows and evaluates the performance of the models, using a real-world dataset.
- **Chapter 6** interprets these findings, presenting an overall conclusion and suggesting potential future areas of research.

Chapter 2

Historical Overview and Background

2.1 Literature Review

As this paper is concerned with an examination into how we can robustly estimate a semi-functional linear regression, also referred to as a partially functional linear regression, we conducted a literature review of the existing methods. Firstly, we ventured into the origin of semi-functional linear regression. The first paper that aimed to model scalar response variable with both functional and non-functional covariates was titled 'Semi-functional partial linear regression', written by Germán Aneiros-Pérez and Philippe Vieu (2006) [1]. Their paper laid the groundwork for the extensions that followed. The primary extension of interest is the robust estimation of these scalar response values. The first approach to robustly estimate the response was written by Huang et al. (2015), who introduced the sieve M-estimator, a novel approach to ensure robustness against outliers [16]. Building on this research, Zhou et al. (2016) proposed a method of M-Estimation for partially functional linear regression models that was based on splines, generating response values that were robust to outliers and heavy-tailed errors [27]. Boente and Vahnovan (2017), Cai et al. (2020), and Boente et al. (2020) expanded upon these methods of robust estimation [5, 7, 4]. A shift from this approach was proposed by Shan et al. (2020), who utilised Bayesian robust estimation using heavy-tailed distributions [23].

As this field of research is still in its infancy, the amount of research compared to other fields is relatively sparse. As such, the majority of work has been focused on the use of sieve M-estimators to make estimations robust to outlier, as shown by the papers mentioned prior. However, the approach suggested by Shan et al. posits that the utilisation of heavy-tailed distributions in a Bayesian regression can account for robustness more effectively.

Empirical studies are useful to properly illustrate the benefits of various theoretical concepts in the real world. Furthermore, for research to be accepted, it must be properly scrutinized. As the work of Shan et al. (2020) is still new, it has not been extended or properly analysed. This prompts further research into this case. In their paper, they used their proposed methods on data related to impact of air pollution on mortality, taken from the NMMAPS data package [23]. Their findings showed that the proposed method utilising heavy-tailed distributions performed more effectively than using a normal distribution. This was validated by a DIC calculation.

In summary, we found that there is a need for a refinement and further analysis of currently proposed Bayesian methods to conduct robust estimation. In particular, the current work lacks a thorough analysis of the existing methods. Therefore, addressing these gaps is not merely an academic exercise, but rather crucial in order to ensure accuracy and reliability of current methods.

Our review of the literature on semi-functional linear regression models covered the history, key theories, and existing gaps in the literature. These gaps showcase the need for further research into Bayesian methods, building on the work of Shan et al. (2020). Furthermore, the review revealed interesting areas for future research, such as the incorporation of skewness into the Bayesian methods, allowing for the modelling of data that exhibits asymmetries, which appears as a natural extension of the work of Shan et al. (2020).

2.2 Functional Linear Regression

A Functional Linear Regression (FLR) is an extension of the traditional linear regression that allows for the incorporation of functional data. In this section, we explore the topic of FLRs, their mathematical representations, and common strategies for coefficient estimation. FLRs can involve a functional response variable, a scalar response variable, or a combination of both, depending on the model specified [25].

Mathematically, a simple FLR model with a scalar response can be represented as follows [25]:

$$Y = \alpha + \int \beta(t) X(t) dt + \epsilon, \tag{2.1}$$

where Y is a scalar response, α is a scalar intercept, $X(t)$ is a square-integrable stochastic process, $\beta(t)$ is an unknown square-integrable function, and ϵ is the random error.

Another type of common FLR model is one that contains functional terms in the response variable as well as the regressors. This is referred to as a fully functional model, sometimes referred to as a FLM [25]. This type of model can be seen below:

$$Y(t) = \beta_0(t) + \int \beta(s,\ t) X(s) ds + \epsilon(t), \tag{2.2}$$

where $Y(t)$ is a functional response, $\beta_0(t)$ is a functional intercept, $X(s)$ is a square-integrable stochastic process, $\beta(s,\ t)$ is the regression coefficient surface, approximated via basis expansion, and $\epsilon(t)$ is the random error.

Although FLRs allow for the modelling of complex relationships within data, there are some limitations. One being, that FLRs assume a linear relationship between the response and predictors. Furthermore, it can be a computationally intensive task, especially when dealing with large or high-dimensional data.

For our purposes, we are concerned with estimating a semi-functional linear regression. This can be represented as [7]:

$$Y = \alpha^T \mathbf{Z} + \int \beta(t) X(t) dt + \epsilon, \tag{2.3}$$

where Y is a scalar response, α is an unknown p-dimensional parameter vector, $\mathbf{Z}$ is the p-dimensional predictor vector, defined as $\mathbf{Z} = (Z_1, \ldots, Z_p)^T$ [26], $X(t)$ is a square-integrable stochastic process, $\beta(t)$ is an unknown square-integrable function, and ϵ is the random error.

Here, we now explicitly account for the use of non-functional data. As such, these regressions can now utilise 'hybrid' datasets. We use this implementation going forward, both in our simulated studies and real-data analysis.

2.2.1 Common Strategies for Functional Coefficient Estimation

Following the foundation laid out by Ramsey and Dalzell, numerous papers have elaborated on estimation methods [24]. As such, there are now a variety of methods we can use to estimate functional coefficients such as: least squares estimation, penalised regression, and Bayesian approaches. In order to implement these approaches, we first need to represent the functional data as a set of discrete values, one popular method of doing so is through basis expansion [20].

The concept of basis expansion can be illustrated by the functional coefficient $\beta(t)$, given an orthonormal basis φ_k, where $k \geq 1$. For an orthonormal basis, each basis function is orthogonal (i.e., independent) to every other and each has a unit length. We can represent $\beta(t)$ as follows [25]:

$$\beta(t) = \sum_{k=1}^{\infty} \beta_k\ \varphi_k(t), \tag{2.4}$$

where $\beta(t)$ represents the coefficient function, β_k represents the coefficients associated with the basis functions. These coefficients are generated from the data and assign weighting to each basis function. $\varphi_k(t)$ represents the k-th basis function.

The basis expansion also allows us to represent $X(t)$ as follows [25]:

$$X(t) = \sum_{k=1}^{\infty} A_k \, \varphi_k(t), \tag{2.5}$$

where $X(t)$ represents the predictor function, A_k represents the coefficients associated with the basis functions. These coefficients are generated from the data and assign weighting to each basis function, and $\varphi_k(t)$ represents the k-th basis function.

These properties allow us to represent the functional linear regression as such:

$$Y = \alpha + \sum_{k=1}^{\infty} \beta_k A_k + \epsilon, \tag{2.6}$$

In practice, conducting basis expansion often requires truncation, primarily because it is computationally inefficient to sum over an infinite set of basis functions. To clarify, while the set of potential basis functions is infinite, to represent specific functions, we only need a finite set to get a good approximation. To help find the right finite value, we can use Functional Principal Component Analysis (FPCA).

FPCA allows for efficient truncation by identifying and retaining only the most important basis functions—those that explain the most variance in the data. By doing so, it reduces the dimensionality of the data without sacrificing a significant amount of information, allowing us to generate an appropriate approximation. For example, after conducting FPCA, we are given some suitable truncation value for K, where $K < \infty$, that explains the amount of variation in the data we deem to be important. This results in the formula below:

$$Y = \alpha + \sum_{k=1}^{K} \beta_k A_k + \epsilon, \tag{2.7}$$

where Y represents the response variable, α represents the intercept term, β_k represents the coefficients associated with A_k, A_k represents the coefficients gained through projecting the $X(t)$ onto the k-th basis function, and ϵ represents the error term.

After completing the basis expansion, there are several methods by which we can estimate coefficient values, such as: least-square estimations, penalised residual sum of squares and Bayesian estimation [20].

There are a multitude of packages in R that allow us to conduct FDA such as: fda, fda.usc, fts, and refund. In our analysis, we use the packages fda and fda.usc.

2.3 Functional Principal Component Analysis

Functional principal component analysis (FPCA) can be thought of as an extension of the classical multivariate principal component analysis [20]. Functional principal component analysis (FPCA) is a method for lowering the dimensionality of functional data by projecting the data on a finite-dimensional sub-space, lowering the cost incurred from modelling the data, whilst retaining the most important information. This can be accomplished by decomposing the data into a set of orthogonal basis functions, called functional principal components, that capture the most important patterns of variation in the data. The finiteness of the representation is key here. This process allows us to retain the most important values, and remove everything else, just as with multivariate PCA.

One key difference with FPCA is that we often need to smooth our functional data. Smoothing is necessary as our data is often a set of discrete values indexed over a continuum such as time. This process aims to capture the underlying trends of the data, and remove the noise. There are several methods to smooth data such as: spline smoothing, kernel smoothing, and wavelet smoothing. Ullah and Finch (2013) through their systematic review found that B-spline smoothing was the most popular technique in the papers they reviewed. The popularity of B-spline smoothing makes sense, as B-Splines are generated through a recursive process which normalises values. Hence, they stay between 0 and 1, meaning the basis matrix values are generally smaller than other techniques, making the approach less computationally heavy than alternatives.

Once our data is ready, we can conduct FPCA, as follows. Let's denote the functional data as $\{x_i(t)\}$, where $i = 1, \ldots, n$ indexes the observations at the various time points. We assume that the

functions are centered, so that $\int x_i(t)dt = 0$ for all i, meaning the cross-sectional means are equal to zero [20]. Through FPCA we aim to find some orthonormal basis functions $\{\phi_k(t)\}$, that explain most of the variation in the data [20].

We can calculate the functional principal components by solving the eigenvalue problem for the covariance operator of the data. The covariance function can be represented as [3]:

$$c(s,t) = \frac{1}{n}\sum_{i=1}^{n} x_i(s)x_i(t),$$

where s and t are time points. This shows the similarity/dissimilarity between functions at different times. From this we can generate the covariance operator, given as [3]:

$$C(s,t)x(s) = \int c(s,t)x_i(t)dt,$$

The eigenvalue problem for the covariance operator is given by:

$$C(s,t)\phi_k(t) = \lambda_k\phi_k(s),$$

where λ_k is the kth order eigenvalue and $\phi_k(t)$ is the kth order orthonormal basis functions. The eigenvalues and eigenvectors can be calculated using singular value decomposition (SVD) or eigenfunction expansion [20].

As with traditional PCA, the functional principal components are ordered by their corresponding eigenvalues, with the first functional principal component explaining the most variation, and the second explaining the second and so on.

After we have calculated the FPCs, the original functions can be expressed as a linear combination of the functional principal components, using the coefficients as weights as shown [3]:

$$x_i(t) = \sum_{k=1}^{\infty} \xi_{ik}\phi_k(t),$$

where ξ_{ik} is the coefficient for the kth functional principal component of the ith function. The coefficients are generated by projecting the original functions onto the functional principal components, using inner products [3]:

$$\xi_{ik} = \langle x_i, \phi_k \rangle = \int x_i(t)\phi_k(t)dt,$$

where $\langle \cdot, \cdot \rangle$ denotes the inner product on the space of square integrable functions [3].

In practice, the infinite sum in the above equation is truncated to a finite number of terms, K. The choice of K depends on the amount of variation in the data we want to explain. We can utilise scree plots or compute the cumulative sum of explained variance to help us decide the value of K.

2.4 Scale-Mixture of Normal

The Scale-Mixture of Normal (SMN) distribution is the result of various normally distributed variables, where the variance is not fixed, but instead is a random variable governed by another (independent) probability distribution [11]. The benefits of the SMN distribution are that it allows us to approximate high dimensional probability densities (particularly useful for Bayesian analysis) [19], as well as providing a method by which we can model heavy-tailed data. These distributions are still symmetric (as with a typical normal distribution); however, the tails can be thicker depending on how the variance is scaled [21].

We use U to denote the random variable by which we scale the variance of the normally distributed variable. To illustrate, we can consider the random variable Y, which has an SMN distribution given as $Y \sim \text{SMN}(\mu, \sigma^2, H; \kappa)$ [23], this can be represented stochastically as:

$$Y = \mu + U^{-\frac{1}{2}}Z, \tag{2.8}$$

where $Z \sim \mathcal{N}(0, \sigma^2)$, U is the (strictly positive) mixing parameter with cdf $H(.;\kappa)$ which is modulated by the parameter κ, and $\mu \in \mathbb{R}$ is the central location parameter of the normal distribution [11, 23].

The probability density function (pdf) of Y can be written as:

$$f(y) = \int_0^\infty \phi\left(y|\mu, u^{-1}\sigma^2\right) dH(u;\kappa), \tag{2.9}$$

Likelihood calculations involving SMN distributions often become untraceable as n increases [2, 20]. To circumvent this issue, we require the use of numerical approximations. One such method is through Hierarchical Bayes modelling. The hierarchical Bayes representation of the above distribution is as follows:

$$\begin{aligned} Y|U = u &\sim \mathcal{N}\left(\mu, u^{-1}\sigma^2\right) \\ U &\sim H(.;\kappa), \end{aligned}$$

This allows us to make numerical approximations and generate posteriors which can be used in MCMC algorithms [23]. Depending on the probability law of U, the SMN distribution can mimic many distributions. We opt to consider these four special cases [18] in this paper, following Shan et al.(2020) [23]:

- **Student-t Distribution:** if U follows a Gamma distribution with parameters $\left(\frac{\kappa}{2}, \frac{\kappa}{2}\right)$, then Y would follow a Student-t distribution with κ degrees of freedom [23].
- **Normal Distribution:** if U is equal to 1 with a probability of 1, then Y would follow a Normal distribution [18].
- **Slash Distribution:** if U follows a Beta distribution with parameters Beta(κ, 1), then Y would follow a Slash distribution [13].
- **Contaminated Normal (CN) Distribution:** if U_i follows a discrete probability distribution. Where U_i:

  $$\begin{cases} U_i = 1, & w.p.\ 1-\kappa_1, \\ U_i = \kappa_2, & w.p.\kappa_1 \end{cases},$$

  where the parameters $\kappa_1 \in [0,1]$ and $\kappa_2 \in [0,1]$. The resulting SMN distribution would be a Contaminated Normal.

As illustrated in Figure 2.1, the Student-t, CN, and Slash distributions all have heavier tails than Normal.

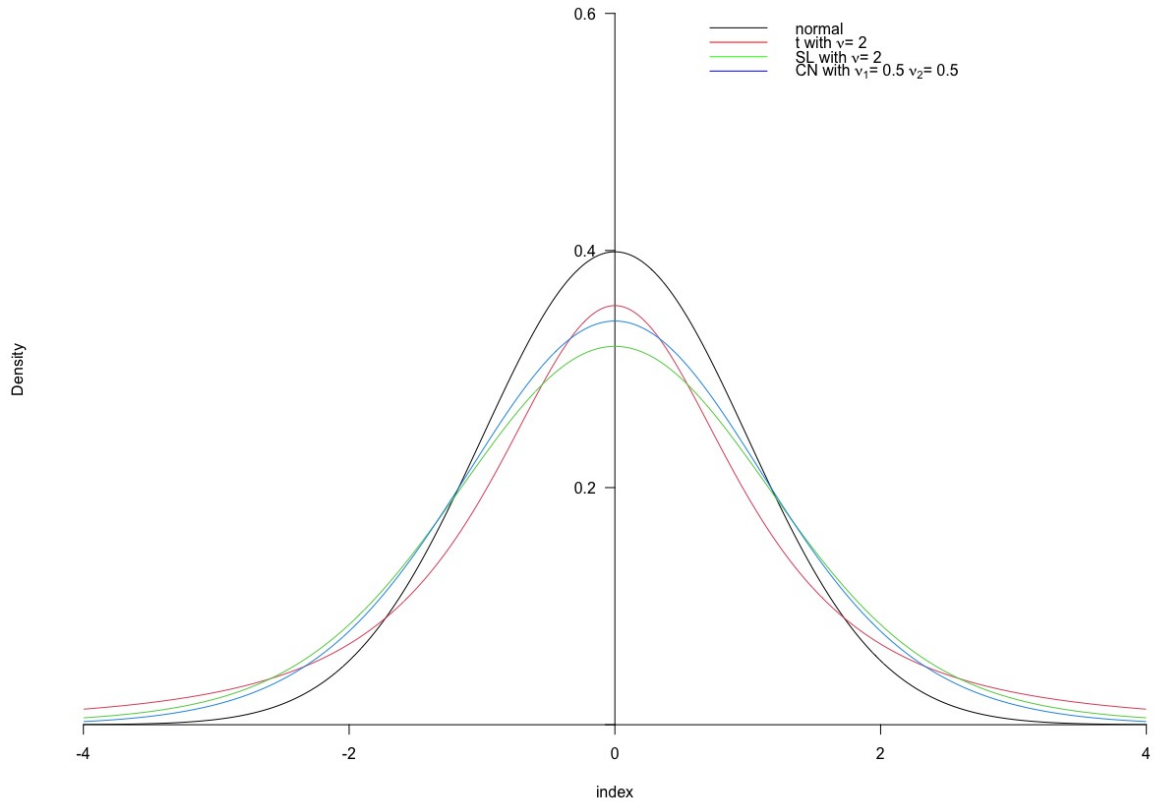


Figure 2.1: Density profiles of Normal, Student-t, Slash, and CN distributions

2.5 Bayesian Inference

In Bayesian analysis, we formalise our beliefs of the parameters (θ) in a statistical model [14], as opposed to the frequentist approach where specific numerical values are hypothesised. This belief, which accounts for the uncertainty about the parameter values, is updated based on Bayes' Formula as new (model) data (X) is collected [14]. In Bayesian analysis, our beliefs are formalised as distributions over the possible range of values for the underlying parameters.

In Bayesian analysis, both the data and the parameters are treated as random variables, which, from a probabilistic perspective, gives rise to a mutually dependent relationship. As with traditional analysis, the link between the parameters and data is represented through the likelihood function, $p(X|\theta)$, which gives the likelihood of a value occurring in the data X for different values of θ, this makes up one part of this mutually dependent relationship. The other key part, which is unique to Bayesian analysis, is the prior distribution, $p(\theta)$. This distribution reflects our initial beliefs about the parameter θ. When we have no prior knowledge about a parameter, we use a non-informative prior, which does not significantly impact our analysis [17].

These two elements, the prior distribution and the likelihood function, come together through the Bayes' theorem, giving the posterior distribution:

$$p(\theta|X) = \frac{p(X|\theta) \cdot p(\theta)}{p(X)} \tag{2.10}$$

The posterior distribution $p(\theta|X)$ represents our updated beliefs about θ after incorporating the prior information $p(\theta)$ and the observed data X. It tells us about the uncertainty of θ given the information we have observed. Here, $p(X)$ serves as a normalising constant, ensuring the posterior integrates to 1 [14]:

$$p(X) = \int p(X|\theta) \cdot p(\theta)\, d\theta \tag{2.11}$$

However, if the data we are using is high-dimensional, then likelihood calculations can become too complex and practically infeasible. This difficulty comes from the integration of the normalising constant. In these cases, we use techniques such as Markov Chain Monte Carlo to approximate [14]. We can also represent the posterior as:

$$\text{posterior} \propto \text{likelihood} \times \text{prior} \tag{2.12}$$

We use this generalisation in our modeling when the high-dimensional nature of the data makes the process of calculating the normalising constant infeasible. As such, we use this approach in our methodology.

2.6 MCMC

As explained earlier in our overview of SMN distributions, due to the intractability of the likelihood function based on the SMN distributions as n (sample size) increases, we need to use MCMC to implement a hierarchical Bayesian approach. Markov Chain Monte Carlo (MCMC) algorithms are used to simulate multivariate probability distributions [8]. MCMC algorithms use Markov chains and Monte Carlo simulations to generate samples of values from a specified posterior distribution [22], which we can then use to make inferences.

There are many ways we can simulate Markov Chains; two popular methods are the Gibbs Sampler and the Metropolis-Hastings algorithm. For our analysis, we utilised both of these methods.

2.6.1 The Gibbs Sampler

The Gibbs sampling algorithm works by generating random samples from a known probability distribution. This method is best used when the conditional distributions of the variables within the joint distribution are known or can be easily calculated [9].

The Gibbs sampler is an iterative procedure. We start with an initial set of values, which are used to generate our first set of samples from the joint distribution. After the initial samples have been

generated, the algorithm updates each value of interest iteratively, using the conditional distributions, while keeping the value of the rest of the variables constant [9].

The process can be explained as follows:

1. Choose the variable you want to update.
2. Sample a new value for this variable with conditional distribution, taking into account the current values of the other variables in the system.
3. Update the variable using the generated sample.
4. Repeat the process for each variable.

We run the algorithm for as many iterations as specified. After a sufficient number of iterations, the algorithm will eventually converge to a stationary distribution, which we can then make inferences on.

2.6.2 The Metropolis-Hastings algorithm

The Metropolis-Hastings algorithm generates random samples that are meant to follow a complex probability distribution that is too difficult to sample values from directly. The algorithm achieves this through a rejection sampling approach [8].

The Metropolis-Hastings algorithm also follows an iterative approach; however, it generates candidate values from a specified proposal distribution rather than our target distribution as direct sampling is not possible, unlike with the Gibbs Sampling approach. The candidate values generated from the proposal distribution are then evaluated against a function, and an acceptance probability to decide whether or not to accept or reject each value is calculated [9].

The process can be explained as follows:

1. Generate a candidate value from the proposal distribution.
2. Calculate acceptance probability ratio.
3. Accept the candidate value with acceptance probability. If accepted, the candidate value replaces the current value. Otherwise, the old value is retained.

This process is to be carried out for as many iterations as you specified in the algorithm until it converges to a stationary distribution. The Markov Chain samples generated by this algorithm converge to the target probability distribution we could not directly sample from. This occurs due to the acceptance probability ratio, which ensures that the values generated are close to our target distribution. Afterwards, these samples can then be used for Bayesian inference [8].

Chapter 3

Methodology

3.1 Model Specification

We consider the Functional Linear Regression:

$$Y = \int_0^1 \beta(t)X(t)\,dt + \epsilon, \tag{3.1}$$

where Y is a scalar response, X is a square-integrable stochastic process, β is a unknown square-integrable function, ϵ is the random error.

The random error ϵ is assumed to follow a SMN distribution, as explained earlier in the paper.

By using suitable truncated basis expansions (for β and X) on the space of square-integrable functions, as discussed in Section 2.2.1, model (3.1) becomes:

$$Y = \boldsymbol{\theta}^*\boldsymbol{\xi} + \epsilon, \tag{3.2}$$

where $\boldsymbol{\theta}, \boldsymbol{\xi} \in \mathbb{R}^d$, with $\boldsymbol{\theta}$ being our unknown parameter of interest and $\boldsymbol{\xi}$ represents a data matrix.

In order to account for the possibility of our data following a heavy-tailed distribution, we examined four special cases of the SMN distribution. To allow for the different cases, we need to use four Bayesian hierarchical models. Here we outline the Bayesian hierarchical models used for each case, with the associated priors for each variable of interest. We utilise the hyper-parameters $a, b, c, d, \boldsymbol{\vartheta}, \boldsymbol{\Omega}$ depending on the case specified.

Distribution Notation

- $\mathcal{N}$ represents the Normal distribution.
- $\mathcal{G}$ represents the Gamma distribution.
- $\mathcal{E}$ represents the Exponential distribution.
- $\mathcal{U}$ represents the Uniform distribution.
- $\mathcal{B}$ represents the Beta distribution.
- $\mathbb{I}$ represents an indicator function that is used to truncate the values supported by a distribution, between a specified interval.

3.2 Normal Distribution

The hierarchical model for the Normal distribution is the simplest case, since this occurs when U_i takes a value of 1 as outlined earlier in the paper, resulting in the model below [23]:

$$\begin{aligned} Y_i|\boldsymbol{\theta}, \sigma^2 &\sim \mathcal{N}\left(\boldsymbol{\theta}^*\boldsymbol{\xi}_i, \frac{1}{\sqrt{\tau}}\right) \\ \boldsymbol{\theta} &\sim \mathcal{N}(\boldsymbol{\vartheta}, \boldsymbol{\Omega}) \\ \tau &\sim \mathcal{G}(a, b), \end{aligned}$$

For notational convenience, we use the reciprocal of variance $\tau = 1/\sigma^2$. With this model, the joint posterior distribution of $(\boldsymbol{\theta}, \tau)$ is characterised by the fully conditional distributions:

$$\boldsymbol{\theta} \sim \mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma}) \tag{3.3}$$

$$\tau \sim \mathcal{G}\left(a + \frac{n}{2}, b + \frac{1}{2}\sum_{i=1}^{n}(Y_i - \boldsymbol{\theta}^*\boldsymbol{\xi}_i)\right), \tag{3.4}$$

where we denote [23]:

$$\boldsymbol{\mu} = \boldsymbol{\Sigma}\left(\boldsymbol{\Omega}^{-1}\boldsymbol{\vartheta} + \tau\sum_{i=1}^{n} U_i Y_i \boldsymbol{\xi}_i\right), \ \boldsymbol{\Sigma} = \left(\boldsymbol{\Omega}^{-1} + \tau\sum_{i=1}^{n} U_i \boldsymbol{\xi}_i \boldsymbol{\xi}_i^*\right)^{-1}. \tag{3.5}$$

As this is the Normal distribution, where U_i takes the value 1, as explained in Section 2.4

The corresponding Gibbs Sampling algorithm for this model can be implemented as follows:

Algorithm 1 Gibbs Sampling for the Normal Distribution

1: Assign suitable values to hyper-parameters $a, b, \boldsymbol{\vartheta}, \boldsymbol{\Omega}$.
2: Initialise the parameter vectors $(\boldsymbol{\theta}, \boldsymbol{U}, \lambda, \tau, \kappa)$ with suitable values.
3: **for** *iteration* $= 1, 2, \ldots,$ N **do**
4: Generate $\bar{\boldsymbol{\theta}} \sim \mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma})$ according to (3.3), where $\boldsymbol{\mu}$ and $\boldsymbol{\Sigma}$ are given by (3.5)
5: $\boldsymbol{\theta} \leftarrow \bar{\boldsymbol{\theta}}$
6: Generate $\bar{\tau}$ according to (3.4)
7: $\tau \leftarrow \bar{\tau}$
8: **end for**
9: Discard burn-in samples.

3.3 Student-t Distribution

For the Student-t distribution, the Hierarchical Bayesian model is constructed as follows [23]:

$$\begin{aligned} Y_i|U_i, \boldsymbol{\theta}, \sigma^2 &\sim \mathcal{N}\left(\boldsymbol{\theta}^*\boldsymbol{\xi}_i, \frac{1}{\sqrt{\tau U_i}}\right) \\ U_i|\kappa &\sim \mathcal{G}\left(\frac{\kappa}{2}, \frac{\kappa}{2}\right) \\ \boldsymbol{\theta} &\sim \mathcal{N}(\boldsymbol{\vartheta}, \boldsymbol{\Omega}) \\ \kappa &\sim \mathcal{E}(\lambda) \cdot \mathbb{I}(2, \infty) \\ \tau &\sim \mathcal{G}(a, b) \\ \lambda &\sim \mathcal{U}(c, d), \end{aligned}$$

Based on the priors specified within the model, the joint posterior distribution of $(\boldsymbol{\theta}, \boldsymbol{U}, \lambda, \kappa, \tau)$ is characterised by the following fully conditional distributions [23].

$$\boldsymbol{\theta} \sim \mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma}) \tag{3.6}$$

$$U_i \sim \mathcal{G}\left(\frac{\kappa}{2}+1, \frac{\kappa}{2}+\frac{\tau}{2}(Y_i-\boldsymbol{\theta}^*\boldsymbol{\xi}_i)^2\right) \tag{3.7}$$

$$\lambda \sim \mathcal{G}(2,\kappa)\cdot\mathbb{I}_{(c,d)} \tag{3.8}$$

$$\tau \sim \mathcal{G}\left(a+\frac{n}{2}, b+\frac{1}{2}\sum_{i=1}^{n} U_i(Y_i-\boldsymbol{\theta}^*\boldsymbol{\xi}_i)\right) \tag{3.9}$$

$$\kappa \sim \prod_{i=1}^{n} \frac{(\kappa/2)^{\frac{\kappa}{2}}}{\Gamma(\kappa/2)} U_i^{\frac{\kappa}{2}-1} \exp\left(-\frac{\kappa}{2}U_i\right) \exp\left(-\lambda\kappa\right) \mathbb{I}(2,\infty), \tag{3.10}$$

where $\boldsymbol{\mu}$ and $\boldsymbol{\Sigma}$ are given by (3.5).

To sample from the joint posterior, we combine a Gibbs sampler with a Metropolis-Hastings algorithm used to sample from the conditional posterior of κ. As explained earlier, the Gibbs sampler works by generating a series of values for $(\boldsymbol{\theta}, \boldsymbol{U}, \lambda, \kappa, \tau)$ by sampling each of the values from their fully conditional posterior distribution, taking into account the other relevant parameters used. For the standard distributions we use the predefined R functions, while for the non-standard distributions (in this case κ) we utilise a Metropolis-Hastings algorithm to generate appropriate values.

The Gibbs Sampling process for the Student-t case can be explained as follows:

Algorithm 2 Gibbs Sampling for the Student-t Distribution

1: Assign suitable values to the hyper-parameters $a, b, c, d, \boldsymbol{\vartheta}, \boldsymbol{\Omega}$.
2: Initialise the parameter vectors $(\boldsymbol{\theta}, \boldsymbol{U}, \lambda, \tau, \kappa)$ with suitable values.
3: **for** *iteration* $= 1, 2, \ldots,$ N **do**
4: Generate $\bar{\boldsymbol{\theta}} \sim \mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma})$, where $\boldsymbol{\mu}$ and $\boldsymbol{\Sigma}$ are given by (3.5)
5: $\boldsymbol{\theta} \leftarrow \bar{\boldsymbol{\theta}}$
6: Generate $\bar{\boldsymbol{U}}$ according to (3.7)
7: $\boldsymbol{U} \leftarrow \bar{\boldsymbol{U}}$
8: Generate $\bar{\lambda}$ according to (3.8)
9: $\lambda \leftarrow \bar{\lambda}$
10: Generate $\bar{\tau}$ according to (3.9)
11: $\tau \leftarrow \bar{\tau}$
12: Generate $\bar{\kappa}$ according to (3.10), using a MH method
13: $\kappa \leftarrow \bar{\kappa}$
14: **end for**
15: Discard burn-in samples.

The Metropolis-Hastings algorithm used to sample κ is conducted as follows:

Algorithm 3 Metropolis-Hastings for κ

1: Propose new value for $\bar{\kappa}$ using $\mathcal{E}(\lambda)\cdot\mathbb{I}(2,\infty)$.
2: Calculate the log-likelihood of the proposed $\bar{\kappa}$ derived from (3.10), denoted as $\pi(\bar{\kappa})$
3: Calculate the log-likelihood of the previous κ derived from (3.10), denoted as $\pi(\kappa)$
4: Accept proposal $\bar{\kappa}$ with probability min $\left(\frac{\pi(\bar{\kappa})}{\pi(\kappa)}, 1\right)$.
5: If accepted set $\kappa \leftarrow \bar{\kappa}$, otherwise keep previous value.

3.4 Slash Distribution

The hierarchical model used for the Slash distribution is similar to our model specified in the Student-t case, but the prior of U_i and truncation of κ differ.

We use the following hierarchical model for the Slash distribution [13]:

$$\begin{aligned} Y_i|U_i, \boldsymbol{\theta}, \sigma^2 &\sim \mathcal{N}\left(\boldsymbol{\theta}^*\boldsymbol{\xi}_i, \frac{1}{\sqrt{\tau U_i}}\right) \\ U_i|\kappa &\sim \mathcal{B}(\kappa, 1)\cdot\mathbb{I}(0,1) \\ \boldsymbol{\theta} &\sim \mathcal{N}(\boldsymbol{\vartheta}, \boldsymbol{\Omega}) \\ \kappa &\sim \mathcal{E}(\lambda) \\ \tau &\sim \mathcal{G}(a, b) \\ \lambda &\sim \mathcal{U}(c, d), \end{aligned}$$

Based on this model the joint posterior distribution of $(\boldsymbol{\theta}, \boldsymbol{U}, \lambda, \kappa, \tau)$ is characterised by the following fully conditional distributions:

$$\boldsymbol{\theta} \sim \mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma}) \tag{3.11}$$

$$U_i \sim \mathcal{G}\left(\frac{\kappa}{2}+1, \frac{\kappa}{2}+\frac{\tau}{2}(Y_i-\boldsymbol{\theta}^*\boldsymbol{\xi}_i)^2\right)\cdot\mathbb{I}(0,1) \tag{3.12}$$

$$\lambda \sim \mathcal{G}(2,\kappa)\cdot\mathbb{I}(c,d) \tag{3.13}$$

$$\tau \sim \mathcal{G}\left(a+\frac{n}{2}, b+\frac{1}{2}\sum_{i=1}^{n}U_i(Y_i-\boldsymbol{\theta}^*\boldsymbol{\xi}_i)\right) \tag{3.14}$$

$$\kappa \sim \mathcal{G}\left(n+1, \lambda-\sum_{i=1}^{n}\log(U_i)\right), \tag{3.15}$$

where $\boldsymbol{\mu}$ and $\boldsymbol{\Sigma}$ are given by (3.5).

This methodology was a departure from that outlined by Shan et al. (2020), who posit that the posterior of U_i must be truncated on $(1,\infty)$. However, recognising that the prior is only supported on the interval (0,1) the posterior must be also. As such, we have adapted the methodology to reflect this change. This instance shows the importance of a holistic review of innovative papers. As with the Student-t, we use a Gibbs Sampling approach to generate the values for $(\boldsymbol{\theta}, \boldsymbol{U}, \lambda, \kappa, \tau)$. As the fully conditional posteriors are all standard distributions, we can use only the predefined R functions, resulting in a simpler computation.

The Gibbs Sampling process for the Slash is as follows:

Algorithm 4 Gibbs Sampling for the Slash Distribution

1: Assign suitable values to hyper-parameters $a, b, c, d, \boldsymbol{\vartheta}, \boldsymbol{\Omega}$.
2: Initialise the parameter vectors $(\boldsymbol{\theta}, \boldsymbol{U}, \lambda, \tau, \kappa)$ with suitable values.
3: **for** *iteration* $= 1, 2, \ldots,$ N **do**
4: Generate $\bar{\boldsymbol{\theta}} \sim \mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma})$, where $\boldsymbol{\mu}$ and $\boldsymbol{\Sigma}$ are given by (3.5)
5: $\boldsymbol{\theta} \leftarrow \bar{\boldsymbol{\theta}}$
6: Generate $\bar{\boldsymbol{U}}$ according to (3.12)
7: $\boldsymbol{U} \leftarrow \bar{\boldsymbol{U}}$
8: Generate $\bar{\lambda}$ according to (3.13)
9: $\lambda \leftarrow \bar{\lambda}$
10: Generate $\bar{\tau}$ according to (3.14)
11: $\tau \leftarrow \bar{\tau}$
12: Generate $\bar{\kappa}$ according to (3.15)
13: $\kappa \leftarrow \bar{\kappa}$
14: **end for**
15: Discard burn-in samples.

3.5 Contaminated Normal Distribution

The model utilised for the Contaminated Normal distribution differs most dramatically from the hierarchical models mentioned previously. For this application, we consider the prior for U_i to be generated for a discrete probability distribution. Where U_i can take the values:

$$\begin{cases} U_i = 1, & w.p.\ 1-\kappa_1, \\ U_i = \kappa_2, & w.p.\kappa_1 \end{cases},$$

where the parameters $\kappa_1 \in [0,1]$ and $\kappa_2 \in [0,1]$.

The remainder of the hierarchical model implemented can be represented as follows:

$$\begin{aligned} Y_i|U_i, \boldsymbol{\theta}, \sigma^2 &\sim \mathcal{N}\left(\boldsymbol{\theta}^*\boldsymbol{\xi}_i, \frac{1}{\sqrt{\tau U_i}}\right) \\ \boldsymbol{\theta} &\sim \mathcal{N}(\boldsymbol{\vartheta}, \boldsymbol{\Omega}) \\ \kappa_1 &\sim \mathcal{U}(0,1) \\ \kappa_2 &\sim \mathcal{B}(c,d) \\ \tau &\sim \mathcal{G}(a,b), \end{aligned}$$

Based on the above model, the fully conditional distributions of $(\boldsymbol{\theta}, \boldsymbol{U}, \lambda, \tau, \kappa_1, \kappa_2)$ can be represented as follows, where $\boldsymbol{\mu}$ and $\boldsymbol{\Sigma}$ are given by (3.5):

$$\boldsymbol{\theta} \sim \mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma}) \tag{3.16}$$

$$\tau \sim \mathcal{G}\left(a + \frac{n}{2}, b + \frac{1}{2}\sum_{i=1}^{n} U_i(Y_i - \boldsymbol{\theta}^*\boldsymbol{\xi}_i)\right) \tag{3.17}$$

$$\kappa_1 \sim \mathcal{B}\left(1 + \frac{\sum_{i=1}^{n}(1-u_i)}{1-\kappa_2},\ 1 + \frac{\sum_{i=1}^{n}(u_i-\kappa_2)}{1-\kappa_2}\right), \tag{3.18}$$

The conditional posterior for U_i is given by

$$\begin{cases} U_i = 1, & w.p.\ 1-\pi, \\ U_i = \kappa_2, & w.p.\ \pi \end{cases}, \tag{3.19}$$

where the probability π can be computed as

$$\pi = \frac{\kappa_1\sqrt{\kappa_2}\cdot\exp\left(-\frac{\kappa_2\tau}{2}(Y_i - \boldsymbol{\theta} * \boldsymbol{\xi}_i)^2\right)}{\kappa_1\sqrt{\kappa_2}\cdot\exp\left(-\frac{\kappa_2\tau}{2}(Y_i - \boldsymbol{\theta} * \boldsymbol{\xi}_i)^2\right) + (1-\kappa_1)\exp\left(-\frac{\tau}{2}(Y_i - \boldsymbol{\theta} * \boldsymbol{\xi}_i)^2\right)}.$$

This approach once again deviates from that explained by Shan et al. (2020). Here we opted to combine a methodology explained by Garay et al. (2015), in their paper on Bayesian analysis of censored linear regression models [13]. This explanation is more easily accessible, than the work of Shan.

As with the student-t case, we need to implement a Metropolis-Hastings Algorithm alongside our Gibbs Sampling approach. In this case, we utilise the MH process to generate suitable values for $\bar{\kappa_2}$, where the proposal values are generated from a Beta distribution [23] utilising the chosen hyperparameters:

$$\bar{\kappa_2} \sim \mathcal{B}(c,d), \tag{3.20}$$

Given the independent sample $U_1 = u_1, \ldots, U_n = u_n$, $\bar{u_i}$ can take the values:

$$\begin{cases} \bar{u_i} = 1, & \text{if } u_i = 1, \\ \bar{u_i} = \bar{\kappa_2}, & \text{if } u_i = \kappa_2 \end{cases}, \tag{3.21}$$

We utilise the joint posterior distribution [23] of κ_2 and u given as:

$$p(u_1, \ldots, u_n, \kappa_2| \sim) \propto \left[\prod_{i=1}^{n} u_i\right]^{\frac{1}{2}} \kappa_1^{\frac{n-\sum u_i}{1-\kappa_2}} (1-\kappa_1)^{\frac{\sum u_i - n\kappa_2}{1-\kappa_2}}, \tag{3.22}$$

After generating our candidate values for κ_2 and u_i, we accept the values with probability:

$$\alpha = \min\{1, \frac{p(u_1', \ldots, u_n', \kappa_2')q(\kappa_2)}{p(u_1, \ldots, u_n, \kappa_2)q(\kappa_2')}\}, \tag{3.23}$$

where q denotes the Beta proposal density (from which the candidate value $\bar{\kappa}$ was sampled).

The Gibbs Sampling process is implemented as follows:

Algorithm 5 Gibbs Sampling for the CN Distribution

1: Assign suitable values to hyper-parameters $a, b, c, d, \boldsymbol{\vartheta}, \boldsymbol{\Omega}$.
2: Initialise the parameter vectors $(\boldsymbol{\theta}, \boldsymbol{U}, \lambda, \tau, \kappa)$ with suitable values.
3: **for** *iteration* $= 1, 2, \ldots,$ N **do**
4: Generate $\bar{\boldsymbol{\theta}} \sim \mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma})$, where $\boldsymbol{\mu}$ and $\boldsymbol{\Sigma}$ are given by (3.5)
5: $\boldsymbol{\theta} \leftarrow \bar{\boldsymbol{\theta}}$
6: Generate $\bar{\boldsymbol{U}}$ according to (3.19)
7: $\boldsymbol{U} \leftarrow \bar{\boldsymbol{U}}$
8: Generate $\bar{\kappa_1}$ according to (3.18)
9: $\kappa_1 \leftarrow \bar{\kappa_1}$
10: Generate $\bar{\tau}$ according to (3.17)
11: $\tau \leftarrow \bar{\tau}$
12: Generate $\bar{\kappa_2}$ according to (3.20), using MH method
13: $\kappa_2 \leftarrow \bar{\kappa_2}$
14: Generate $\bar{u}$ according to (3.21), using MH method
15: $u \leftarrow \bar{u}$
16: **end for**
17: Discard burn-in samples.

The Metropolis-Hastings algorithm used to sample κ_2 and u is conducted as follows:

Algorithm 6 Metropolis-Hastings for κ

1: Propose new value for $\bar{\kappa_2}$ according to (3.20)
2: Generate new u values, according to (3.21)
3: Calculate the log-likelihood of the proposed and previous κ_2 using the Beta distribution density function
4: Calculate the log-likelihood of the joint posterior of κ_2 and u, according to (3.22)
5: Accept proposal values according to (3.23)
6: If accepted set $\kappa_2 \leftarrow \bar{\kappa_2}$ and $u_i \leftarrow \bar{u_i}$, otherwise keep previous values.

Selection of Hyperparameters

The hyperparameters used for the models, $a, b, c, d, \boldsymbol{\vartheta}, \boldsymbol{\Omega}$, reflect our prior knowledge of the associated parameters. These hyperparameters inform the prior distributions of the parameters of interest. As we have no prior beliefs or experience, we use values that generate uninformative priors. We opted to use the values elaborated on by Shan et al. (2020) and Garay et al. (2015).

a, b: In every model, these hyperparameters are related to the prior distribution of τ, which is assumed to follow a Gamma distribution, $\mathcal{G}(a, b)$. For each implementation, we set $a = b = 0.001$ [23].

c, d: For both the Slash and Student-t, these hyperparameters represent the boundaries of the uniform prior distribution for λ, which is assumed to follow a Uniform distribution [13], $\mathcal{U}(c, d)$, as well as representing the truncation points of the posterior. In both cases $c = 0.001$, and $d = 1$. For the Contaminated Normal distribution, c and d represent the alpha and beta of our proposal κ_2 values, respectively. Following Shan et al. (2020), we set $c = d = 1$.

$\boldsymbol{\vartheta}$: This is the mean vector associated with the multivariate normal prior distribution of $\boldsymbol{\theta}$, $\mathcal{N}(\boldsymbol{\vartheta}, \boldsymbol{\Omega})$. We chose to set this equal to 1 initially, and have it vary by each successive simulation.

$\boldsymbol{\Omega}$: This represents the covariance matrix for the multivariate normal prior distribution for $\boldsymbol{\theta}$. To reflect our uninformed prior, this was generated as a matrix with diagonals equal to 1000 and all else

set to 0. This results in a large variance and independence of the parameters due to the 0 correlation [13].

Burn-in and Convergance

As the goal of an MCMC implementation is to pull values from a stationary distribution, we had to ensure the number of iterations was large enough and that we removed the samples given before convergence (burn-in). As such, we set each model to run for 50,000 iterations, with a burn-in of 10,000 following Garay et al. (2015), which was shown to result in a stationary distribution for all the parameters of interest.

To check for convergence, we utilised trace plots of the parameters of interest, as evidenced below:

Model Selection

There are multiple criteria by which we can assess the goodness of fit of a model such as; Akaike Information Criterion (AIC) and Bayes Information Criterion (BIC), often used in traditional statistical analysis. In a Bayesian context, the Deviance Information Criterion (DIC) is a popular measure. This criterion works as both a measure of model fit and complexity [13]. Due to the computational complexity of calculating the true DIC, we use a suitable approximation utilising our MCMC samples after the burn-in period[13], which is given as follows:

$$\widehat{DIC} = 2\bar{D} - D(\hat{\theta}),$$

where $\bar{D}$ is the posterior mean of the deviance and $D(\hat{\theta})$ is the deviance evaluated at the posterior mean parameter estimate[13]. The lower the value of the DIC, the better the model fits the data accounting for the model's complexity.

We also choose to use the Extended Akaike Information Criterion (EAIC) and the Extended Bayesian Information Criterion (EBIC), which can be thought of as an extension of the DIC. The EAIC aims to balance goodness of fit and model complexity, whilst the EBIC has a harsher punishment for model complexity. The EAIC can be represented as:

$$\text{EAIC} = \bar{D} + 2 \times p,$$

where $\bar{D}$ is the posterior mean of the deviance, and p is the number of parameters in the model [12].

The EBIC can be represented as:

$$\text{EBIC} = \bar{D} + p \times \log(n),$$

where $\bar{D}$ represents the posterior mean of the deviance, p represents the number of parameters in the model, and n represents the size of data [12]. The key difference in the EBIC is the addition of the log term, penalising the complexity of models more if the sample size is larger. As with DIC, the lower the value for both these measures, the better the model fit.

Another popular comparison criterion is the Log Pseudo-Marginal Likelihood (LPML). This aims to provide a way to measure the predictive accuracy of a given model. We can think of LPML as a variation of cross-validation, as it seeks to quantify how the model performs on unseen data [12]. The formula for the LMPL is given as:

$$\text{LPML} = \sum_{i=1}^{n} \log\left(CPO_i\right),$$

where CPO_i stands for the Conditional Predictive Ordinate of the i^{th} observation. The CPO_i measures the out-of-sample predictive fit of the model [12]. Which is given by the formula:

$$CPO_i = \frac{1}{Q} \sum_{q=1}^{Q} \left(\frac{1}{\pi(y_i|\theta_q)} \right)^{-1}.$$

This formula is an approximation we can get using the MCMC samples. Unlike other measures, the higher the LPML, the better. A higher LPML value indicates that the model is able to generalise to unseen data better.

Our last criterion of interest is the Watanabe-Akaike Information Criterion (WAIC). Like the LPML, it measures out-of-sample expectation [15]. There are two approaches to calculate the WAIC [12]; we utilise the first in this paper. To calculate this, we first calculate the log pointwise posterior predictive density [15], which we denote as $p(z)$. The formula for this is given as:

$$p(z) = \sum_{i=1}^{n} \log \left(\frac{1}{Q} \sum_{j=1}^{Q} \pi(z_i|\theta_j) \right). \tag{3.24}$$

This, once again, is an approximation that allows us to draw upon our MCMC samples. We can then calculate the first version of WAIC using this and our previous calculation utilised in the DIC [12]. The formula is given as:

$$WAIC_1 = 2p(z) + 2\bar{D}. \tag{3.25}$$

This extensive use of model selection criteria will allow us to further scrutinise the models built upon the work of Shan et al. (2020).

To further validate our models in the simulation studies, we utilise both the Mean Squared Error (MSE) and bias to compare model performance. The formula for both is given as:

$$\text{MSE} = \frac{1}{n} \sum_{i=1}^{n} (\beta_i - \hat{\beta}_i)^2, \tag{3.26}$$

where $\hat{\beta}_i$ represents the estimated value, β_i represents the true value, and n is the sample size. The bias can be calculated as follows:

$$\text{Bias} = \frac{1}{n} \sum_{i=1}^{n} (\beta_i - \hat{\beta}_i). \tag{3.27}$$

We use these metrics to evaluate the accuracy and consistency of our models. A lower MSE indicates that a model more accurately captures the data, which is what we want. Bias tells us if the model tends to overestimate or underestimate the parameter values. Ideally, we want a bias as close to zero as possible.

Chapter 4

Simulated Datasets

4.1 Simulated Dataset 1

We generate a simulated functional dataset to test the hierarchical models' performance. We allow the errors of the simulated data to follow non-normal distributions, allowing for heavy tails. We use this study to see how our models handle the heavy-tailed data. We calculate the bias and MSE of our parameter estimates to analyse our model performance. We use DIC, EAIC, EBIC, LMPL, and WAIC1 as relative performance measures.

4.1.1 Data Generation

The response values are generated from the following functional linear regression:

$$Y = \alpha Z_1 + \int_0^1 \beta(t)X(t)dt + \epsilon, \tag{4.1}$$

where:

$$\begin{aligned} \alpha &= 3 \\ Z_1 &\sim \mathcal{U}(-1,1) \\ \beta(t) &= \sqrt{3}\sum_{j=1}^{50} \frac{1}{j^4}\sin(j\pi t) \\ X(t) &= \sqrt{3}\sum_{j=1}^{50} \eta_j \sin(j\pi t) \\ \eta_j &\sim N(0, 2/j), \end{aligned}$$

where $\beta(t)$ is the slope function, α is our non-functional predictor, Z_1 is our non-functional data vector, as explained in Section 2.2, drawn from the Uniform distribution, $X(t)$ is the functional predictor, and η_j represents random coefficients that follow a normal distribution [23]. This approach is an adaptation of the work of Shan et al. (2020) and Cai and Hall (2020) [6, 23]. The error term ϵ is generated based on a specified distribution: Normal, Contaminated Normal (CN), Student-t, Mixture, or Cauchy. We use these five methods to generate errors as they will all uniquely impact the distribution of the simulated data and the generation of heavy-tails. The errors are generated as follows:

$$\epsilon \sim \begin{cases} N(0,1) & \text{for Normal} \\ t(2) & \text{for Student-t} \\ \text{Cauchy}(0,1) & \text{for Cauchy} \\ 0.9 \cdot N(0,1) + 0.1 \cdot N(0,100) & \text{for Contaminated Normal (CN)} \\ 0.7 \cdot N(0,1) + 0.2 \cdot N(0,10) + 0.1 \cdot N(0,100) & \text{for Mixture} \end{cases} \tag{4.2}$$

To be able to work with the data, we perform functional principal component analysis (FPCA) on the basis expansion of the functional predictor $X(t)$. There is a nuanced debate around how to tackle the problem of choosing the number of basis for smoothing your data. In our case, we approached this using a two-fold technique. Firstly, we generated a sequence of potential values, choosing the value that gave the lowest selection criteria scores, and we then confirmed the appropriateness of this selection by plotting the true curves against the smoothed data.

For this simulated dataset, we kept enough FPCs required to explain over 90% of the variation in the original data. This was accomplished by using a cumulative variance plot, shown in Figure 4.1.

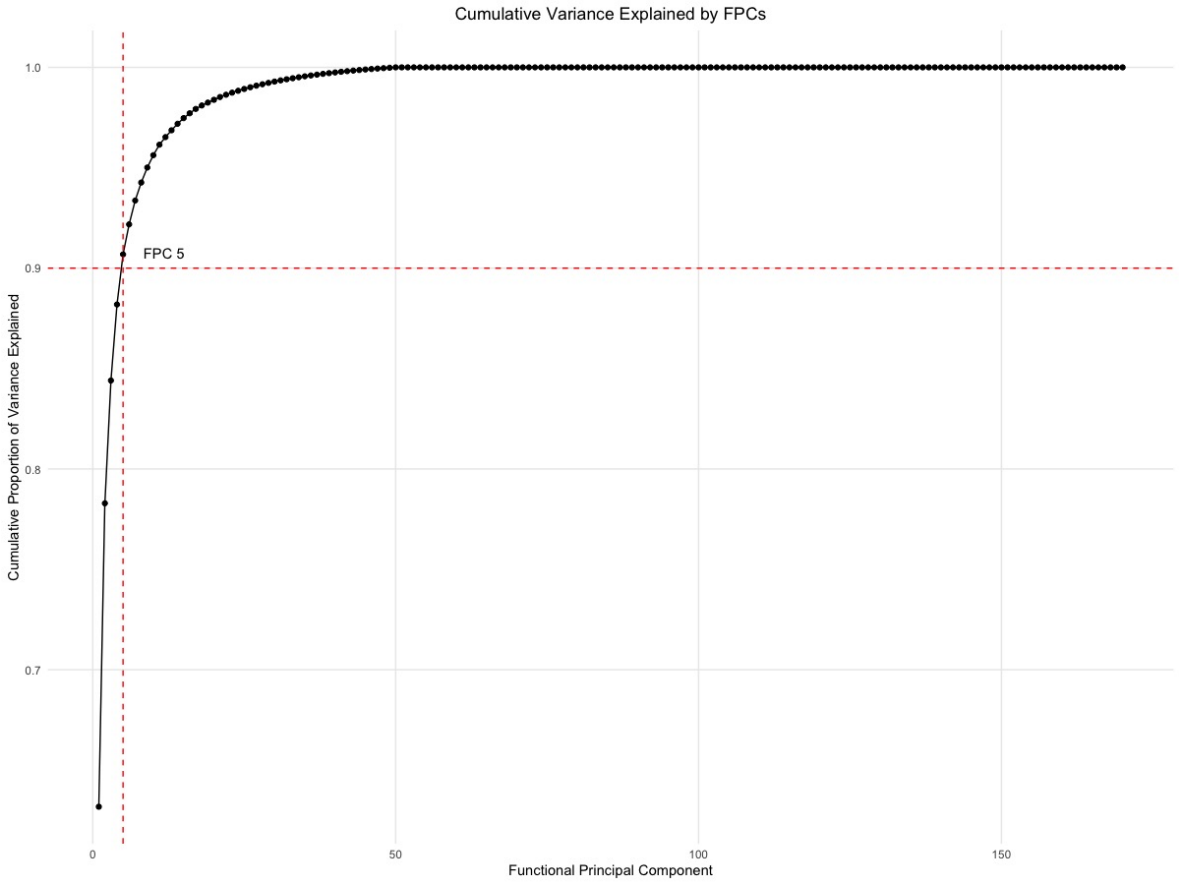


Figure 4.1: Cumulative proportion of variance explained, using simulated data 1 with Normal errors

4.1.2 Results

Two sets of simulated data were generated using the approach explained in Section 4.1.1, differing by sample size. We used sample sizes of 100 and 500, respectively. We compared the performance of our four models: Normal, Student-t, CN, and Slash. Each model's relative performance was evaluated using the selection criterion outlined in the methodology.

Looking at the performance of the models for sample size 100 (Table 4.1), we can clearly see that the models perform differently based on the error distribution specified. When the errors were derived from the normal distribution, as we would expect, the Normal model performed the best for the majority of selection criteria. Interestingly, however, the Normal model did not perform the best on the LPML selection criterion. As explained in the methodology, this criteria can be thought of as an out-of-sample prediction error technique. This result indicates that the CN model would perform better at this task. This outcome could be the result of sample size. As such, the larger sample size results will be interesting to compare. However, the difference is low. For the Student-t distributed errors, the Student-t model was shown to be the best. Interestingly, the Student-t model also appeared to perform the best using the other error distributions. Our theory is that as Student-t allows for the largest tails,

Error	Model	DIC	EAIC	EBIC	LPML	WAIC1
Normal	Normal	**293.91**	**300.84**	**319.08**	-147.56	**286.84**
Normal	Student-t	299.21	306.38	327.22	-150.81	291.10
Normal	CN	294.27	305.04	328.49	**-147.14**	287.04
Normal	Slash	315.88	322.10	342.94	-156.95	306.10
t	Normal	453.10	460.03	478.27	-230.68	446.03
t	Student-t	**395.97**	**404.43**	**425.27**	**-199.63**	**388.42**
t	CN	449.62	458.28	481.73	-236.30	445.08
t	Slash	452.04	454.69	475.53	-222.61	438.69
Cauchy	Normal	819.78	826.96	845.20	-421.30	812.96
Cauchy	Student-t	**527.61**	**536.25**	**557.09**	**-265.17**	**520.22**
Cauchy	CN	828.42	822.58	846.02	-411.48	854.29
Cauchy	Slash	797.42	788.55	809.39	-390.18	772.55
CN	Normal	728.04	734.97	753.21	-368.02	720.97
CN	Student-t	**544.81**	**553.46**	**574.30**	**-273.70**	**537.44**
CN	CN	726.81	734.91	758.35	-430.79	720.04
CN	Slash	721.56	723.12	743.96	-356.29	707.12
Mixture	Normal	977.52	984.54	1002.77	-495.45	970.54
Mixture	Student-t	**678.66**	**687.21**	**708.05**	**-341.02**	**671.19**
Mixture	CN	982.58	983.11	1006.55	-545.18	984.04
Mixture	Slash	959.31	958.80	979.65	-474.19	942.80

Table 4.1: Model fit criteria results for sample size of 100. The bold values indicate 'best' performance for given error.

given our models, this will often perform the best when there are many outliers or extremely heavy tails. The larger tails of the Student-t distribution can be seen in Figure 2.1.

The results were almost identical for the larger sample size of 500 (Table 4.3). Once again, the Student-t model seems to perform the best. As we can see, the models that allow for the heavy-tailed data perform better than the Normal, which does not. This finding is consistent with the results gathered by Shan et al. (2020), with the added benefit of greater selection criteria and the application to new data. Interestingly, the Normal model once again performed worse than the CN model when looking at the LPML, even after increasing the sample size.

To improve upon the robustness of our analysis, we also calculate the Mean-Squared Error (MSE) and bias for the parameters β and α. This provides insight into the accuracy and consistency of each model (Tables 4.2 and Table 4.4).

For the sample size of 100, under the Normal errors, the results were inconsistent. Although all models gave relatively low and close values, there was no clear 'best' performer. Also, the Normal model did not perform the best once. Overall, the Student-t model outperformed, giving the lowest MSE and bias across a range of error types.

These trends were generally consistent for the sample size of 500. As we increased the sample size, the results became more consistent. For example, the Student-t model now also gave the lowest bias for beta, suggesting as we increase sample size, there is more stability in parameter estimation.

Overall, the results clearly demonstrate that by creating models that can accommodate heavy-tailed distributions, we can improve the models' performance and the robustness of our inferences. Whilst some models performed best for certain error types, the Student-t model seemed to be the best in terms of model fit and predictive accuracy in general. The models were run 50 times, differing the starting values for beta, and the values were averaged. As stated in the methodology, each model ran for 50,0000 iterations, with a burn-in of 10,000.

		MSE		Bias	
Error	Model	Beta	Alpha1	Beta	Alpha1
Normal	Normal	0.04180	0.06272	0.00560	-0.03183
Normal	Student-t	0.04294	**0.05801**	**0.00517**	-0.03024
Normal	CN	0.04171	0.06257	0.00559	-0.03191
Normal	Slash	**0.04156**	0.07901	0.00554	**-0.02973**
t	Normal	0.26443	0.42726	0.00218	0.00783
t	Student-t	**0.07477**	**0.10271**	-0.00954	0.00884
t	CN	0.17275	0.31125	-0.00188	**-0.00500**
t	Slash	0.26715	0.50454	**-0.00014**	0.01095
Cauchy	Normal	57.43384	95.96784	-0.35051	0.89504
Cauchy	Student-t	**0.13495**	**0.18910**	**0.00509**	**0.04973**
Cauchy	CN	22.69951	64.05803	-0.23702	0.54756
Cauchy	Slash	52.61551	95.16179	-0.34167	0.78723
CN	Normal	3.93864	6.00629	0.08723	0.26042
CN	Student-t	**0.08316**	**0.14367**	**0.01638**	**-0.00653**
CN	CN	2.84517	4.74337	0.07005	0.18087
CN	Slash	3.92660	7.59189	0.08318	0.28313
Mixture	Normal	37.23576	73.34255	0.42486	1.34950
Mixture	Student-t	**0.13004**	**0.28030**	**-0.00965**	**-0.01036**
Mixture	CN	20.38357	48.82277	0.24360	1.15273
Mixture	Slash	36.91908	90.51767	0.42399	1.37861

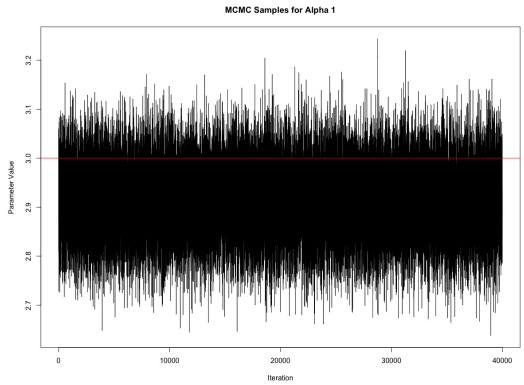
Table 4.2: MSE and Bias of parameters for sample size of 100. The bold values indicate lowest values for given error.

Error	Model	DIC	EAIC	EBIC	LPML	WAIC1
Normal	Normal	**1426.15**	**1433.14**	**1462.64**	-713.61	**1419.14**
Normal	Student-t	1430.16	1437.79	1471.50	-715.53	1422.41
Normal	CN	1426.40	1437.26	1475.19	**-713.18**	1419.26
Normal	Slash	1508.84	1515.26	1548.97	-753.50	1499.26
t	Normal	2451.49	2458.49	2487.99	-1241.16	2444.49
t	Student-t	**1973.25**	**1981.86**	**2015.57**	**-987.98**	**1965.95**
t	CN	2389.85	2384.73	2422.66	-1168.85	2476.96
t	Slash	2309.23	2302.85	2336.57	-1147.73	2286.85
Cauchy	Normal	4823.03	4830.27	4859.78	-2447.06	4816.27
Cauchy	Student-t	**2604.10**	**2613.02**	**2646.74**	**-1303.13**	**2597.02**
Cauchy	CN	4934.42	4808.91	4846.84	-2250.04	5339.90
Cauchy	Slash	4532.87	4483.45	4517.16	-2243.12	4467.45
CN	Normal	3620.91	3627.90	3657.40	-1814.02	3613.90
CN	Student-t	**2684.56**	**2693.50**	**2727.21**	**-1343.30**	**2677.49**
CN	CN	3604.73	3616.66	3654.59	-1956.27	3604.72
CN	Slash	3519.82	3519.86	3553.58	-1754.28	3503.86
Mixture	Normal	4873.49	4880.50	4910.00	-2443.87	4866.50
Mixture	Student-t	**3338.03**	**3346.94**	**3380.66**	**-1670.35**	**3330.94**
Mixture	CN	4875.89	4857.41	4895.34	-2693.49	4938.70
Mixture	Slash	4699.25	4697.20	4730.91	-2343.19	4681.20

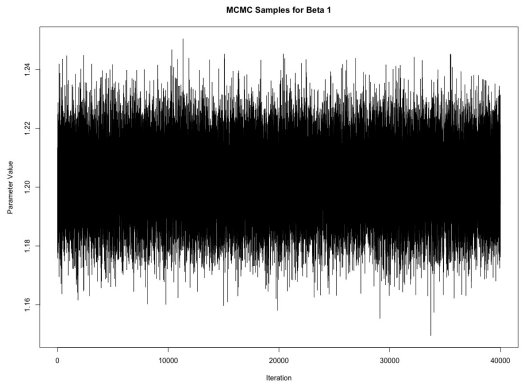
Table 4.3: Model fit criteria results for sample size of 500. The bold values indicate 'best' performance for given error.

		MSE		Bias	
Error	Model	Beta	Alpha1	Beta	Alpha1
Normal	Normal	0.00812	0.01036	0.00174	0.00230
Normal	Student-t	0.00837	**0.01022**	0.00153	**0.00217**
Normal	CN	0.00812	0.01037	0.00173	0.00236
Normal	Slash	**0.00804**	0.01310	**0.00147**	0.00298
t	Normal	0.07848	0.13022	0.00895	-0.03483
t	Student-t	**0.01488**	**0.02293**	**0.00458**	0.01589
t	CN	0.03963	0.08454	0.00724	**-0.00197**
t	Slash	0.08055	0.15525	0.00903	-0.03622
Cauchy	Normal	77.92732	103.39868	0.69665	1.61452
Cauchy	Student-t	**0.02491**	**0.03242**	**0.00129**	**0.00366**
Cauchy	CN	27.74261	69.93759	0.42352	1.17903
Cauchy	Slash	73.11541	103.30319	0.68547	1.50061
CN	Normal	0.75642	0.94535	0.02736	-0.08584
CN	Student-t	**0.01239**	**0.02203**	**0.00587**	**-0.00665**
CN	CN	0.62301	0.82638	0.02390	-0.06933
CN	Slash	0.76438	1.23369	0.02620	-0.09248
Mixture	Normal	8.97402	10.32738	-0.14714	-0.01839
Mixture	Student-t	**0.01989**	**0.04078**	**-0.00244**	**-0.00278**
Mixture	CN	4.92881	7.33527	-0.12975	-0.10398
Mixture	Slash	9.08345	13.19630	-0.14454	-0.00661

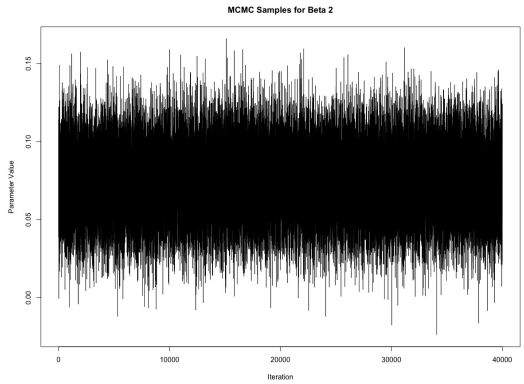
Table 4.4: MSE and Bias of parameters for sample size of 500. The bold values indicate lowest values for given error.



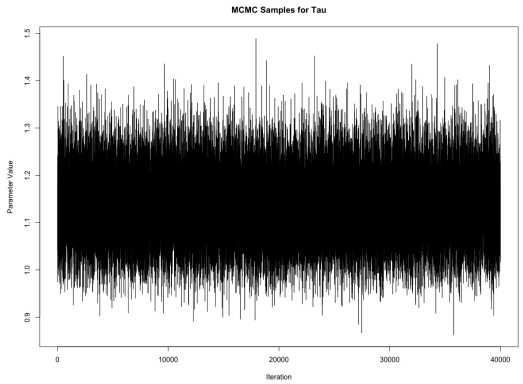
(a) Trace plot for Alpha 1



(b) Trace plot for Beta 1



(c) Trace plot for Beta 2



(d) Trace plot for Tau

Figure 4.2: Trace plots of select parameters for simulated dataset 1: Incorporating Normal errors with a sample size of 500 using the Normal model

4.2 Simulated Dataset 2

To test our models' performance again, we generate a new simulated dataset. We use the same metrics as earlier to compare relative performance. This new dataset will allow us to evaluate how our models handle heavy tails for a variety of data.

4.2.1 Data Generation

The response values are generated from the following semi-functional linear regression:

$$Y = \alpha_1 Z_1 + \alpha_2 Z_2 + \int_0^1 \beta(t)X(t)dt + \epsilon, \tag{4.3}$$

where:

$$\begin{aligned}
\alpha_{\mathbf{1}} &= 2 \\
\alpha_{\mathbf{2}} &= 3 \\
Z_1 &\sim \mathcal{U}(0,1) \\
Z_2 &\sim \mathcal{B}(3,2) \\
\beta(t) &= \sum_{j=1}^{50} \frac{1}{1+j} t^{(j-1)} \\
X(t) &= \sum_{j=1}^{50} \eta_j t^{(j-1)} \\
\eta_j &\sim N(0,1),
\end{aligned}$$

where $\beta(t)$ is the slope function derived from a polynomial function, α_1 and α_2 are our non-functional predictors, Z_1 and Z_2 are our non-functional data vectors, with values generated as explained above, $X(t)$ is the functional predictor (now generated by a polynomial basis expansion), and η_j represents the random coefficients.

As with the previously simulated data, the error term ϵ is generated based on a specified distribution: Normal, Contaminated Normal (CN), Student-t, Mixture, or Cauchy. The errors are generated as follows:

$$\epsilon \sim \begin{cases}
N(0,1) & \text{for Normal} \\
t(2) & \text{for Student-t} \\
\text{Cauchy}(0,1) & \text{for Cauchy} \\
0.9 \cdot N(0,1) + 0.1 \cdot N(0,100) & \text{for Contaminated Normal (CN)} \\
0.7 \cdot N(0,1) + 0.2 \cdot N(0,10) + 0.1 \cdot N(0,100) & \text{for Mixture}
\end{cases} \tag{4.4}$$

We again, perform functional principal component analysis on the simulated dataset. We used the same approach to choose the number of basis for our B-Splines as in simulation 1. Based on the cumulative variance explained plot, we chose to incorporate 4 FPCs into our model. This explained over 99% of the variance, as shown in Figure 4.3.

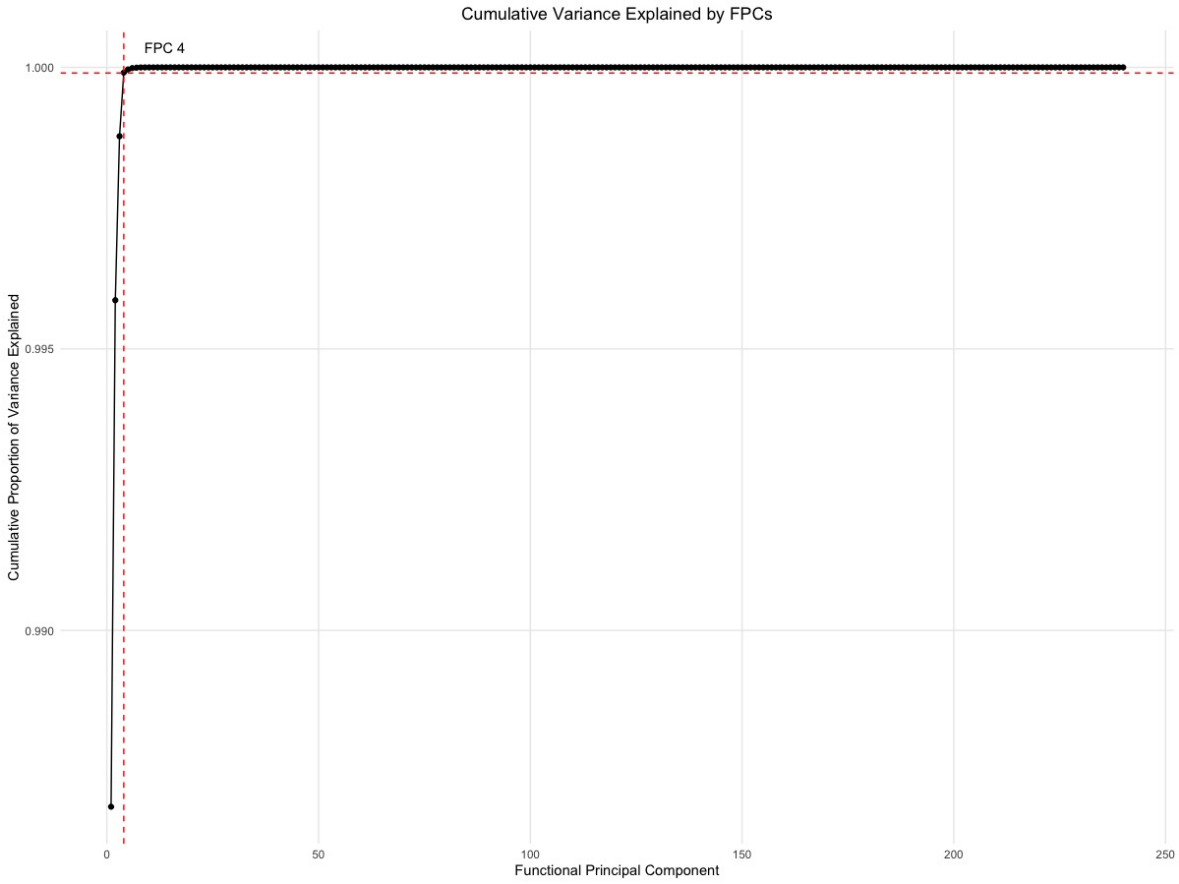


Figure 4.3: Cumulative proportion of variance explained, using simulated data 2 with Normal errors

4.2.2 Results

Error	Model	DIC	EAIC	EBIC	LPML	WAIC1
Normal	Normal	**288.08**	**295.01**	**313.25**	-144.66	**281.01**
Normal	Student-t	293.33	300.54	321.38	-147.97	285.22
Normal	CN	288.43	299.20	322.65	**-144.25**	281.20
Normal	Slash	310.04	316.31	337.15	-154.07	300.31
t	Normal	451.12	458.09	476.32	-230.20	444.09
t	Student-t	**393.47**	**401.86**	**422.71**	**-198.49**	**385.86**
t	CN	447.52	455.62	479.07	-227.54	446.25
t	Slash	450.45	452.90	473.74	-221.85	436.90
Cauchy	Normal	859.84	867.74	885.98	-443.19	853.74
Cauchy	Student-t	**535.32**	**543.99**	**564.83**	**-268.97**	**527.95**
Cauchy	CN	873.26	864.83	888.28	-425.85	908.83
Cauchy	Slash	837.80	824.26	845.10	-408.18	808.26
CN	Normal	515.22	522.19	540.43	-263.00	508.19
CN	Student-t	**398.53**	**407.17**	**428.01**	**-200.63**	**391.19**
CN	CN	512.76	519.11	542.56	-305.42	507.39
CN	Slash	502.54	504.04	524.88	-246.78	488.04
Mixture	Normal	974.28	982.44	1000.68	-494.14	968.44
Mixture	Student-t	**670.34**	**678.93**	**699.77**	**-336.77**	**662.90**
Mixture	CN	980.20	981.30	1004.75	-545.01	986.16
Mixture	Slash	951.98	952.57	973.42	-470.75	936.57

Table 4.5: Model fit criteria results for sample size of 100. The bold values indicate 'best' performance for given error.

We conduct the same tests as with simulated dataset 1. We used the same sample sizes and evaluation criteria. The results of the selection criteria for sample size 100 are given in Table 4.5. Once again, it is clear to see that the model fit varies significantly depending on both the error structure and the model type. When the simulated data has normally distributed errors, the lowest DIC, EAIC, and EBIC are given by the Normal model. However, once again, the highest LPML (indicating the best fit) was awarded to the CN model, a common theme. Also, for the Student-t errors, the Student-t model seems to perform the best in this case, giving the lowest DIC, EAIC, and EBIC. Notably, for the Cauchy error, the Student-t model again stands out with considerably lower DIC, EAIC, and EBIC values compared to other models, once again showing its strong performance. For the larger sample size, shown in Table 4.7, the outcomes remained the same.

As with simulation 1, we calculate the MSE and bias for all our parameters of interest. For the sample size of 100 (shown in Table 4.6), the Student-t model again outperforms in the majority of cases. We witness inconsistency in which models give the lowest MSE and bias using the normal errors. When using a sample size of 500 (shown in Table 4.8) and normal errors, the consistency has improved. However, the Student-t takes the spotlight once again, performing the best in the majority of cases.

In conclusion, these results emphasise the significance of understanding the underlying distribution of the data when conducting a semi-functional linear regression. These results clearly show the effectiveness of the proposed methodologies to better handle outliers in data. These findings once again align with that of Shan et al. (2020) while also building upon it. Here, we clearly illustrate the versatility of these models and their ability to generate robust inferences. Once again, we ran 50 simulations for 50,000 iterations each and a burn-in of 10,000. To validate that our models converged, we utilised trace plots. In Figure 4.4, trace plots for selected parameters generated from the simulated data using the CN errors can be seen. From these plots, convergence is clear. The parameters Beta 1-3 represent the unknown parameters of interest in relation to our data matrix that relate to the truncated basis expansion of the functional component Beta.

		MSE			Bias		
Error	Model	Beta	Alpha1	Alpha2	Beta	Alpha1	Alpha2
Normal	Normal	5.66006	**0.18294**	**0.15398**	-0.00802	0.00778	-0.00836
Normal	Student-t	6.00380	0.19375	0.16639	**-0.00797**	-0.01046	**0.00471**
Normal	CN	5.65822	0.18298	0.15401	-0.00805	0.00693	-0.00775
Normal	Slash	**5.64559**	0.22963	0.19209	-0.00822	**0.00635**	-0.00900
t	Normal	18.00675	1.47354	1.53084	-0.00530	0.01196	0.17421
t	Student-t	**7.74410**	**0.35598**	**0.29714**	**-0.00193**	**0.01940**	**0.01221**
t	CN	11.69848	1.01005	0.95163	-0.00643	0.00236	0.11239
t	Slash	17.37013	1.78545	1.87860	-0.00510	0.00799	0.17543
Cauchy	Normal	392.24109	143.29747	123.03847	1.42768	1.65847	-0.88215
Cauchy	Student-t	**11.90584**	**0.65923**	**0.48913**	**-0.00552**	**0.09195**	**-0.02781**
Cauchy	CN	194.07153	103.60319	89.16795	0.82869	1.12865	-0.53346
Cauchy	Slash	348.17512	145.98115	123.19851	1.34345	1.57547	-0.82141
CN	Normal	41.64869	2.22576	1.58888	-0.03816	0.17421	-0.20238
CN	Student-t	**5.60435**	**0.32670**	**0.23881**	**-0.00598**	**0.06226**	**-0.07997**
CN	CN	28.27245	1.62611	1.17585	-0.03045	0.16287	-0.17445
CN	Slash	40.07772	2.73279	1.99960	-0.03554	0.20752	-0.21339
Mixture	Normal	465.27795	143.09403	134.17729	0.03520	1.08112	-1.16381
Mixture	Student-t	**11.61362**	**0.94252**	**0.71426**	**-0.00110**	**0.07340**	**-0.03697**
Mixture	CN	326.82007	102.37095	96.00872	0.04467	0.91410	-0.85325
Mixture	Slash	416.88636	173.19317	159.18005	0.02297	0.90305	-0.98129

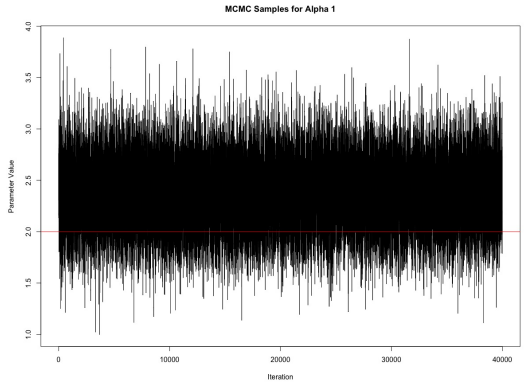
Table 4.6: MSE and Bias of parameters for sample size of 100. The bold values indicate lowest values for given error.

Error	Model	DIC	EAIC	EBIC	LPML	WAIC1
Normal	Normal	**1420.14**	**1427.13**	**1456.63**	-710.59	**1413.13**
Normal	Student-t	1424.38	1431.91	1465.63	-712.68	1416.55
Normal	CN	1420.41	1431.27	1469.20	**-710.16**	1413.27
Normal	Slash	1503.54	1509.95	1543.67	-750.83	1493.95
t	Normal	2350.64	2357.63	2387.13	-1187.02	2343.63
t	Student-t	**1951.72**	**1960.26**	**1993.98**	**-977.24**	**1944.38**
t	CN	2295.37	2296.62	2334.55	-1140.35	2356.28
t	Slash	2229.65	2225.84	2259.56	-1108.86	2209.84
Cauchy	Normal	4863.90	4871.62	4901.13	-2471.12	4857.62
Cauchy	Student-t	**2597.11**	**2606.04**	**2639.76**	**-1299.59**	**2590.03**
Cauchy	CN	4986.72	4849.57	4887.50	-2247.12	5429.69
Cauchy	Slash	4561.36	4504.31	4538.02	-2254.55	4488.31
CN	Normal	2610.60	2617.60	2647.10	-1311.49	2603.60
CN	Student-t	**1936.59**	**1945.47**	**1979.19**	**-969.61**	**1929.50**
CN	CN	2591.17	2587.57	2625.50	-1483.68	2618.50
CN	Slash	2460.45	2460.27	2493.98	-1224.65	2444.27
Mixture	Normal	4904.06	4911.57	4941.07	-2459.31	4897.57
Mixture	Student-t	**3322.69**	**3331.61**	**3365.33**	**-1662.63**	**3315.61**
Mixture	CN	4909.16	4889.92	4927.86	-2713.71	4976.56
Mixture	Slash	4726.25	4725.06	4758.78	-2357.06	4709.06

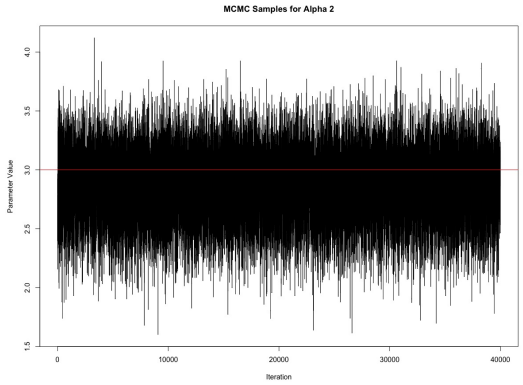
Table 4.7: Model fit criteria results for sample size of 500. The bold values indicate 'best' performance for given error.

		MSE			Bias		
Error	Model	Beta	Alpha1	Alpha2	Beta	Alpha1	Alpha2
Normal	Normal	0.83521	0.04151	0.03296	0.00021	-0.01897	0.01253
Normal	Student-t	0.85650	**0.04103**	**0.03198**	**0.00003**	**-0.01565**	**0.01159**
Normal	CN	**0.83301**	0.04151	0.03294	0.00020	-0.01856	0.01249
Normal	Slash	0.83443	0.04968	0.03978	0.00010	-0.01797	0.01210
t	Normal	5.03176	0.31381	0.23455	0.00272	0.05202	-0.03291
t	Student-t	**0.78899**	**0.05821**	**0.05376**	**0.00157**	**0.02840**	**-0.01572**
t	CN	2.80881	0.20781	0.16182	0.00218	0.03308	-0.02156
t	Slash	5.03287	0.36858	0.27229	0.00288	0.04950	-0.03001
Cauchy	Normal	180.22230	92.32448	93.54095	0.72245	0.57098	-0.48777
Cauchy	Student-t	**1.66418**	**0.10161**	**0.07916**	**-0.00091**	**-0.01177**	**-0.00285**
Cauchy	CN	77.21145	60.70760	58.76353	0.42668	0.34804	-0.31149
Cauchy	Slash	171.97624	92.95790	95.07802	0.71224	0.56287	-0.48566
CN	Normal	7.15638	0.56373	0.39455	0.00291	0.10813	-0.15350
CN	Student-t	**0.90195**	**0.04708**	**0.03707**	**0.00004**	**0.00350**	**-0.01142**
CN	CN	4.56249	0.39131	0.27867	0.00048	0.06877	-0.09956
CN	Slash	7.13984	0.65956	0.47105	0.00283	0.10828	-0.15465
Mixture	Normal	339.23739	33.56390	30.92789	**-0.00009**	-0.52715	0.46599
Mixture	Student-t	**1.28877**	**0.11627**	**0.10113**	-0.00426	**-0.02766**	**0.04032**
Mixture	CN	195.52449	24.17357	22.00309	-0.00120	-0.39826	0.51033
Mixture	Slash	335.79708	43.42831	38.63558	0.00366	-0.58478	0.48563

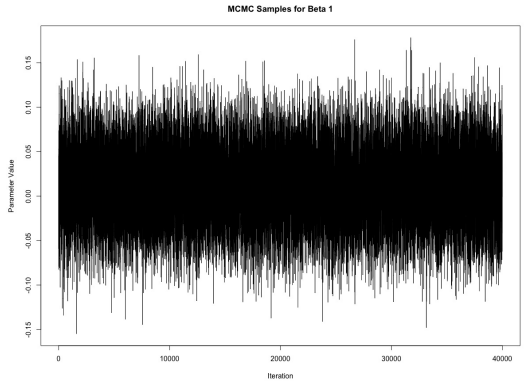
Table 4.8: MSE and Bias of parameters for sample size of 500. The bold values indicate lowest values for given error.



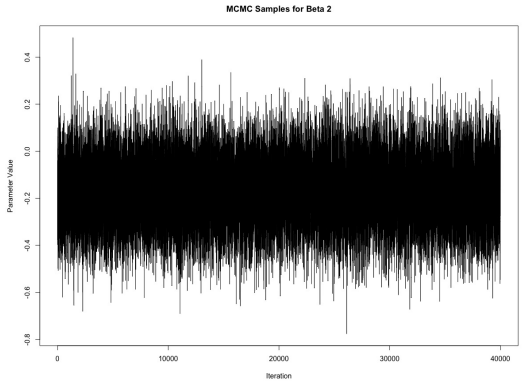
(a) Trace plot for Alpha 1



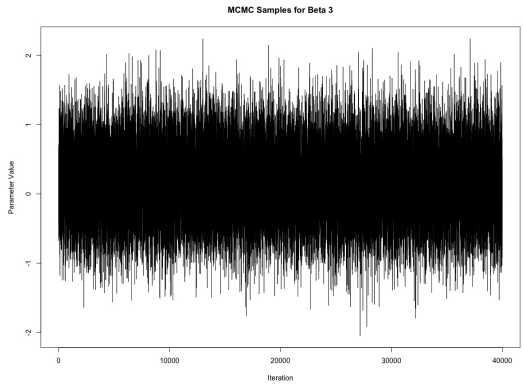
(b) Trace plot for Alpha 2



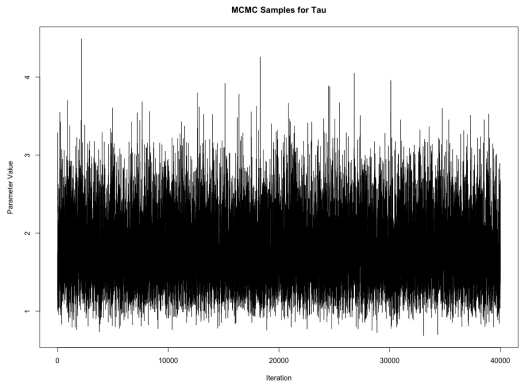
(c) Trace plot for Beta 1



(d) Trace plot for Beta 2



(e) Trace plot for Beta 3



(f) Trace plot for Tau

Figure 4.4: Trace plots of select parameters for simulated dataset 2: Incorporating CN errors with a sample size of 100 using the Student-t model

Chapter 5

Real-world Data Application

5.1 Real-Data Analysis

This section aims to apply our models to a real-world functional dataset. Using the methodologies outlined earlier, we aim to evaluate the performance of these models. However, as this is a real-data analysis, we can no longer calculate the MSE and bias for the parameters, as with the simulated data. As such, we generate a table of the parameter posterior mean values, standard deviations, and the highest posterior density (HPD) interval with a 95% confidence level. The models' relative performance is compared using DIC, EAIC, EBIC, LMPL, and WAIC fit criteria.

5.1.1 Dataset Description

For our real-data analysis, we decided to analyse the Tecator dataset, available in the fda.usc package in R [1]. This dataset contains information on 215 chopped meat samples. The dataset gives each sample's fat, protein, and moisture percentages; these values are all scalars. The functional portion of the dataset is the result of a spectrometric analysis of the samples, which gives the absorbance of each wavelength of light of every meat sample. The wavelengths of light used in the study vary from 850 to 1050 nm, evenly spaced over 100 grid spaces [1]. We denote each curve as T_i. We can therefore represent the curves as $T_i = (T_i(t_1), \ldots, T_i(t_{100}))$.

We aim to use this hybrid dataset to validate the performance of our models. Applying our models to a dataset unseen by Shan et al. (2020). This semi-functional linear regression aims to estimate the fat percentage of a specific meat sample, given the spectrometer curve and protein and moisture percentages.

A graph of the functional data can be seen in Figure 5.1. This dataset has previously been analysed by a number of researchers to test robust estimation techniques, such as Ferraty and Vieu (2006) and Aneiros-Pérez and Vieu (2006) [10, 1].

We can see a visualisation of the distribution of the residuals of the data through the histogram and qq-plot in Figure 5.2. As shown, the data's residuals are not normally distributed, adding credence to the use of our hierarchical models to better handle the data.

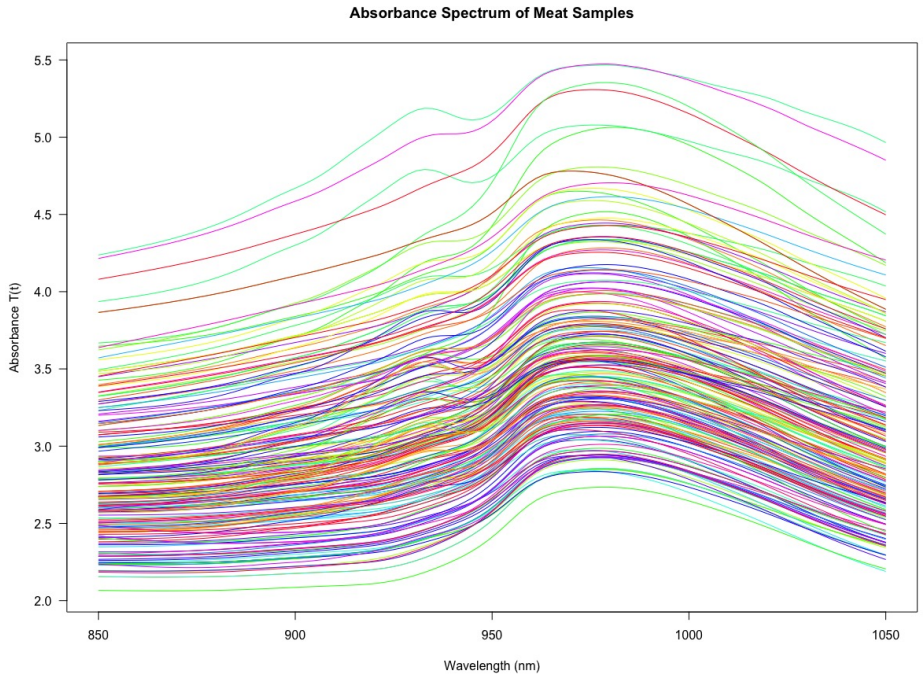
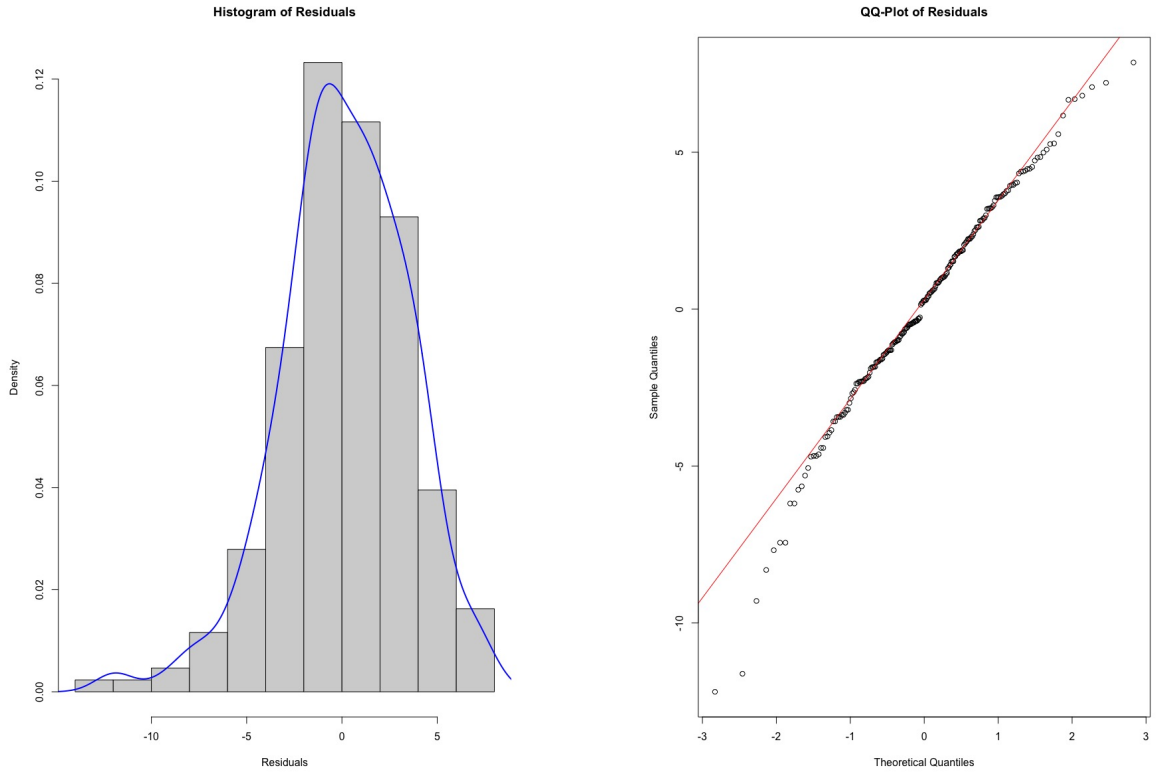


Figure 5.1: Absorbance spectrum of meat samples across wavelengths 850-1050 nm



(a) Histogram of residuals

(b) QQ-plot of residuals

Figure 5.2: Histogram and qq-plot of residuals using Normal model

5.1.2 Model Specification

To predict the fat content of the meat samples using the spectrometric curves and the variables related to protein and moisture contents, we constructed the following model:

$$Y_i = \alpha_1 Z_{i1} + \alpha_2 Z_{i2} + \int_{850}^{1050} \beta(t) T_i(t) dt + \epsilon_i, \tag{5.1}$$

where Y_i is the fat percentage of the i^{th} meat sample, α_1 and α_2 represent the scalar coefficients for the protein and moisture percentages, respectively. Z_{i1} and Z_{i2} represent the protein and moisture percentages of the i^{th} meat sample, respectively. $\beta(t)$ is the functional coefficient associated with the spectrometric curves. $T_i(t)$ is the spectrometric curve for the i^{th} meat sample at wavelength t. ϵ_i is the error term for the i^{th} meat sample.

Once again, we conduct FPCA to lower the dimensionality of this data. We utilise a cumulative variance plot to decide upon the number of FPCs to use. In this case, we chose to use enough FPCs to explain over 99.99% of the variation in the data, resulting in the use of 9 FPCs. Once again, we follow the same approach to smoothing our data as we did in the simulation studies.

5.1.3 Results

	DIC	EAIC	EBIC	LPML	WAIC1
Normal	1075.98	1088.75	1129.19	-541.08	1064.75
Student-t	**1037.12**	**1051.60**	**1095.42**	**-526.10**	**1025.12**
CN	1075.00	1091.79	1138.98	-546.11	1063.79
Slash	1108.52	1120.32	1164.13	-554.37	1094.32

Table 5.1: Selection criteria results for each model on the Tecator dataset

The results from our analysis, shown in Table 5.1, clearly illustrate the effectiveness of using heavy-tailed distributions as a method to account for outliers in the data. As we can see, the Student-t model performed the best according to all the model selection criteria we used. This finding is consistent with both simulations, which also showed strong performance of the Student-t model. These tests improve upon the robustness of Shan et al. (2020) by implementing the proposed and refined models upon a new real-world dataset and increasing the range of model selection criteria.

A few interesting things emerge after looking at the results in Table 5.2. First of all, we witness that for β_5 to β_9, the Student-t model appears to have the lowest standard deviation in samples. Also, Tau varies across all models, with the largest being in the Student-t. All of the models have two parameters whose HPD interval includes zero. The importance of these parameters could be investigated further.

Using the values generated in Table 5.2 under the Student-t model, we plotted the graph of the estimated functional component $\beta(t)$, shown in Figure 5.3, where the solid line represents the mean estimate, and the shaded area represents the upper and lower credible intervals.

The results from Table 5.1 provide proof of the model's ability to handle data outliers. The simulations were run 100 times, and the results were averaged. As with the other simulations, we generated trace plots of the MCMC samples for select parameters to visualise convergence. In this case, we've shown the trace plots for the Student-t model on the real data, as shown in Figure 5.4.

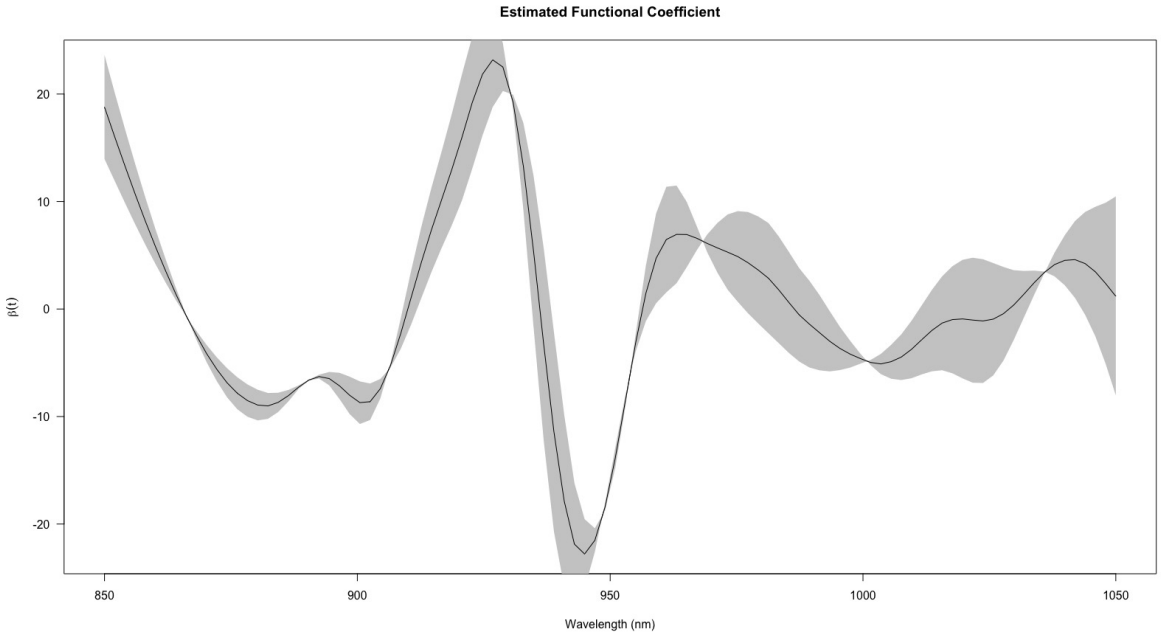
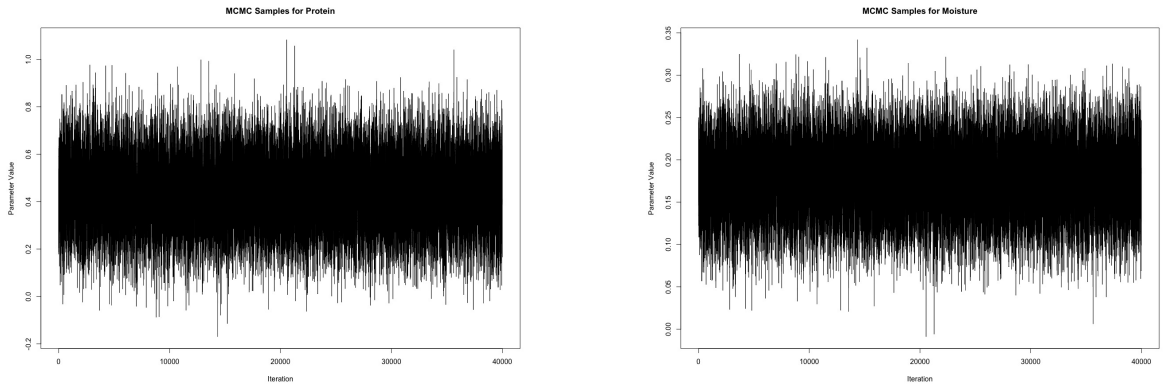
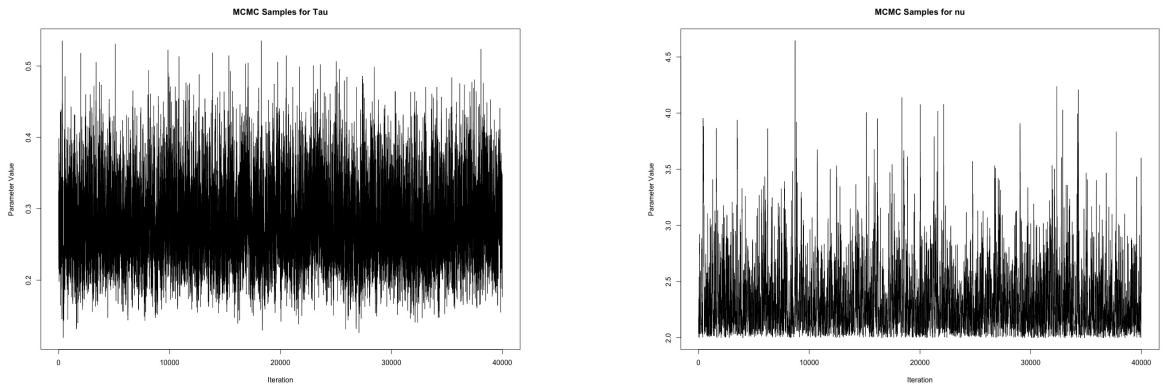


Figure 5.3: Plot of estimated functional component $\beta(t)$ for Tecator dataset, using the Student-t Model



(a) Trace plot for Protein

(b) Trace plot for Moisture



(c) Trace plot for Tau

(d) Trace plot for Nu

Figure 5.4: Trace plots of select parameters for Tecator Dataset: using the Student-t model

Model	Parameter	Mean	SD	HPD Lower	HPD Upper
Normal	β_1	0.74	0.03	0.68	0.80
Normal	β_2	-3.11	0.34	-3.76	-2.45
Normal	β_3	-16.80	0.92	-18.59	-15.00
Normal	β_4	-26.36	1.17	-28.65	-24.07
Normal	β_5	58.08	3.24	51.73	64.41
Normal	β_6	65.33	5.54	54.46	76.19
Normal	β_7	15.64	9.55	-3.09	34.33
Normal	β_8	41.55	12.31	17.48	65.70
Normal	β_9	-36.95	22.94	-81.62	8.19
Normal	α_1	0.60	0.16	0.28	0.92
Normal	α_2	-0.26	0.06	-0.38	-0.15
Normal	τ	0.12	0.01	0.10	0.15
Student-t	β_1	0.92	0.04	0.85	1.00
Student-t	β_2	-3.43	0.34	-4.11	-2.77
Student-t	β_3	-21.02	0.90	-22.78	-19.24
Student-t	β_4	-32.62	1.20	-34.94	-30.24
Student-t	β_5	60.09	2.63	54.95	65.23
Student-t	β_6	50.18	4.70	41.19	59.53
Student-t	β_7	-5.84	8.69	-22.92	11.04
Student-t	β_8	84.63	10.68	63.55	105.44
Student-t	β_9	-40.96	18.49	-77.14	-4.59
Student-t	α_1	0.22	0.13	-0.03	0.47
Student-t	α_2	-0.08	0.05	-0.17	0.01
Student-t	τ	0.42	0.09	0.27	0.60
CN	β_1	0.75	0.03	0.68	0.82
CN	β_2	-3.06	0.34	-3.73	-2.40
CN	β_3	-16.93	0.93	-18.76	-15.10
CN	β_4	-26.46	1.17	-28.74	-24.17
CN	β_5	57.97	3.21	51.71	64.28
CN	β_6	64.83	5.52	54.02	75.66
CN	β_7	14.62	9.62	-4.23	33.43
CN	β_8	43.88	12.79	18.82	68.88
CN	β_9	-38.21	22.88	-82.91	6.62
CN	α_1	0.57	0.17	0.25	0.90
CN	α_2	-0.25	0.06	-0.37	-0.14
CN	τ	0.13	0.01	0.10	0.15
Slash	β_1	0.73	0.04	0.65	0.82
Slash	β_2	-3.13	0.40	-3.92	-2.34
Slash	β_3	-16.59	1.23	-19.01	-14.19
Slash	β_4	-26.21	1.48	-29.11	-23.32
Slash	β_5	58.25	4.02	50.38	66.14
Slash	β_6	64.53	6.93	51.01	78.17
Slash	β_7	16.56	12.21	-7.29	40.52
Slash	β_8	42.08	17.04	8.97	75.63
Slash	β_9	-35.91	25.61	-85.99	14.29
Slash	α_1	0.60	0.20	0.21	0.98
Slash	α_2	-0.26	0.07	-0.40	-0.13
Slash	τ	0.28	0.04	0.21	0.37

Table 5.2: Parameter posterior means, standard deviations, and HPD intervals

Chapter 6

Discussion and Concluding Remarks

Throughout this paper, we explored and expanded upon the field of Functional Data Analysis (FDA). As stated in the Introduction, our primary focus was on robust estimation techniques for semi-functional linear regression models. Through our exploration of FDA, it became clear that the field is ever-growing, and even though many authors have expanded upon works related to the field, there is still much work to be done. Though the techniques made by Ramsay and Dalzell provided a solid foundation, they remain susceptible to outliers. As explained in Section 2.1, several papers have been written to tackle the issue of robust estimation. However, the work of Shan et al. (2020) remains as one of the few papers that aim to apply a Bayesian approach to the problem. As such, it further highlighted the need to expand and evaluate the methodologies proposed in this paper, as they had yet to be built on.

In order to meaningfully contribute to the work, we needed to analyse the methodologies proposed by Shan et al. (2020) thoroughly. Through this analysis, we saw the first cracks in the proposals outlined in their work. One glaring error presented itself in the appendix of the proposal put forward by Shan et al. (2020). As mentioned in Section 3.4 for the prior and posterior used in the Slash case, they truncate the values on $(1, \infty)$. Implementing their method directly would result in an infinite loop as this interval was outside where the data was supported. This clearly illustrates the value of reviewing scientific papers. As a result, we have managed to meaningfully contribute to the field by addressing this error. This could have gone unnoticed, as it remains unaddressed in their paper and WinBugs code.

Furthermore, we found their explanation of the Contaminated Normal model to be lacking. As such, we expanded upon this methodology, given in Section 3.5. In order to improve the correctness of the Slash model and the clarity of their explanations for the CN model, we invested substantial time into researching the pitfalls of both cases in the work of Shan et al. (2020). We aimed to remedy them as precisely as possible. To accomplish this, we drew upon the work of Garay et al. (2015) and combined their approach with the work of Shan et al. (2020). This allowed us to effectively apply the proposed methodologies in R after the refinements.

After coding our models, we tested them on a variety of datasets. The results of our simulation studies, given in Chapter 4, were largely promising. Here, we expanded upon the work of Shan et al. (2020) once again by generating varied datasets. The functional component of our first simulation was generated through a unique Fourier Basis expansion, and in the second simulation, it was generated through a Polynomial Basis expansion. The results of these simulations proved that the models were robust to outliers, with the Student-t case performing highly on all tests. This finding adds credence to the work of Shan et al. (2020). An interesting trend emerged where the value of the LPML criteria, given normally distributed errors, was the lowest for the CN model and not the Normal model for both simulations. This finding was interesting and continued through increases in sample size. As explained in Section 3.5, this can be thought of as a measure of out-of-sample prediction error, telling us how well the model generalises to the unseen data. Although this finding was surprising, the absolute difference between the values was very small and, therefore, not particularly significant. Also, the insignificance of this fact is increased as the CN is a derivation of the Normal, being the closest to it among the models made. This means if the mixing of the CN is low, then the results would essentially match that produced by the Normal.

We tested our models on a real dataset in Chapter 5. This again showed the out-performance of

the Student-t model, adding more backing to the validity of these models. The benefit being the use of the refined work of Shan et al. (2020) in a new context.

Although these findings were good, they did not come without cost. As one would imagine, the compute power required to run these simulations was vast and posed a range of problems. As expected, the implementation in WinBugs was more efficient than our code in R. However, where the code lacks in efficiency, it more than makes up for in versatility. Our code is both intuitive and highly customisable, and as such, any future research that aims to expand upon this work can easily alter this existing code base.

On the topic of future research, through our research into Bayesian techniques, SMN distributions, and FDA, an interesting topic presented itself. We fell upon the topic of Scale-Mixture-of-Skew-Normal distributions. Just as the models generated through the SMN distributions allow for heavy tails in data, these distributions also allow for the incorporation of skewness. As such, they allow us to model data with asymmetries better, which can be applicable in a wide range of fields, for example, finance. Therefore, we believe that future research can be done in this area, expanding upon our versatile code base and methodology.

Bibliography

[1] Germán Aneiros-Pérez and Philippe Vieu. Semi-functional partial linear regression. *Statistics & Probability Letters*, 76(11), Jun 2006.

[2] Adelchi Azzalini and Antonella Capitanio. *The Skew-Normal and Related Families*. Cambridge University Press, Dec 2013.

[3] Philippe Besse and James O. Ramsay. Principal components analysis of sampled functions. *Psychometrika*, 51(2), Jun 1986.

[4] Graciela Boente, Matias Salibian-Barrera, and Pablo Vena. Robust estimation for semi-functional linear regression models. *Computational Statistics & Data Analysis*, 152, Jun 2020.

[5] Graciela Boente and Alejandra Vahnovan. Robust estimators in semi-functional partial linear regression models. *Journal of Multivariate Analysis*, 154, Feb 2017.

[6] T. Tony Cai and Peter Hall. Prediction in functional linear regression. *The Annals of Statistics*, 34(5), Oct 2006.

[7] Xiong Cai, Liugen Xue, and Fei Lu. Robust estimation with a modified huber's loss for partial functional linear models based on splines. *Journal of the Korean Statistical Society*, 49(4), Dec 2020.

[8] Ming-Hui Chen, Qi-Man Shao, and Joseph G. Ibrahim. Monte carlo methods in bayesian computation. *Springer Series in Statistics*, 2000.

[9] Siddhartha Chib. Chapter 57 - markov chain monte carlo methods: Computation and inference. In James J. Heckman and Edward Leamer, editors, *Handbook of Econometrics*, volume 5 of *Handbook of Econometrics*, pages 3569–3649. Elsevier, 2001.

[10] Frédéric Ferraty and Philippe Vieu. *Nonparametric functional data analysis*. Springer, 2006.

[11] Clécio da Silva Ferreira, Heleno Bolfarine, and Víctor H. Lachos. Skew scale mixtures of normal distributions: Properties and estimation. *Statistical Methodology*, 8(2):154–171, Aug 2011.

[12] Aldo M. Garay, Heleno Bolfarine, Victor H. Lachos, and Celso R.B. Cabral. Bayesian analysis of censored linear regression models with scale mixtures of normal distributions. *Journal of Applied Statistics*, 42(12):2694–2714, 2015.

[13] Aldo M. Garay, Heleno Bolfarine, Victor H. Lachos, and Celso Rômulo Barbosa Cabral. Bayesian analysis of censored linear regression models with scale mixtures of normal distributions. *Journal of Applied Statistics*, 42(12), Aug 2015.

[14] Andrew Gelman, John B. Carlin, and Hal S. Stern. *Bayesian Data Analysis Ed. 2*. Taylor & Francis, Jan 2003.

[15] Andrew Gelman, Jessica Hwang, and Aki Vehtari. Understanding predictive information criteria for bayesian models, 2013.

[16] Lele Huang, Huiwen Wang, Hengjian Cui, and Siyang Wang. Sieve m-estimator for a semi-functional linear model. *Science China Mathematics*, 58(11), Jul 2015.

[17] Ben Lambert. *A Student's Guide to Bayesian Statistics*. SAGE Publications Ltd, Aug 2018.

[18] Mehrdad Naderi, Elham Mirfarah, Matthew Bernhardt, and Ding-Geng Chen. Semiparametric inference for the scale-mixture of normal partial linear regression model with censored data. *Journal of Applied Statistics*, 49(12):3022–3043, Aug 2022.

[19] Zhaohui S. Qin, Paul Damien, and Stephen Walker. Scale Mixture Models with Applications to Bayesian Inference. *AIP Conference Proceedings*, 690(1):394–395, 11 2003.

[20] James O. Ramsay and Bernard W. Silverman. *Functional Data Analysis*. Springer Science & Business Media, Apr 2013.

[21] Peter E. Rossi. *Mixtures of Normals*, pages 1–58. Princeton University Press, 2014.

[22] Nicolai Schipper and Jespersen. An introduction to markov chain monte carlo. *SSRN Electronic Journal*, Mar 2010.

[23] Guodong Shan, Yiheng Hou, and Baisen Liu. Bayesian robust estimation of partially functional linear regression models using heavy-tailed distributions. *Computational Statistics*, 35(4):2077–2092, Dec 2020.

[24] Shahid Ullah and Caroline F. Finch. Applications of functional data analysis: A systematic review. *BMC Medical Research Methodology*, 13(1), Mar 2013.

[25] Jane-Ling Wang, Jeng-Min Chiou, and Hans-Georg Müller. Functional data analysis. *Annual Review of Statistics and Its Application*, 3(1), Jun 2016.

[26] Jianjun Zhou, Zhao Chen, and Qingyan Peng. Polynomial spline estimation for partial functional linear regression models. *Computational Statistics*, 31(3), Sep 2016.

[27] Jianjun Zhou, Jiang Du, and Zhimeng Sun. M-estimation for partially functional linear regression model based on splines. *Communications in Statistics - Theory and Methods*, 45(21), Aug 2016.