Generalised Additive Model (GAM)

Fuad Jamari

Supervised by Matteo Fasiolo
Level UG
20 Credit Points

April 29, 2024

Abstract

This essay concerns about the fundamentals of generalised additive model and its application in modelling real life data. In relation to the discussion on generalised additive models, this essay also presented the notions behind quantile estimation using regression methodology. Moving from the basic theories in both subjects the essay further develops connections between generalised additive model using cubic regression spline function in deriving complex interactions between data to produce robust conditional quantile estimation models. The main areas of Mathematics study discussed in this essay are probability and statistics. For instance, this essay explores the probability theories behind linear modelling such as the application of maximum likelihood estimation in estimating the predictor coefficients of regression model and statistical test to analyse the performance of the quantile estimation model. In the latter part of the essay, there will be simulations of quantile regression model onto real life data regarding the air quality in Bristol city in the year 2021. The data are collected from open source data platform *Kaggle* prepared as `Bristol, UK Air Quality Continuous Data` that consists of records for air quality archived by the UK Air Information Resource (AIR) (Iyer, 2022). The details of this data is provided in the references list as [Iye22]. In particular, the objective of the simulations is to model a reliable extreme quantile regression that estimate the 0.95-th quantile of ozone concentration in relation to the temperature of the air. In regards to providing hand to hand working of this modelling procedures, along with this essay are source codes that work on R programming language to guide readers to follow through each practical work and simulation. In order to get better experience in engaging with the practical simulations, readers can access the public repository `group_project_gam` created under GitHub account `foo-art` to download the copy of the original source code entitled `project_script.R` [foo24].

Keywords : Generalised Additive Model; Quantile Regression; Cubic Regression Spline; Ozone; Temperature.

Declaration

I hereby declare that this Group Project essay entitled Generalised Additive Model (GAM) is of my own original work under the supervision of Matteo Fasiolo. This work is completed for the completion requirement of unit MATH30009 under the School of Mathematics in the University of Bristol. Any form of external notions or information gathered in the writing process of this essay are duly credited and documented in the text and in the reference section presented at the end of the essay.

Contents

1 Introduction **4**

2 Generalised Additive Model **6**

2.1 Probabilistic approach in linear modelling 6

2.1.1 Derivation of conditional estimator for response variable 7

2.1.2 Parametric assumption on response variable in linear modelling . . . 8

2.1.3 Estimating predictor coefficients using optimal distribution parameter 10

2.1.4 Maximum likelihood estimation of predictor coefficients for normally distributed response 10

2.1.5 Relationship between linear model for normally distributed response with ordinary least square model 13

2.2 Generalisation of linear models 14

2.2.1 Fundamentals and components of generalised additive model 15

2.2.2 Estimator of conditional response using generalised additive model . 16

2.3 Generalised additive model on univariate observation 17

2.3.1 Modelling simulation using cubic polynomial basis function 18

2.3.2 Application of cubic regression spline function in generalised additive model 20

2.3.3 Bias-variance decomposition in linear modelling 21

2.3.4 Addition of penalty component in generalised additive model 22

3 Quantile Regression **24**

3.1 Derivation of quantile in terms of conditional probability 24
3.2 Parametric approach to estimate conditional quantile 26
3.2.1 Simulation of parametric approach for conditional quantile estimation 27
3.3 Performance Analysis 29
3.3.1 Hypothesis test to evaluate the accuracy of conditional quantile estimate 30
3.4 Non-parametric approach to estimate conditional quantile 32
3.4.1 Introduction to pinball loss function and its role in quantile regression 33
3.4.2 Assessing non-parametric quantile regression model estimation using hypothesis test 36
3.5 Model evaluation using train-test split data 40
3.6 Semi-parametric approach to estimate conditional quantile 43
3.6.1 Construction of smooth form of pinball loss function 44
3.6.2 Weak parametric assumption in determining model weight parameter 45
3.6.3 Construction of semi-parametric quantile regression model 46
3.6.4 Implementation of K-fold cross validation in determining tuning hyperparameter for semi-parametric quantile regression model 48
3.6.5 Assessing semi-parametric quantile regression model estimation using hypothesis test 51
3.6.6 Train-test split to assess the performance of semi-parametric model . 52
3.7 Summary 55

4 Conclusion **56**
4.1 Reflection 56

A R Source Code **59**

B Semi-Parametric Quantile Regression Examples **80**

Chapter 1

Introduction

A generalised additive model (GAM) is an extended form of generalised linear model where the former concerns about finding linear dependence between a response variable, $Y \in \mathbb{R}$ and unknown smooth functions consists of input values from studied covariates, $\mathbf{X} \in \mathbb{R}^p$ (Hastie, and Tibshirani, 1986 [HT86]). Liberation in the choice of smooth functions allows for the study of complex non-linear dependence between responses and covariates. In this project, we are interested to explore the application of generalised additive model to draw estimation for the τ-th quantile of response variable Y denoted by $\mu_\tau \in \mathbb{R}$ conditioned by the respective covariate $\mathbf{X}$. In doing so we will look into different approaches to engage with this problem. In Chapter 3 we will discuss on the relevance of parametric assumption towards quantile estimation and how it influences the performance of the model. The three main approaches in this project are categorised based on their reliance towards parametric assumption on our observations. In particular, we will delve into studying three main approaches in quantile estimation which is as follow :

- **Parametric approach -** The highest degree of probability distribution assumption where we used generalised additive model to estimate the parameters of the assumed probability distribution of our observations. In which from the information on the estimate parameter, we will attempt on retrieving the reading for the τ-th quantile conditioned on the input covariates.
- **Non-parametric approach -** This method requires the least probability distribution assumption on our observations in order to model the τ-th conditional quantile estimation. In order to implement this approach, we will study the role of asymmetric pinball loss function and how the minimisation of the loss function will direct us to estimate the τ-th quantile of observations.
- **Semi-parametric approach -** Developed from our understanding on the non-parametric approach, we introduced the generalised version of pinball loss function that consists of additional weight parameters. According to Cheng and Sun (2006) these weight parameters can be expressed as deterministic values in terms of the estimated probability density function that requires minimal parametric assumptions in doing so [CS06].

In the latter half of Chapter 3, we will study the framework and modelling procedures of each methodologies aside from discussing on their potential strengths and weaknesses. In

extending the capability of these methodologies, this project is motivated by the assessment of air quality in the city of Bristol. Taking into specific context of the problem, we want to learn the effect of temperature towards the concentration of ozone in the proximity of Bristol city. The significance of this real life context is due to the threatening affects that can results from excessive ozone pollution. Besides its famous contribution as the primary ultraviolet protector, relatively high concentration of ozone has bad influences towards human health. These include cardiovascular diseases as well as breathing difficulties that can further causes lung inflammation due to the reactive properties of oxidants component in ozone that can induce reactions with moist lining of organisms' breathing system (Martins *et al.*, 2017 [Mar+17]). In addition to this, as a byproduct from photolytic decomposition of nitrogen oxides, ozone formation is heavily influenced by exposure towards sunlight and heat (Zhang, Wei and Fang, 2019 [ZWF19]). This further discussed in previous finding from [ZWF19] where ozone is at a relatively higher concentration during summer whilst being the dominant pollutant with respect to other compounds. In accordance to this correlation, we highlight the significance of projecting the extreme reading for the concentration of ozone with respect to temperature for better assessment towards the risk of suffering the former consequences. Hence, this project will be focusing on the implementation of modelling procedure using generalised additive model to estimate the extreme quantile of ozone concentration based on recorded temperature. In this essay, we will start off by introducing the theories and characteristics of generalised additive model in Chapter 2 before exploring the said approaches of quantile regression model in Chapter 3.

Chapter 2

Generalised Additive Model

In this Chapter we will study the theories of generalised additive model and simulate the implementation of it using Maximum Likelihood Estimation to comprehend the methodology of fitting generalised additive model onto real observations from Bristol Air Quality data. Since generalised additive model is the generalisation of linear model, it is simpler to comprehend the mechanics of the modelling rules based on simplified version of linear regression before exploring more on generalised additive model.

2.1 Probabilistic approach in linear modelling

Generalised additive model are developed from the basic notion of probability. This idea attempts to express a random variable in terms of other sets of random variables or precisely as the linear transformation of covariates. This construction of model starts off from the basic form of simple linear regression.

Definition 2.1.1 (Linear Regression (pp. 2, [Woo17]))**.** Let $n \in \mathbb{Z}^+$ number of observations for response variable $Y \in \mathbb{R}$ and explanatory covariates $\mathbf{X} \in \mathbb{R}^p$ respectively. Then for all $i \in \{1, ..., n\}$ and $\epsilon_i \in \mathbb{R}$, there exist vector $\boldsymbol{\beta} \in \mathbb{R}^p$ such that

$$y_i = \sum_{j=1}^{p} \beta_j x_{ij} + \epsilon_i, \tag{2.1}$$

where ϵ_i is an independent random error whereas $\boldsymbol{\beta}$ are called predictor coefficients.

Based on Definition 2.1.1, the response variable Y is expressed as a linear transformation of the covariates $\mathbf{X}$ and some random error ϵ. In the context of probabilistic model, randomised effect like random error ϵ in our model represents uncertainty towards the model for the response, Y in terms of $\mathbf{X}$. Hence, in an effort to control the random behaviour towards our model, we need to fix some conditions on the random error ϵ.

Definition 2.1.2 (Random Error)**.** Consider for all $i \in \{1, ..., n\}$, we have ϵ_i to be random error in linear regression model as in Definition 2.1.1. We let $\epsilon_i \overset{\text{iid}}{\sim} \mathrm{N}(0, \sigma^2)$ such that $\sigma^2 \geq 0$. Thus, for all $i \in \{1, ..., n\}$ and $j \in \{1, ..., n\}$ we have

(i) $\mathbb{E}(\epsilon_i) = 0$,

(ii) $\text{Var}(\epsilon) = \sigma^2$ and

(iii) $\text{Cov}(\epsilon_i, \epsilon_j) = 0$ for all $i \neq j$.

Taking the knowledge from Definition 2.1.1 and Definition 2.1.2, we can now model the predicted values for response variable, y denoted by μ for a given observation $\mathbf{x} \in \mathbf{X}$. In which in this case, we require μ to only be depended on $\mathbf{x}$.

2.1.1 Derivation of conditional estimator for response variable

Lemma 2.1.1. *Let $\mu \in \mathbb{R}$ be the estimation of response $Y \in \mathbb{R}$ that depends on covariate $\mathbf{X} \in \mathbb{R}^p$. Then for some $\boldsymbol{\beta} \in \mathbb{R}^p$ we can defined μ as*

$$\mu(\mathbf{X}) = \mathbb{E}(Y|\mathbf{X}) = \mathbf{X}^\intercal \boldsymbol{\beta}, \tag{2.2}$$

which is the expectation for response Y conditioned on $\mathbf{X}$. Therefore, μ is a random variable consists of a linear transformation of covariate $\mathbf{X}$.

Proof. Firstly, we need to proof that $\mu \in \mathbb{R}$ is a random variable. Considering the expectation for random variable of response denoted by $\mathbb{E}(Y)$.

$$\mathbb{E}(Y) = \sum_{\omega \in \Omega} Y(\omega) p(\omega), \tag{2.3}$$

then we define $k > 0$ number of partitions on observations of covariates, $\{\mathbf{x}_1, ...\mathbf{x}_k\} \in \mathbf{X}$. On the other hand, we let partitions of $A = \{A_1, ..., A_k\}$ where $A_i = \{\omega \in \Omega : \mathbf{X}(\omega) = \mathbf{x}_i\}$. Then, substituting this expression into (2.3) and by applying Law of Total Probability, we have

$$\begin{aligned}
\mathbb{E}(Y) &= \sum_{\omega \in \Omega} Y(\omega) \left[\sum_{i=1}^{k} p(w|\mathbf{x}_i) \mathbb{P}(\mathbf{X} = \mathbf{x}_i) \right] \\
&= \sum_{i=1}^{k} \left[\sum_{\omega \in \Omega} Y(\omega) p(w|\mathbf{x}_i) \right] \mathbb{P}(\mathbf{X} = \mathbf{x}_i) \\
&= \sum_{i=1}^{k} \mathbb{E}(Y|\mathbf{X} = \mathbf{x}_i) \mathbb{P}(\mathbf{X} = \mathbf{x}_i),
\end{aligned} \tag{2.4}$$

which by the Law of Iterated Expectation, we can deduce the expression on the right-hand side as

$$\begin{aligned}
\mathbb{E}(Y) &= \sum_{i=1}^{k} \mathbb{E}(Y|\mathbf{X} = \mathbf{x}_i) \mathbb{P}(\mathbf{X} = \mathbf{x}_i) \\
&= \mathbb{E}[\mathbb{E}(Y|\mathbf{X})] \\
&= \mathbb{E}[\mu(\mathbf{X})].
\end{aligned} \tag{2.5}$$

Since the function expectation only takes input of random variable, therefore $\mu = \mathbb{E}(Y|\mathbf{X})$ is a random variable. Next, we need to show that μ is a linear transformation of covariate $\mathbf{X}$. This can be done by simply substituting Y in (2.5) with the expanded form of Y in Definition 2.1.1,

$$\mu(\mathbf{X}) = \mathbb{E}(Y|\mathbf{X}) = \mathbb{E}\left[\left(\sum_{j=1}^{p}\beta_j x_j + \epsilon\right)\middle|\mathbf{X}\right], \tag{2.6}$$

using invariant property of expectation and independent property of random error ϵ we obtained

$$\begin{aligned}\mu(\mathbf{X}) = \mathbb{E}(Y|\mathbf{X}) &= \mathbb{E}\left[\left(\sum_{j=1}^{p}\beta_j X_j + \epsilon\right)\middle|\mathbf{X}\right] \\ &= \left(\sum_{j=1}^{p}\mathbb{E}(\beta_j X_j|\mathbf{X})\right) + \mathbb{E}(\epsilon) \\ &= \sum_{j=1}^{p}\beta_j X_j,\end{aligned} \tag{2.7}$$

since we know from Definition 2.1.2 that $\mathbb{E}(\epsilon) = 0$. Therefore, $\mu = \mathbb{E}(Y|\mathbf{X})$ is a random variable defined as linear transformation of covariate $\mathbf{X} = (X_1, ..., X_p)^\intercal$ for a given $\boldsymbol{\beta} \in \mathbb{R}^p$. □

2.1.2 Parametric assumption on response variable in linear modelling

Referring to Lemma 2.1.1, we know that it is possible to construct the estimation of response $\mu(\mathbf{x})$ given a sets of observations of covariates $\mathbf{x} \in \mathbf{X}$ if we can determine the values of the predictor coefficients $\boldsymbol{\beta}$. The question now is how can we do so? One of the answer to this is by approximating the distribution for the response variable Y. This can be done by setting a few rules for the distribution of the response variable Y.

Definition 2.1.3 (Exponential Family (p. 1, [GS]))**.** The distribution of random variable Y is said to arise from exponential family if the probability density function of Y denoted as $f(\cdot)$ can be written as

$$f(y;\boldsymbol{\theta}) = g(y)\cdot\exp\{\boldsymbol{\theta}^\intercal t(y) - h(\boldsymbol{\theta})\}, \tag{2.8}$$

for some canonical parameters $\boldsymbol{\theta} \in \mathbb{R}^m$ and functions $g : \mathbb{R} \to \mathbb{R}$ and $h : \mathbb{R}^m \to \mathbb{R}$. Whereas $t : \mathbb{R} \to \mathbb{R}^m$ be defined as sufficient statistics.

Example 1. Normal distribution belongs to the exponential family. Consider random variable $Y \sim N(\mu, \sigma^2)$, we have the probability density function $f(\cdot)$ defined as

$$f(y;\mu,\sigma^2) = \frac{1}{\sqrt{2\pi\sigma^2}}\exp\left\{-\frac{(y-\mu)^2}{2\sigma^2}\right\}, \tag{2.9}$$

where by substituting $1/\sqrt{\sigma^2} = \exp\{-\log(\sigma)\}$ in (2.9), we have the following

$$f(y;\mu,\sigma^2) = \frac{1}{\sqrt{2\pi}}\exp\left\{-\frac{(y-\mu)^2}{2\sigma^2} - \log(\sigma)\right\}, \tag{2.10}$$

then by expanding the inner square term in (2.10) we obtain

$$\begin{aligned} f(y;\mu,\sigma^2) &= \frac{1}{\sqrt{2\pi}}\exp\left\{-\frac{y^2}{2\sigma^2} + \frac{y\mu}{\sigma^2} - \frac{\mu^2}{2\sigma^2} - \log(\sigma)\right\} \\ &= \frac{1}{\sqrt{2\pi}}\exp\left\{\begin{pmatrix}-1/2\sigma^2 & \mu/\sigma^2\end{pmatrix}\cdot\begin{pmatrix}y^2\\ y\end{pmatrix} - \left(\frac{\mu^2}{2\sigma^2} + \log(\sigma)\right)\right\}. \end{aligned} \tag{2.11}$$

In which by comparing (2.11) to Definition 2.1.3 we have

$$g(y) = \frac{1}{\sqrt{2\pi}},\quad \boldsymbol{\theta} = \begin{pmatrix}-1/2\sigma^2\\ \mu/\sigma^2\end{pmatrix},\quad t(y) = \begin{pmatrix}y^2\\ y\end{pmatrix},\quad h(\boldsymbol{\theta}) = -\frac{\theta_1^2}{4\theta_2^2} - \frac{1}{2}\log(-2\theta_2). \tag{2.12}$$

From Definition 2.1.3, we realise that by making distribution assumptions towards the conditional response variable $Y|\mathbf{X}$ we somehow introduce unknown distribution parameters $\boldsymbol{\theta}$ to be estimated instead of the predictor coefficients $\boldsymbol{\beta}$ as in Lemma 2.1.1. Considering that our distribution is from the exponential family with parameter $\boldsymbol{\theta}$, it is possible to express $\boldsymbol{\theta}$ in terms of $\mu = \mathbb{E}(Y|X)$ using a canonical link function.

Definition 2.1.4 (Link Function (p. 42, [And23]))**.** let response variable Y be of the member of exponential family with probability density function $f(\cdot)$ with parameter $\theta \in \mathbb{R}$. Then, $g(\cdot)$ is said to be a canonical link function if for $\mu = \mathbb{E}(Y|\mathbf{X})$ we have

$$g(\mu) = \theta. \tag{2.13}$$

Example 2. Let $Y|\mathbf{X} \sim \text{Exponential}(\theta)$ for $\theta > 0$. Then we have the expectation of $Y|\mathbf{X}$ to be

$$\mu = \mathbb{E}(Y|\mathbf{X}) = \frac{1}{\theta}. \tag{2.14}$$

In which by comparing the estimation of $Y \in \mathbb{R}$ conditioned by covariates $\mathbf{X} \in \mathbb{R}^p$ from Lemma 2.1.1 with (2.14) and for some $\boldsymbol{\beta} \in \mathbb{R}^p$ we have

$$\mu = \frac{1}{\theta} = \mathbf{X}^\intercal\boldsymbol{\beta}, \tag{2.15}$$

where we can defined a link function $g(z) = 1/z$ such that

$$g(\mu) = \frac{1}{\mathbf{X}^\intercal\boldsymbol{\beta}} = \theta. \tag{2.16}$$

Assuming the distribution of our response variable Y follows Definition 2.1.3, it is possible to find the optimal solution for the unknown predictor coefficients $\boldsymbol{\beta}$ by firstly finding the most suitable distribution parameters $\boldsymbol{\theta}$. Only then we can use the inverse of the link function to obtain the solution for $\boldsymbol{\beta}$.

2.1.3 Estimating predictor coefficients using optimal distribution parameter

Generally, the problem of solving the probability density function parameters $\boldsymbol{\theta}$ can be solved by using the maximum likelihood estimation method (MLE). This method involves computation of unknown parameters $\boldsymbol{\theta}$ based on which maximises the likeliness of obtaining the observations of the response variables $y_i \in Y$ for all $i = \{1, ..., n\}$ (Wood, 2017 [Woo17]).

Definition 2.1.5 (Maximum Likelihood Estimate (MLE) (p. 37, [Yu22]))**.** For all $i = \{1, ..., n\}$, let y_i be independent and identically distributed observations of Y with (assumedly) known probability distribution with probability density function $f(\cdot|\boldsymbol{\theta})$. Where $\boldsymbol{\theta}$ be some unknown parameters. We determine the maximum likelihood estimate for θ denoted by $\hat{\boldsymbol{\theta}}_{mle}$ as

$$\hat{\boldsymbol{\theta}}_{mle} = \arg\max_{\boldsymbol{\theta}} L_n(\boldsymbol{\theta}|\mathbf{y}) = \arg\max_{\boldsymbol{\theta}} \prod_{i=1}^{n} f(y_i|\boldsymbol{\theta}). \tag{2.17}$$

From Definition 2.1.5, we can see how the likelihood function $L_n(\cdot|\mathbf{y})$ represented by the product of the marginal probability density function for all observations $y_i \in Y$. This is due to our assumption for each individual observation to be independent and identically distributed (Yu, 2022 [Yu22]). Extending from the maximum likelihood estimation method for $\hat{\boldsymbol{\theta}}_{mle}$, we can also define the solution for predictor coefficients $\boldsymbol{\beta} \in \mathbb{R}^p$ using the same method considering that our response is defined as in Definition 2.1.1. This is due to the fact that in exponential family case, we have $\theta = g(\mu)$ for some function $g : \mathbb{R} \to \mathbb{R}$. In addition, by using Lemma 2.1.1 we have $\mu = \mathbf{X}\boldsymbol{\beta}$ for some unknown $\boldsymbol{\beta} \in \mathbb{R}^p$. Hence, the Maximum Likelihood Estimate for $\boldsymbol{\beta} = \hat{\boldsymbol{\beta}}_{mle}$ be defined as

$$\hat{\boldsymbol{\beta}}_{mle} = \arg\max_{\boldsymbol{\beta}} \prod_{i=1}^{n} f(y_i|g(\mathbf{x}_i^\intercal\boldsymbol{\beta})), \tag{2.18}$$

considering that the assumed distribution of the response variable Y follows the criteria in Definition 2.1.3. Just to clarify the role of the link function $g(\cdot)$ is to transform the the estimation for our responses, μ into suitable input parameters $\boldsymbol{\theta}$ for our assumed distribution of $Y|\mathbf{X}$. To understand the mechanics of this method of estimation, we will try to simulate this model estimation on an assumed normally distributed response $Y \in \mathbb{R}$ conditioned by one covariate or univariate $X \in \mathbb{R}$.

2.1.4 Maximum likelihood estimation of predictor coefficients for normally distributed response

Let response variable $Y \in \mathbb{R}$ and univariate $X \in \mathbb{R}$. Assuming that $Y|X \sim N(\mu, \sigma^2)$ where using results from Lemma 2.1.1 we have $\mu = X\beta$ for some positive value σ^2. By considering $n > 0$ number of identical and independently distributed observations $y \in Y$ and $x \in X$, we can write the Likelihood function or joint probability density function, $L_n(\cdot)$ for observations y in terms of x,

$$
\begin{aligned}
L_n(\mu, \sigma^2|\mathbf{y}) &= \prod_{i=1}^{n} f(y_i|x_i\beta, \sigma^2) \\
&= \prod_{i=1}^{n} \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left\{-\frac{(y_i - \mathbf{x}_i\beta)^2}{2\sigma^2}\right\},
\end{aligned} \tag{2.19}
$$

where by using Definition 2.1.5, the solution for Maximum Likelihood Estimate for $\beta = \hat{\beta}_{mle}$ be defined as

$$
\hat{\beta}_{mle} = \arg\max_{\boldsymbol{\beta}} \prod_{i=1}^{n} \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left\{-\frac{(y_i - x_i\beta)^2}{2\sigma^2}\right\}. \tag{2.20}
$$

In solving (2.20), we can find the solution of β for which the derivative of the expression in (2.19) is equal to zero which indicates the stationary point of the function (Jackie, 2004 [Jac04]). Since we are dealing with product of exponentials it is feasible to perform log transformation to our function to simplify the problem. In fact log transformation does not influence the optimal solution for (2.20) because logarithm function is a monotonic increasing function which sustain the location of maximum [Yu22]. After using log transformation on (2.19) we have as follows

$$
\begin{aligned}
\ell_n(\mu, \sigma^2|\mathbf{y}) &= \log\left[\prod_{i=1}^{n} \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left\{-\frac{(y_i - x_i\beta)^2}{2\sigma^2}\right\}\right] \\
&= \sum_{i=1}^{n}\left[\log\left(\frac{1}{\sqrt{2\pi\sigma^2}}\right) - \left(\frac{(y_i - x_i\beta)^2}{2\sigma^2}\right)\right] \\
&= -\log\left[(2\pi\sigma^2)^{\frac{n}{2}}\right] - \sum_{i=1}^{n}\left(\frac{(y_i - x_i\beta)^2}{2\sigma^2}\right) \\
&= -\frac{n}{2}\left[\log(\sigma^2) + \log(2\pi)\right] - \frac{\sum_{i=1}^{n}(y_i - x_i\beta)^2}{2\sigma^2}
\end{aligned} \tag{2.21}
$$

In finding the optimal solution to (2.20), find the first derivative to (2.21) in terms of both β and σ^2,

$$
\begin{aligned}
\frac{\delta\ell_n(\mu, \sigma^2|\mathbf{y})}{\delta\beta} &= -\frac{\sum_{i=1}^{n} -2x_i(y_i - x_i\beta)}{2\sigma^2} \\
&= \frac{\sum_{i=1}^{n} x_iy_i - x_i^2\beta}{\sigma^2},
\end{aligned} \tag{2.22}
$$

$$
\begin{aligned}
\frac{\delta\ell_n(\mu, \sigma^2|\mathbf{y})}{\delta\sigma^2} &= -\frac{n}{2\sigma^2} - \frac{\sum_{i=1}^{n}\left[0 \cdot 2\sigma^2 - 2(y_i - x_i\beta)^2\right]}{(2\sigma^2)^2} \\
&= -\frac{n}{2\sigma^2} + \frac{\sum_{i=1}^{n}(y_i - x_i\beta)^2}{2\sigma^4}.
\end{aligned} \tag{2.23}
$$

Then, we will find the stationary point by equating both (2.22) and (2.23) to zero so we obtained

$$\frac{\delta \ell_n(\mu, \sigma^2|\mathbf{y})}{\delta \beta} = \frac{\sum_{i=1}^{n} x_i y_i - x_i^2 \beta}{\sigma^2} = 0, \tag{2.24}$$

$$\frac{\delta \ell_n(\mu, \sigma^2|\mathbf{y})}{\delta \sigma^2} = -\frac{n}{2\sigma^2} + \frac{\sum_{i=1}^{n}(y_i - x_i\beta)^2}{2\sigma^4} = 0. \tag{2.25}$$

To find the closed form solution for $\hat{\beta}_{mle}$, we will solve (2.25) first,

$$\begin{aligned} \frac{\delta \ell_n(\mu, \sigma^2|\mathbf{y})}{\delta \sigma^2} = -\frac{n}{2\sigma^2} + \frac{\sum_{i=1}^{n}(y_i - x_i\beta)^2}{2\sigma^4} &= 0 \\ \frac{n}{2\sigma^2} &= \frac{\sum_{i=1}^{n}(y_i - x_i\beta)^2}{2\sigma^4} \\ \sigma^2 &= \frac{\sum_{i=1}^{n}(y_i - x_i\beta)^2}{n}, \end{aligned} \tag{2.26}$$

then we solve (2.24) by substituting σ^2 from (2.26) into (2.24),

$$\begin{aligned} \frac{\delta \ell_n(\mu, \sigma^2|\mathbf{y})}{\delta \beta} = \frac{\sum_{i=1}^{n} x_i y_i - x_i^2 \beta}{\sigma^2} &= 0 \\ n \cdot \frac{\sum_{i=1}^{n} x_i y_i - x_i^2 \beta}{\sum_{i=1}^{n}(y_i - x_i\beta)^2} &= 0 \\ \frac{\sum_{i=1}^{n} x_i^2 \beta}{\sum_{i=1}^{n}(y_i - x_i\beta)^2} &= \frac{\sum_{i=1}^{n} x_i y_i}{\sum_{i=1}^{n}(y_i - x_i\beta)^2} \\ \hat{\beta}_{mle} = \frac{\sum_{i=1}^{n} x_i y_i}{\sum_{i=1}^{n} x_i^2} &= \sum_{i=1}^{n} \frac{x_i y_i}{x_i^2}. \end{aligned} \tag{2.27}$$

Thus, we have the maximum likelihood estimate $\hat{\beta}_{mle}$ in terms of observations $y_i \in Y$ and $x_i \in X$ where $x_i \neq 0$ for all $i = \{1, ..., n\}$. In addition by using substituting (2.27) into (2.26) we obtain the maximum likelihood estimation for $\hat{\sigma}^2_{mle}$ to be

$$\hat{\sigma}^2_{mle} = \frac{\sum_{i=1}^{n}(y_i - x_i\hat{\beta}_{mle})^2}{n}. \tag{2.28}$$

In addition to the closed form solution for the predictor coefficients by maximum likelihood estimation in (2.27), one may wonder if the solution is applicable to other form of best fit linear regression problem. The answer to such question is basically depend on the expression of the first derivative expression of the model objective function. Likewise, the case of solving linear model for normally distributed response is equivalent to the ordinary least square problem due to the similarities in their first derivative expression. In order to investigate this relationship, we can compare the finalised closed form expression of their respective predictor coefficient, $\boldsymbol{\beta}$.

2.1.5 Relationship between linear model for normally distributed response with ordinary least square model

Ordinary least square problem concerns with finding the best regression estimator as defined in Lemma 2.1.1 by minimising the expected mean square error between the response variable Y with the associated estimate of Y denoted by μ. This problem can be defined as follows,

$$\hat{\mu} = \arg\min_{\mu} \mathbb{E}[||Y - \mu||^2], \tag{2.29}$$

However, due to the lack of information regarding the actual probability distribution of our response in common scenario, ordinary least square can be expressed in its empirical form. Beside its practicality in general application, the empirical version of ordinary least square also do not involve us making of parametric assumptions towards our model. Specifically in the case of simple linear regression model, the empirical ordinary least square model is defined as follows.

Definition 2.1.6 (Empirical Ordinary Least Square model (p. 3, [Woo17]))**.** Let $n > 0$ number of observations for response $Y \in \mathbb{R}$ and covariates $\mathbf{X} \in \mathbb{R}^p$. Considering Definition 2.1.1 for the simple regression model and Definition 2.1.2 to describe the model random error, we can express the regression model for the predictor coefficients $\boldsymbol{\beta} \in \mathbb{R}^p$ of ordinary least square as follows,

$$\hat{\boldsymbol{\beta}}_{ols} = \arg\min_{\boldsymbol{\beta}} \sum_{i=1}^{n} (y_i - \mathbf{x}_i^\intercal \boldsymbol{\beta})^2 \tag{2.30}$$

In linear regression model, we require our fitted model or estimate to be close to the real value of the responses. Hence, the relevance in ordinary least square described in Definition 2.1.6 that minimises the squared distance between the response observations and their respective fitted values. In which aligns with the goal of minimising the error in our estimation. By such, we are interested to know whether the solution to Definition 2.1.6 represent a direct consequence of solving linear regression model through maximum likelihood estimation method of normally distributed response proposed in equation (2.20).

Proposition 2.1.1. Let $n > 0$ sized vector containing observations of response $(y_1, ..., y_n)^\intercal = \mathbf{y} \in \mathbb{R}^n$ and n observations of covariates $(\mathbf{x}_1, ..., \mathbf{x}_n)^\intercal = \mathbf{X} \in \mathbb{R}^{n \times p}$. The solution for the predictor coefficients $\boldsymbol{\beta} \in \mathbb{R}^p$ by ordinary least square method is defined as follows

$$\hat{\boldsymbol{\beta}}_{ols} = (\mathbf{X}^\intercal \mathbf{X})^{-1} \mathbf{X}^\intercal \mathbf{y} = \sum_{i=1}^{n} \frac{\mathbf{x}_i y_i}{\mathbf{x}_i^\intercal \mathbf{x}_i}, \tag{2.31}$$

such that $\mathbf{X}$ be a full rank matrix.

Proof. Let $n > 0$ sized vector containing observations of response $(y_1, ..., y_n)^\intercal = \mathbf{y} \in \mathbb{R}^n$ and n observations of covariates $(\mathbf{x}_1, ..., \mathbf{x}_n)^\intercal = \mathbf{X} \in \mathbb{R}^{n \times p}$ and consider the minimisation problem from Definition 2.1.6. We expand the squared error in matrix form we have

$$\begin{aligned} \sum_{i=1}^{n} ||y_i - \mathbf{x}_i^\intercal \boldsymbol{\beta}||^2 &= (\mathbf{y} - \mathbf{X}\boldsymbol{\beta})^\intercal (\mathbf{y} - \mathbf{X}\boldsymbol{\beta}) \\ &= [\mathbf{y}^\intercal - (\mathbf{X}\boldsymbol{\beta})^\intercal](\mathbf{y} - \mathbf{X}\boldsymbol{\beta}) \\ &= [\mathbf{y}^\intercal - \boldsymbol{\beta}^\intercal \mathbf{X}^\intercal](\mathbf{y} - \mathbf{X}\boldsymbol{\beta}) \\ &= (\mathbf{y}^\intercal \mathbf{y} - \mathbf{y}^\intercal \mathbf{X}\boldsymbol{\beta}) - (\boldsymbol{\beta}^\intercal \mathbf{X}^\intercal \mathbf{y} - \boldsymbol{\beta}^\intercal \mathbf{X}^\intercal \mathbf{X}\boldsymbol{\beta}) \end{aligned} \tag{2.32}$$

In the second line of (2.32), we apply transpose of a sum property while in the third line, we apply transpose of a product property. In the last line of (2.32), we use distributive property of matrices. From this, we differentiate (2.32) in terms of $\boldsymbol{\beta}$.

$$\frac{\delta}{\delta\boldsymbol{\beta}}[(\mathbf{y}^\intercal\mathbf{y} - \mathbf{y}^\intercal\mathbf{X}\boldsymbol{\beta}) - (\boldsymbol{\beta}^\intercal\mathbf{X}^\intercal\mathbf{y} - \boldsymbol{\beta}^\intercal\mathbf{X}^\intercal\mathbf{X}\boldsymbol{\beta})] = -y^\intercal\mathbf{X} - (\mathbf{X}^\intercal\mathbf{y} - 2\mathbf{X}^\intercal\mathbf{X}\boldsymbol{\beta}), \tag{2.33}$$

In order to obtain the minimum of (2.32), we equate (2.33) to zero to compute the stationary point of (2.32). Thus, we have

$$\begin{aligned}
\frac{\delta}{\delta\boldsymbol{\beta}}[(\mathbf{y}^\intercal\mathbf{y} - \mathbf{y}^\intercal\mathbf{X}\boldsymbol{\beta}) - (\boldsymbol{\beta}^\intercal\mathbf{X}^\intercal\mathbf{y} - \boldsymbol{\beta}^\intercal\mathbf{X}^\intercal\mathbf{X}\boldsymbol{\beta})] &= 0 \\
-y^\intercal\mathbf{X} - (\mathbf{X}^\intercal\mathbf{y} - 2\mathbf{X}^\intercal\mathbf{X}\boldsymbol{\beta}) &= 0 \\
-2\mathbf{X}^\intercal\mathbf{y} - 2\mathbf{X}^\intercal\mathbf{X}\boldsymbol{\beta} &= 0 \\
2\mathbf{X}^\intercal\mathbf{X}\boldsymbol{\beta} &= 2\mathbf{X}^\intercal\mathbf{y} \\
\hat{\boldsymbol{\beta}}_{ols} &= (\mathbf{X}^\intercal\mathbf{X})^{-1}\mathbf{X}^\intercal\mathbf{y}.
\end{aligned} \tag{2.34}$$

Considering the element wise form of (2.34), we can expressed $\hat{\boldsymbol{\beta}}_{ols}$ as follows,

$$\hat{\boldsymbol{\beta}}_{ols} = \sum_{i=1}^{n} \frac{\mathbf{x}_i y_i}{\mathbf{x}_i^\intercal\mathbf{x}_i}. \tag{2.35}$$

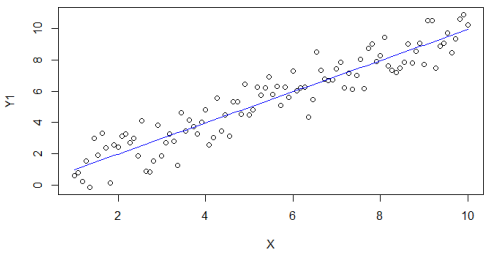
□

Comparing the result for Proposition 2.1.1 to (2.27), we realise the similarities in the closed form expression for the regression estimator $\boldsymbol{\beta}$, i.e. $\hat{\boldsymbol{\beta}}_{ols} = \hat{\boldsymbol{\beta}}_{mle}$. Hence, we know by fact that solving linear regression model by maximum likelihood estimation with normality assumption indeed minimises the error between the response variable Y and their respective linear estimate $\mu(\mathbf{X}) = \mathbf{X}\boldsymbol{\beta}$. Now that we have known the fundamentals of simple linear regression, how do all of this relates to the general case of generalised additive model? In doing so, we will follow through the main criteria of generalised additive model and how the generalisation is adapted from simple linear regression.

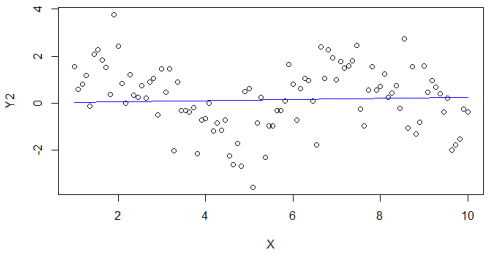
Remarks : Just to be reminded there are further explorations on the properties for the estimation of β and σ^2 in [Woo17]. However in this project we are not concern by those properties in depth since we want to focus mainly on applying the results onto real modelling procedures.

2.2 Generalisation of linear models

Learning from the objective of linear regression in previous section, we know that it is possible to study the dependence between a response variable Y with the respective covariate $\mathbf{X}$ by introducing some predictor coefficients $\boldsymbol{\beta}$. However, we do realise that this approach only feasible for obvious linear interaction between response Y and the associated covariates $\mathbf{X}$. This effects can be seen in the diagram below. Figure 2.1 below shows observations of toy responses $Y1$ and $Y2$ given covariate X. The fitting of linear regression onto the observations are performed using R built-in function `glm` that leverages Maximum Likelihood Estimation method.



(a) Appropriate for estimation of $Y1$.



(b) Inappropriate for estimation of $Y2$.

Figure 2.1: Fitting simple linear regression onto different responses $Y1$ and $Y2$ (blue).

From the above Figures 2.1, we can make visual comparison for the accuracy of the estimation for the respective responses $Y1$ and $Y2$. As we can see from Figures 2.1a, simple linear regression fit suitably to the trend of the observation of response $Y1$. However, it is not the case for the observation of response $Y2$ since its dependence to the observations of covariate x is non-linear. In fact, the mean squared error of the fitted values and the actual observations of the response is relatively higher when fitting simple linear model onto $Y2$ which is $\text{MSE}(\boldsymbol{\beta}; Y2) = 1.392804$ compared to fitting simple linear model onto $Y1$ which is $\text{MSE}(\boldsymbol{\beta}; Y1) = 0.8062881$. Hence, the need for generalised additive model to allow the capturing of complex non-linear interactions between response variable and the associated covariates.

2.2.1 Fundamentals and components of generalised additive model

Definition 2.2.1 (Generalised Additive Model (GAM) (p. 1549, [WPS16]))**.** Let $n \in \mathbb{Z}^+$ number of observations for response variable $Y \in \mathbb{R}$ and explanatory covariates $\mathbf{X} \in \mathbb{R}^p$ respectively. Then for all $i \in \{1, ..., n\}$ and $\epsilon_i \in \mathbb{R}$, there exist $m > 0$ number of unknown basis function $b_j : \mathbb{R}^p \to \mathbb{R}$ for all $j \leq M$ such that

$$y_i = \sum_{j=1}^{m} b_j(\mathbf{x}_i) + \epsilon_i, \tag{2.36}$$

where ϵ_i is an independent random error whereas m is a predetermined value.

In the case of generalised additive model, we leverages the sum of basis functions that takes the input of covariates instead of the covariates themselves. As such we can study various non-linear interactions between the response variable and the covariates depending on the transformation applied on the covariates by the basis function. In particular we consider the basis function to be as follows.

Definition 2.2.2 (Basis Function)**.** $b : \mathbb{R}^p \to \mathbb{R}$ is a linear transformation on r number of fixed linearly independent transformations of $\mathbf{X} \in \mathbb{R}^p$ denoted by $h_k(X)$ for $k \leq r$. Hence for some $\boldsymbol{\beta} \in \mathbb{R}^r$, the basis function can be expanded as

$$b(\mathbf{x}) = \sum_{k=1}^{r} \beta_k h_k(\mathbf{x}). \tag{2.37}$$

Example 3. In order to understand this better, take for example a cubic polynomial function. let $x \in \mathbb{R}$ and $f(\cdot)$ represents the cubic polynomial function of x. Then for $\boldsymbol{\beta} \in \mathbb{R}^3$ we have $b(\cdot)$ be

$$b(x) = \sum_{i=0}^{3} \beta_i x^i = \beta_0 + \beta_1 x + \beta_2 x^2 + \beta_3 x^3, \tag{2.38}$$

and by comparing with definition of basis function in Definition 2.2.1, we also write $b(\cdot)$ as

$$b(x) = \sum_{k=0}^{3} \beta_k h_k(x) = \sum_{i=0}^{3} \beta_i x^i, \tag{2.39}$$

where $h_0(x) = x^0 = 1$, $h_1(x) = x$, $h_2(x) = x^2$ and $h_3(x) = x^3$. In which in this case, we treat each $h_i(x) = x^i$ as a fixed linearly independent transformation of x and $b(\cdot)$ is a basis function that consists of the linear transformation of $h_i(x)$ for all $i = \{0, ..., 3\}$.

Applying this understanding and by considering Definition 2.2.1 of generalised additive model, we are interested to know whether the non-linearity of the basis function $b(\cdot)$ will complicate the method of estimating the response variable Y. And if so, how can we reduce the non-linear problem of finding the estimation of response Y into linear problem.

2.2.2 Estimator of conditional response using generalised additive model

Lemma 2.2.1. *Let $\mu \in \mathbb{R}$ be estimation of response $Y \in \mathbb{R}$ conditioned on covariate $\mathbf{X} \in \mathbb{R}^p$. Then given $r \times m$ number of fixed linearly independent transformation of $\mathbf{X}$ we can defined μ as*

$$\mu(\mathbf{X}|\boldsymbol{\beta}) = \mathbb{E}(Y|\mathbf{X}) = \sum_{j=1}^{m} \sum_{k=1}^{r} \beta_{jk} h_{jk}(\mathbf{X}). \tag{2.40}$$

Proof. Using equation from definition 2.2.2 we have the response variable Y be defined as

$$\begin{aligned} Y &= \sum_{j=1}^{m} b_j(\mathbf{x}) + \epsilon \\ &= \sum_{j=1}^{m} \sum_{k=1}^{r} \beta_{jk} h_{jk}(\mathbf{X}) + \epsilon, \end{aligned} \tag{2.41}$$

and using $\mu = \mathbb{E}(Y|\mathbf{X})$, we have

$$\begin{aligned} \mu(\mathbf{X}|\boldsymbol{\beta}) = \mathbb{E}(Y|\mathbf{X}) &= \mathbb{E}\left[\left(\sum_{j=1}^{m} \sum_{k=1}^{r} \beta_{jk} h_{jk}(\mathbf{X}) + \epsilon \right) |\mathbf{X} \right] \\ &= \left(\sum_{j=1}^{m} \sum_{k=1}^{r} \beta_{jk} h_{jk}(\mathbf{X}) | \mathbf{X} \right) + \mathbb{E}(\epsilon) \\ &= \sum_{j=1}^{m} \sum_{k=1}^{r} \beta_{jk} h_{jk}(\mathbf{X}), \end{aligned} \tag{2.42}$$

hence, we show μ is a random variable in terms of the linear transformation of $h_{jk}(\mathbf{X})$ for all $j \leq m$ and $k \leq r$ depends on some coefficients $\boldsymbol{\beta} \in \mathbb{R}^{m \times r}$. □

In which compared to the original expression in Definition 2.2.1, we are able to change the non-linear problem of finding the $m > 0$ number of unknown basis function $b(\cdot)$ into linear problem of finding the unknown predictor coefficients $\boldsymbol{\beta} \in \mathbb{R}^{m \times r}$. Hence, we are able to apply maximum likelihood estimation like in the case of simple linear regression to solve the solutions for $\boldsymbol{\beta}$.

Definition 2.2.3 (maximum likelihood estimation for generalised additive model)**.** Using Maximum Likelihood Estimation method from Definition 2.1.5 and for all $i = \{1, ..., n\}$ observations of $y_i \in Y$ and $\mathbf{x}_i \in \mathbf{X}$, we have $\boldsymbol{\beta} = \hat{\boldsymbol{\beta}}_{mle}$ such that

$$\hat{\boldsymbol{\beta}}_{mle} = \arg\max_{\boldsymbol{\beta}} \prod_{i=1}^{n} f(y_i | g(\mu(\mathbf{x}_i | \boldsymbol{\beta}))), \tag{2.43}$$

where $f(\cdot)$ be the probability density function of the response variable Y with assumed distribution from exponential family with estimated parameters $\theta = g(\mu)$ for some canonical link function $g(\cdot)$.

Knowing the fundamentals of generalised additive model, we are interested to learn the application of it. In the next subsection we will discuss the implementation of generalised additive model to approximate a response variable Y using one covariate X.

2.3 Generalised additive model on univariate observation

This subsection concerns about the implementation of generalised additive model to model the relationship between one response variable $Y \in \mathbb{R}$ and one covariate or univariate $X \in \mathbb{R}$. Therefore, considering we have $n \in \mathbb{Z}^+$ number of observations $y \in Y$ and $x \in X$. Due simplicity, we fixed the number of basis function to $m = 1$ and by using general definition of generalised additive model we can model the responses as follows,

$$y_i = b(x_i) + \epsilon_i = \sum_{k=1}^{r} \beta_k h_k(x_i) + \epsilon_i, \tag{2.44}$$

where $\epsilon_i \overset{\text{iid}}{\sim} N(0, \sigma^2)$, for some $\sigma^2 > 0$. In real scenario we are free to choose the our desired definition for the type of transformation of our univariate $h_k(x)$ for all $k = \{1, ..., r\}$. However, it is important to control the complexity of the basis function in order to avoid overfitting in our model. In this case, we start with the basic implementation of cubic polynomial function for $f(\cdot)$. Hence, our model be defined as follows,

$$Y_i = \beta_0 + \beta_1 X_i + \beta_2 X_i^2 + \beta_3 X_i^3 + \epsilon_i. \tag{2.45}$$

Using Lemma 2.2.1, we can express the estimation for Y given observation of univariate $x \in X$ as

$$\begin{aligned}\mu(x|\boldsymbol{\beta}) &= \mathbb{E}(Y|X=x) \\ &= \sum_{i=0}^{3} \beta_i x^i \\ &= \beta_0 + \beta_1 x_i + \beta_2 x_i^2 + \beta_3 x_i^3.\end{aligned} \tag{2.46}$$

2.3.1 Modelling simulation using cubic polynomial basis function

Now, we will try simulating cubic polynomial model into our toy data set of responses $y2 \in Y2$ conditioned by univariate $x \in X$ from Figure 2.1b using the maximum likelihood estimation method from definition 2.2.3. In doing so, we need to first augment the univariate X with the cubic polynomial transformation to get the covariates $\mathbf{X} = \{X, X^2, X^3\}$.

```
X <- cbind(x, x^2, x^3)
```

Then, we fit the cubic polynomial regression onto response $Y2$ using `glm` function.

```
fit <- glm(Y2 ~ X)
```

Next, we compute the mean squared error MSE($\boldsymbol{\beta}; Y2$) of the cubic polynomial model `fit` using `fit$residuals`.

```
fit_mse <- sum(fit$residuals^2) / length(x)
fit_mse

#[1] 0.991202
```

Finally, we plot the fitted values using each components of the cubic polynomial model `fit`.

```
plot(x, y2, xlab = "X", ylab = "Y2")
lines(x, t(fit$coefficients) %*% t(cbind(1, X)), lwd = 2, col="blue")

plot(x, rep(1,length(x)) * fit$coefficients[1], type = "l", col="blue",
    xlab = "x", ylab = expression(beta[0]))
plot(x, x * fit$coefficients[2], type = "l", col="blue", xlab = "x", ylab =
     expression(beta[1] * x))
plot(x, x^2 * fit$coefficients[3], type = "l", col="blue", xlab = "x", ylab
     = expression(beta[2] * x^2))
plot(x, x^3 * fit$coefficients[4], type = "l", col="blue", xlab = "x", ylab
     = expression(beta[3] * x^3))
```

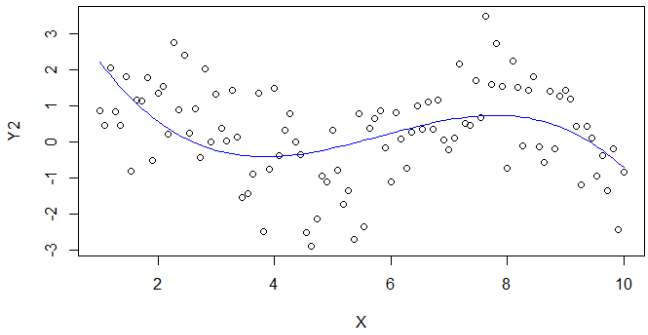
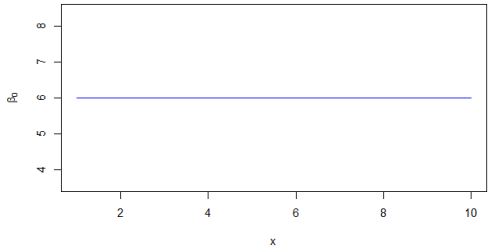
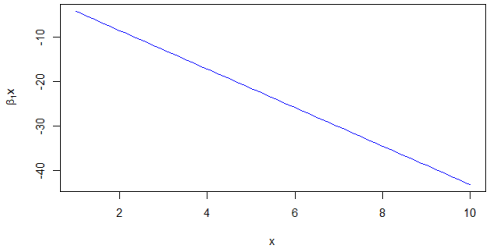


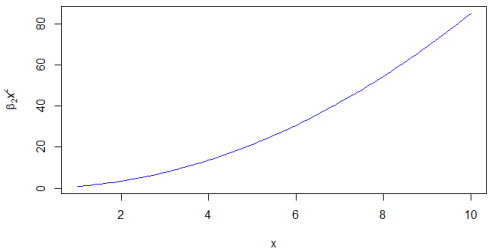
Figure 2.2: Estimation $\mu(x|\boldsymbol{\beta})$ against x using cubic polynomial.



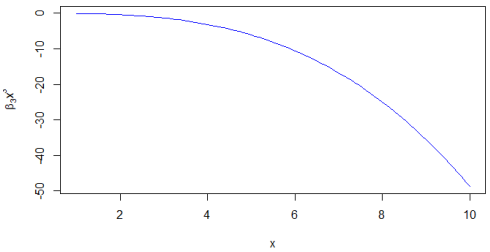
(a) β_0 against X.



(b) $\beta_1 X$ against X.



(c) $\beta_2 X^2$ against X.



(d) $\beta_3 X^3$ against X.

Figure 2.3: Representation of polynomial components $\beta_i x^i$ against X for $i = \{0, ..., 3\}$.

As we can see from Figure 2.2, the model estimation for $\mu(X|\boldsymbol{\beta})$ seems to fit better in comparison to the former example in Figure 2.1b. Adding to that, the mean squared error for cubic polynomial model is $\text{MSE}(\boldsymbol{\beta}; Y2) = 0.991202$ which is smaller in comparison to the simple linear model (i.e. $\text{MSE}(\boldsymbol{\beta}; Y2) \approx 1.39$) to further shows better fitting of regression model onto non-linear data $Y2$. However, cubic polynomial model like such can only get you so far in estimating the non-linear interaction between two variables. Since we predetermined the degree of our polynomial we are restricted to only a certain level of fitness to our model. Thus, there is a tendency for information loss due to underfitting. Concerning to this we want to increase the goodness-of-fit of our model. To accomplished such, we can introduced knots to our cubic polynomial model in which we partitioned our observations into ranges of values so that we can plot cubic polynomial functions to each knots of observations based on the given univariate. This method implemented cubic regression spline function to enhance the goodness-of-fit for the model.

2.3.2 Application of cubic regression spline function in generalised additive model

Definition 2.3.1 (Cubic Regression Spline (pp. 134-135, [Whi+24]). In the context of univariate model, given response variable Y and univariate $X \in \mathbb{R}$ and let $K > 0$ be the number of knots denoted by $\{\xi_j : j = 1, ..., K\}$. The estimate of Y that leverages cubic regression spline function is defined as follows

$$\mu(X|\boldsymbol{\beta}) = \mathbb{E}(Y|X) = \sum_{i=0}^{K+3} \beta_i h_i(X), \tag{2.47}$$

such that for $i \leq 3$, $h_i(X) = X^i$ and for $i > 3$, we have the following

$$h_i(X) = (X - \xi_{i-3})_+^3 = \begin{cases} (X - \xi_{i-3})^3, & \text{if } X > \xi_{i-3} \\ 0, & \text{otherwise} \end{cases} \tag{2.48}$$

in which $\beta_i \in \mathbb{R}$ for all $i = \{0, ..., K + 3\}$ is the predictor coefficients to be estimated.

Using Definition 2.2.2, we can apply cubic regression spline model onto response $Y2$ and univariate X in R by using `gam` function from `mgcv` package. First, we store our observations for response $y2 \in Y2$ and univariate $x \in X$ into dataframe format.

```
df <- data.frame(X = x, Y2 = y2)
```

Then, we fit additive model using `gam` function. We set `bs="cr"` to indicate cubic regression spline functions

```
fitSpline <- gam(Y2 ~ s(X, bs="cr"), data=df)
```

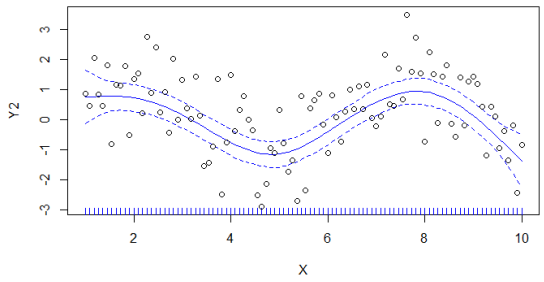


Figure 2.4: Estimation $\mu(x|\boldsymbol{\beta})$ against x using cubic regression spline.

Finally, we compute the mean squared error MSE($\boldsymbol{\beta}$; $Y2$) of the cubic spline model `fitSpline` using `fitSpline$residuals`.

```
fitSpline_mse <- sum(fitSpline$residuals^2) / length(x)
fitSpline_mse
```

Output :

```
[1] 0.7941665
```

Compared to other previous model, the cubic regression spline model in Figure 3.1 fits the observations better than the cubic polynomial model in Figure 2.2. By visually comparing the results, the latter able to capture more significant fluctuation of responses compared to the former. In addition, the mean squared error for cubic spline model is $\text{MSE}(\boldsymbol{\beta}; Y2) = 0.7941665$ which is smaller in comparison to both of the previous models to show that it has the best fit among others in modelling non-linear relationship between response $Y2$ and univariate x. However when implementing cubic regression spline or generalised additive model with multiple additive effect to improve the goodness-of-fit, it is also crucial to address the tendency for overfitting in the selection of our linear model estimates.

2.3.3 Bias-variance decomposition in linear modelling

Overfitting happens when the model are too sensitive to the training observations resulting in capturing every tiny propagation that may be considered as noises. The conflicts of dealing with the tendencies for both underfitting and overfitting is what we called as bias-variance decomposition. To understand how bias-variance decomposition works, we will focus on its effect onto the expected squared error which is the metric that we currently use in assessing the performance of our model estimator in previous Subsection 2.3.1 and Subsection 2.3.2.

Proposition 2.3.1 (Bias-Variance Decomposition on expected squared error (p.5, [And23]))**.** Let $\hat{\mu} \in \mathbb{R}$ be the linear estimate of response $Y \in \mathbb{R}$ conditioned on $\mathbf{X} \in \mathbf{R}^p$. Then, by Definition 2.3.1 and Definition 2.1.2, for some random error $\epsilon \sim \text{N}(0, \sigma^2)$ we have

$$\mathbb{E}[(Y - \hat{\mu}(\mathbf{X}))^2] = \mathbb{E}[bias(\hat{\mu})^2] + Var(\hat{\mu}) + \sigma^2, \tag{2.49}$$

such that $bias(\hat{\mu}) = \mathbb{E}[\hat{\mu}(\mathbf{X})] - \mu(\mathbf{X})$ whereas $Var(\hat{\mu}) = \mathbb{E}[(\hat{\mu}(\mathbf{X}) - \mathbb{E}[\hat{\mu}(\mathbf{X})])^2]$.

Proof. Consider the expectation of the squared error, $\mathbb{E}[(Y - \hat{\mu}(\mathbf{X}))^2]$. By adding and subtracting $\mathbb{E}[\hat{\mu}(\mathbf{X})]$ from the expression we have,

$$\begin{aligned} \mathbb{E}[(Y - \hat{\mu}(\mathbf{X}))^2] &= \mathbb{E}[(Y - \mathbb{E}[\hat{\mu}(\mathbf{X})] + \mathbb{E}[\hat{\mu}(\mathbf{X})] - \hat{\mu}(\mathbf{X}))^2] \\ &= \mathbb{E}[(Y - \mathbb{E}[\hat{\mu}(\mathbf{X})])^2 + (\mathbb{E}[\hat{\mu}(\mathbf{X})] - \hat{\mu}(\mathbf{X}))^2 \\ &\quad - 2(Y - \mathbb{E}[\hat{\mu}(\mathbf{X})])(\mathbb{E}[\hat{\mu}(\mathbf{X})] - \hat{\mu}(\mathbf{X}))], \end{aligned} \tag{2.50}$$

By using invariant property of expectation we can separate (2.50) as follows,

$$\begin{aligned} \mathbb{E}[(Y - \hat{\mu}(\mathbf{X}))^2] &= \mathbb{E}[(Y - \mathbb{E}[\hat{\mu}(\mathbf{X})])^2] + \mathbb{E}[(\mathbb{E}[\hat{\mu}(\mathbf{X})] - \hat{\mu}(\mathbf{X}))^2] \\ &\quad \mathbb{E}[-2(Y - \mathbb{E}[\hat{\mu}(\mathbf{X})])(\mathbb{E}[\hat{\mu}(\mathbf{X})] - \hat{\mu}(\mathbf{X}))] \\ &= \mathbb{E}[(Y - \mathbb{E}[\hat{\mu}(\mathbf{X})])^2] + Var(\hat{\mu}) \\ &\quad - 2\mathbb{E}[(Y - \mathbb{E}[\hat{\mu}(\mathbf{X})])(\mathbb{E}[\hat{\mu}(\mathbf{X})] - \hat{\mu}(\mathbf{X}))], \end{aligned} \tag{2.51}$$

where we expressed $Var(\hat{\mu}) = \mathbb{E}[(\hat{\mu}(\mathbf{X}) - \mathbb{E}[\hat{\mu}(\mathbf{X})])^2]$ in the last equality of (2.51). To solve (2.51), we consider the expression on the right hand side individually. We start off with $\mathbb{E}[(Y - \mathbb{E}[\hat{\mu}(\mathbf{X})])^2]$. By using Definition 2.1.1 and Lemma 2.1.1 to expand $Y = \mu(\mathbf{X}) + \epsilon$

for some independent random variable $\epsilon \sim \mathrm{N}(0, \sigma^2)$ we have

$$\begin{aligned}
\mathbb{E}[(Y - \mathbb{E}[\hat{\mu}(\mathbf{X})])^2] &= \mathbb{E}[(\mu(\mathbf{X}) + \epsilon - \mathbb{E}[\hat{\mu}(\mathbf{X})])^2] \\
&= \mathbb{E}[(\mu(\mathbf{X}) - \mathbb{E}[\hat{\mu}(\mathbf{X})])^2 + \epsilon^2 - 2\epsilon(\mu(\mathbf{X}) - \mathbb{E}[\hat{\mu}(\mathbf{X})])] \\
&= \mathbb{E}[(\mu(\mathbf{X}) - \mathbb{E}[\hat{\mu}(\mathbf{X})])^2] + \mathbb{E}[(\epsilon^2)] - 2\mathbb{E}[\epsilon(\mu(\mathbf{X}) - \mathbb{E}[\hat{\mu}(\mathbf{X})])] \\
&= \mathbb{E}[(\mu(\mathbf{X}) - \mathbb{E}[\hat{\mu}(\mathbf{X})])^2] + Var(\epsilon) + \mathbb{E}[\epsilon]^2 - 2\mathbb{E}[\epsilon]\mathbb{E}[\mu(\mathbf{X}) - \mathbb{E}[\hat{\mu}(\mathbf{X})]] \\
&= \mathbb{E}[(\mu(\mathbf{X}) - \mathbb{E}[\hat{\mu}(\mathbf{X})])^2] + \sigma^2 = \mathbb{E}[bias(\hat{\mu})^2] + \sigma^2,
\end{aligned} \tag{2.52}$$

where we use invariant property of expectation for the second equality and $Var(\epsilon) = \mathbb{E}[\epsilon^2] - \mathbb{E}[\epsilon]^2$ for the third equality in (2.52). Whereas for the fourth equality, we use the properties of ϵ where $\mathbb{E}[\epsilon] = 0$ and $Var(\epsilon) = \sigma^2$. Then, for the final expression we have

$$\begin{aligned}
\mathbb{E}[(Y - \mathbb{E}[\hat{\mu}(\mathbf{X})])(\mathbb{E}[\hat{\mu}(\mathbf{X})] - \hat{\mu}(\mathbf{X}))] &= \mathbb{E}[Y\mathbb{E}[\hat{\mu}(\mathbf{X})] - Y\hat{\mu}(\mathbf{X}) + \mathbb{E}[\hat{\mu}(\mathbf{X})]^2 - \hat{\mu}(\mathbf{X})\mathbb{E}[\hat{\mu}(\mathbf{X})]] \\
&= \mathbb{E}[Y\mathbb{E}[\hat{\mu}(\mathbf{X})]] - \mathbb{E}[Y\hat{\mu}(\mathbf{X})] + \mathbb{E}[\mathbb{E}[\hat{\mu}(\mathbf{X})]^2] - \mathbb{E}[\hat{\mu}(\mathbf{X})\mathbb{E}[\hat{\mu}(\mathbf{X})]] \\
&= \mathbb{E}[Y]\mathbb{E}[\mathbb{E}[\hat{\mu}(\mathbf{X})]] - \mathbb{E}[Y]\mathbb{E}[\hat{\mu}(\mathbf{X})] \\
&\quad + \mathbb{E}[\mathbb{E}[\hat{\mu}(\mathbf{X})]^2] - \mathbb{E}[\hat{\mu}(\mathbf{X})]\mathbb{E}[\mathbb{E}[\hat{\mu}(\mathbf{X})]]
\end{aligned} \tag{2.53}$$

where we use invariant property of expectation for the second equality and independent of random variable for the third equality in (2.53). By using Lemma 2.1.1 to express $\mu(\hat{\mathbf{X}}) = \mathbb{E}[Y|\mathbf{X}]$ and using Tower Property of Conditional Expectation where $\mathbb{E}[\mathbb{E}[Y|\mathbf{X}]] = \mathbb{E}[Y] \in \mathbb{R}$ we have the following

$$\begin{aligned}
\mathbb{E}[(Y - \mathbb{E}[\hat{\mu}(\mathbf{X})])(\mathbb{E}[\hat{\mu}(\mathbf{X})] - \hat{\mu}(\mathbf{X}))] &= \mathbb{E}[Y]\mathbb{E}[\mathbb{E}[\hat{\mu}(\mathbf{X})]] - \mathbb{E}[Y]\mathbb{E}[\hat{\mu}(\mathbf{X})] \\
&\quad + \mathbb{E}[\mathbb{E}[\hat{\mu}(\mathbf{X})]^2] - \mathbb{E}[\hat{\mu}(\mathbf{X})]\mathbb{E}[\mathbb{E}[\hat{\mu}(\mathbf{X})]] \\
&= \mathbb{E}[Y]\mathbb{E}[\mathbb{E}[Y]] - \mathbb{E}[Y]\mathbb{E}[Y] + \mathbb{E}[\mathbb{E}[Y]^2] - \mathbb{E}[Y]\mathbb{E}[\mathbb{E}[Y]] \\
&= \mathbb{E}[Y]^2 - \mathbb{E}[Y]^2 = 0.
\end{aligned} \tag{2.54}$$

Therefore, by adding all component from (2.52) and (2.54) into (2.51) we obtain

$$\begin{aligned}
\mathbb{E}[(Y - \hat{\mu}(\mathbf{X}))^2] &= \mathbb{E}[bias(\hat{\mu})^2] + \sigma^2 + Var(\hat{\mu}) - 2 \cdot 0 \\
&= \mathbb{E}[bias(\hat{\mu})^2] + \sigma^2 + Var(\hat{\mu})
\end{aligned} \tag{2.55}$$

□

In short, according to Avati (2020) we say a model underfits the data when it has large bias but small variance whereas a model overfits the data if has large variance but small bias [Ava20]. Hence, in general routine of modelling one should always consider the tradeoff between bias and variance in order to obtain the most optimal solution in linear modelling. In the case of generalised additive model, the common practice in tackling this issue is by adding some smoothing component to our objective function in Definition 2.2.3 to create a penalised generalised additive model.

2.3.4 Addition of penalty component in generalised additive model

Definition 2.3.2 (Penalised Generalised Additive Model (pp. 1549, [WPS16]))**.** Given $n > 0$ number of observations for response $Y \in \mathbb{R}$ and univariate $X \in \mathbb{R}$. Let $h_k(\cdot)$ be fixed linearly independent transformations for all $k = \{1, ..., r\}$ and $r > 0$. Then, the estimation

of Y denoted by $\mu(\mathbf{X}|\boldsymbol{\beta}) = \sum_{k=1}^{r} \beta_{jk} h_{jk}(\mathbf{X})$ is best defined using predictor coefficients $\boldsymbol{\beta} = \hat{\boldsymbol{\beta}}_{\lambda,mle}$ such that

$$\hat{\boldsymbol{\beta}}_{\gamma,mle} = \arg\max_{\boldsymbol{\beta}} \sum_{i=1}^{n} \log\{f(y_i|g(\mu(\mathbf{x}_i|\boldsymbol{\beta}))\} - \gamma\boldsymbol{\beta}^{\intercal}\boldsymbol{S}\boldsymbol{\beta}, \tag{2.56}$$

where $f(\cdot|\theta)$ be the assumed probability distribution function of $Y|X$ while $g(\cdot)$ be logit link function such that $g(\mu) = \theta$ and for some positive semidefinite matrix $\boldsymbol{S} \in \mathbb{R}^{r\times r}$ and a non-negative value $\gamma \in \mathbb{R}$.

Example 4. This penalty effect can also be understood in terms of graphical representation for the predictor coefficients $\boldsymbol{\beta}$ against the chosen values of tuning hyperparameter $\gamma > 0$. Below is a results of fitting cubic regression spline with $K = 10$ number of knots onto response `RelativeWeight` with univariate `FastingPlasmaGlucose` of `diabetes` dataset obtained from `loon.data` package in R.

Firstly, we load `loon.data` package and initialised dataset `diabetes`.

```
library(loon.data)
data(diabetes)
```

Then, we initialise a zero matrix to store the values of the predictor coefficients $\boldsymbol{\beta}$ that will leverage different tuning hyperparameter $\gamma = \{0.01, ..., 0.1\}$.

```
beta <- matrix(0, 10, 10)
```

After that, we trained cubic regression spline onto observations for `RelativeWeight` and univariate `FastingPlasmaGlucose` using `gam` function with different values of $\gamma = \{0.01, ..., 0.1\}$.

```
for (i in 1:10){
    gamma <- 0.01 * i
    fit <- gam(RelativeWeight ~ s(FastingPlasmaGlucose, k = 10, bs = "cr"),
               data = diabetes, scale = gamma)
    beta[i,] <- fit$coefficients
}
```

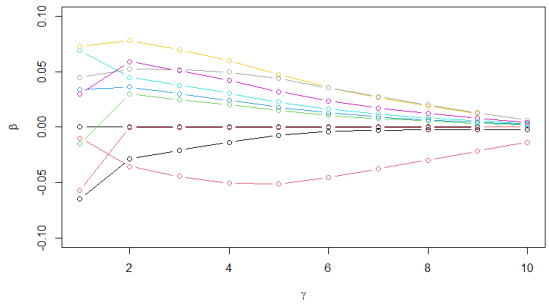


Figure 2.5: Affect of tuning parameter γ onto predictor coefficients $\boldsymbol{\beta}$

Based on Figure 2.5, we can see how every predictor $\boldsymbol{\beta}$ tends to zero as the value of tuning parameter γ increases. This shows that increasing in the value of the tuning hyperparameter will result in less degree of dependence between the response and the covariates. Thus, we can reduce the overfitting affect of our model by choosing suitable value for our tuning parameter γ.

Chapter 3

Quantile Regression

Before delving into the fundamentals in quantile regression, it is important to clarify some basic terminologies. Firstly, what do we meant by quantile of a population? At the highest level, quantile of a population is a specified value at which a portion of the population relatives to the total number of population falls below the specified value. To get a better grasp of it, take an analogy of having a population of 100 participants in a fishing event. Consider that $\tau = 0.9$ and the τ-th quantile of the number of fish caught by each participants be $\mu_\tau = 23$. From the information of the value of μ_τ, we know that there are 90 participants that caught less than 23 fish whereas the top ten participants caught more than or equal to 23 fish repectively. In the context of probability, [SW24] formally defined τ-th quantile μ_τ as follows (Stover and Weisstein, 2024).

Definition 3.0.1 (Quantile [SW24])**.** Let $\tau \in (0, 1)$ and $Y \in \mathbb{R}$ be a random variable distributed by known distribution with probability density function $f(y) = \mathbb{P}(Y = y)$. τ-th quantile of Y, denoted by μ_τ be defined as

$$\mu_\tau = F^{-1}(\tau) = \inf\{y : F(y) \geq \tau\}, \tag{3.1}$$

where $F(y) = \mathbb{P}(Y \leq y)$ is the cumulative distribution function of y.

3.1 Derivation of quantile in terms of conditional probability

Referring to Definition 3.0.1 we know that in order to obtain μ_τ, we need to be well informed of the exact distribution of the variable Y, However, in conventional settings of observations in a sample we cannot be so sure regarding the distribution of the population of the response variable. This issue further becomes more complicated when we consider the dependency of Y on covariates, $\mathbf{X} \in \mathbb{R}^p$. In which by substuting Y in Definition 3.0.1 by the conditional random variable $Y|\mathbf{X}$ we have the formal definition for conditional quantile denoted by $\mu_\tau(\mathbf{X})$.

Definition 3.1.1 (Conditional Quantile)**.** Let $\tau \in (0, 1)$ and $Y \in \mathbb{R}$ and $\mathbf{X} \in \mathbb{R}^p$ be a random variable distributed by known distributions. τ-th quantile of Y conditioned on $\mathbf{X}$, denoted by $\mu_\tau(\mathbf{x})$ be defined as

$$\mu_\tau(\mathbf{x}) = F^{-1}_{y|\mathbf{x}}(\tau) = \inf\{y : F_{y|\mathbf{x}}(y|\mathbf{x}) \geq \tau\}, \tag{3.2}$$

where $F_{y|\mathbf{x}}(y|\mathbf{x}) = \mathbb{P}(Y \leq y|\mathbf{X} = \mathbf{x})$ be the cumulative distribution function of Y conditioned on $\mathbf{X}$.

By Definition 3.1.1, it seems pretty convenient to derive the conditional quantile for our response variable if we know the exact probability distribution of $Y|\mathbf{X}$. However, in the usual context of modelling we are fairly uncertain about the true behaviour of the population of response or even the covariates. Thus, one way of computing the conditional quantile $\mu_\tau(\mathbf{X})$ is by assuming the conditional distribution of $Y|\mathbf{X}$. But, the ease of doing so may results in poor quantile estimation due to exaggerated assumptions. To understand the risk of probabilistic assumption on a conditional variable, we will start by deducing the closed form of the cumulative distribution function of Y conditioned on $\mathbf{X}$.

Definition 3.1.2 (Conditional Probability Density Function (Section 12.1, [NC22]))**.** Consider random variables $Y \in \mathbb{R}$ and $\mathbf{X} \in \mathbb{R}^p$, the conditional probability density function of $Y|\mathbf{X}$ be defined as

$$\begin{aligned} \mathbb{P}(Y = y|\mathbf{X} = \mathbf{x}) &= f_{Y|\mathbf{X}}(y|\mathbf{x}) \\ &= \frac{f_{Y,\mathbf{X}}(y,\mathbf{x})}{f_{\mathbf{X}}(\mathbf{x})} \\ &= \frac{\mathbb{P}(Y = y, \mathbf{X} = \mathbf{x})}{\mathbb{P}(\mathbf{X} = \mathbf{x})}, \end{aligned} \tag{3.3}$$

where $f_{Y,\mathbf{X}}(y,\mathbf{x}) = \mathbb{P}(Y = y, \mathbf{X} = \mathbf{x})$ is the joint probability density function of Y and $\mathbf{X}$.

referring to Definition 3.1.2, we can derive the conditional cumulative distribution function, $F_{y|\mathbf{x}}$ as

$$F_{y|\mathbf{x}}(y|\mathbf{x}) = \frac{F_{Y,\mathbf{X}}(y,\mathbf{x})}{f_{\mathbf{X}}(\mathbf{x})}, \tag{3.4}$$

in which from the above we expressed $F_{y|\mathbf{x}}(\cdot|\mathbf{x})$ in terms of probability density function of $\mathbf{X} = \mathbf{x}$ and joint cumulative distribution function $F_{Y,\mathbf{X}}(\cdot,\mathbf{x})$.

Proof. Let $Y \in \mathbb{R}$ and $\mathbf{X} \in \mathbb{R}^p$ be random variables with conditional probability density function defined as in Definition 3.1.2. Referring to Joyce (2014) [Joy14], we can expand $F_{y|\mathbf{x}}$ as follows,

$$\begin{aligned} F_{Y|\mathbf{X}}(y|\mathbf{x}) &= \frac{\mathbb{P}(Y \leq y, \mathbf{X} = \mathbf{x})}{\mathbb{P}(\mathbf{X} = \mathbf{x})} \\ &= \int_{-\infty}^{y} \frac{f_{Y,\mathbf{X}}(t,\mathbf{x})}{f_{\mathbf{X}}(\mathbf{x})}\, dt \\ &= \frac{1}{f_{\mathbf{X}}(\mathbf{x})} \int_{-\infty}^{y} f_{Y,\mathbf{X}}(t,\mathbf{x})\, dt, \end{aligned} \tag{3.5}$$

where the conditional cumulative distribution function $F_{Y|\mathbf{X}}(\cdot|\mathbf{x})$ can be expressed in terms of the integral of the joint probability density function $f_{Y,\mathbf{X}}(\cdot,\mathbf{x})$ (Balazs and Toth, 2022 [BT22]). Finally, we used the definition of cumulative distribution function to simplify the integral part as

$$\int_{-\infty}^{y} f_{Y,\mathbf{X}}(t,\mathbf{x})\, dt = 0 + \int_{0}^{y} f_{Y,\mathbf{X}}(t,\mathbf{x})\, dt \quad = F_{Y,\mathbf{X}}(y,\mathbf{x}). \tag{3.6}$$

□

Using the fact in equation (3.4), we acknowledge the relationship between the conditional distribution of $Y|\mathbf{X}$ and both the joint distribution of $(Y, \mathbf{X})$ as well as the distribution of $\mathbf{X}$. Hence, in order to determine the τ-th conditional quantile $\mu_\tau(\mathbf{X})$ we have to make a considerably huge parametric assumptions on the probability distribution of our observations for responses and covariates. Nonetheless, due simplicity in its methodology we will attempt on attaining the estimate for conditional quantile by imposing parametric assumptions on our response variable.

3.2 Parametric approach to estimate conditional quantile

In this subsection, we will go through the theory of modelling conditional quantile, $\mu_\tau(\cdot)$ using parametric approach. Then, we will simulate the modelling procedure on the Bristol Air Quality data to assess the accuracy of the modelling technique. In order to understand how we can approach this problem using parametric method we recall the Lemma 2.2.1. In which by making assumption on the distribution of response Y conditioned on covariates $\mathbf{X}$, we can define the estimation of Y as the first order expectation or $\mu(\mathbf{X}|\boldsymbol{\beta}) = \mathbb{E}(Y|\mathbf{X})$. Using the same narrative to the former we can further extend the capability of our assumption to estimate the conditional quantile instead of the expectation of the conditional variable $Y|X$. In general scenario this can be done due to the fact that we can define the cumulative distribution function $F(\cdot|\boldsymbol{\theta})$ explicitly, considering that we can express the parameter $\theta \in \boldsymbol{\theta}$ of our chosen distribution as a function of the estimator $\theta = g(\mu(\mathbf{X}|\boldsymbol{\beta}))$ where $g(\cdot)$ is the canonical link function. In a more formal way, we can define the parametric approach for conditional quantile estimate as follows.

Definition 3.2.1 (Parametric Approach for Conditional Quantile Estimation)**.** Given a response Y and covariates $\mathbf{X}$, let $Y|\mathbf{X}$ to be distributed by exponential family distribution with unknown parameters $\boldsymbol{\theta}$. By considering the estimator $\mu(\mathbf{X}|\boldsymbol{\beta}) = \mathbb{E}(Y|\mathbf{X})$ such that $\mu(\mathbf{X}|\boldsymbol{\beta}) = g^{-1}(\boldsymbol{\theta})$ for some link functions $g(\cdot)$. We can estimate the τ-th conditional quantile for $\tau \in (0, 1)$ as follows,

$$\hat{\mu}_\tau(\mathbf{x}) = F^{-1}_{y|\mathbf{x}}(\tau|g(\mu(\mathbf{x}|\boldsymbol{\beta}))), \tag{3.7}$$

where $F^{-1}_{y|\mathbf{x}}(\cdot|g(\mu(\mathbf{X}|\boldsymbol{\beta})))$ be the inverse of the cumulative distribution function for conditional variable $Y|\mathbf{X}$ with parameters $\boldsymbol{\theta} = g(\mu(\mathbf{X}|\boldsymbol{\beta})$.

In this modelling method, we need to firstly train our model using the basic approach from previous section in order to obtained the predictor coefficients $\boldsymbol{\beta}$ to solve the linear problem of estimating $\mu(\mathbf{X}|\boldsymbol{\beta})$. In which, to do so we implement generalised additive model to get the best fit based on our observations and in computing the inverse of the cumulative function, we can apply built-in function from R to find the conditional quantile of our response variable.

Example 5. Consider the case of response $Y2$ and univariate X from previous example. Say in this case, we want to compute the estimation for 0.95-th quantile of $Y2$ conditioned by univariate X by assuming that $Y2|X \sim N(\mu(\mathbf{X}|\boldsymbol{\beta}), \sigma^2)$. Where in this case, we fixed $\sigma = 1$.

First, we store our response `Y2` and `X` into dataframe format.

```
df <- data.frame(x = x, y2 = y2)
```

Then, we trained additive model to estimate $\mu(\mathbf{X}|\boldsymbol{\beta})$ using `gam` function. We set `bs="cr"` to implement cubic regression spline functions.

```
fitSpline <- gam(y2 ~ s(x, bs="cr"), data=df)
```

After that, we find the fitted values of `Y2` using all observations of `X` and the coefficients of `fitSpline` model.

```
y2pred <- predict(fitSpline)
```

Next, we compute the 0.95-th conditional quantile by leveraging `qnorm` function.

```
condQuan <- rep(0, 100)
for (i in 1:length(y2pred)){
    condQuan[i] <- qnorm(0.95, y2pred[i], 1)
}
```

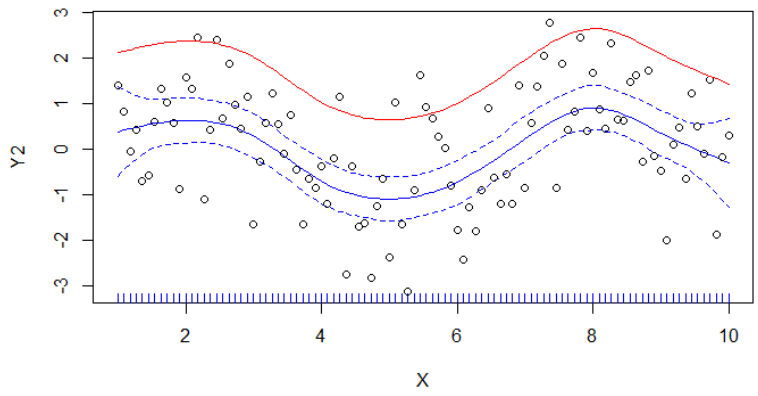


Figure 3.1: 0.95-th conditional quantile (red) and cubic regression spline (blue).

3.2.1 Simulation of parametric approach for conditional quantile estimation

Based on the example above, we can see how convenient it is to make conditional quantile estimator using parametric approach. Now, we are interested to find out how will it perform on the actual `X2021` dataset which contains 65735 observations that consist of 23 variables that records day to day information of Bristol air quality in 2021. In this project however, since our main concern is to study the non-linearity interaction between temperature and ozone concentration at the extreme quantile of $\tau = 0.95$, we will only be using two variables which is `O3` and `Temperature`. For simplicity, throughout the modelling procedure, we will be using `x` to represent `Temperature` whereas `y` to represent `O3`.

First, we store the observations into a dataframe and plot the data to briefly study the overall trend of observations.

```
data <- na.omit(data.frame(y = X2021$O3, x = X2021$Temperature))
data <- data[data$y > 0,]
x <- data$x
y <- data$y
plot(data$x, data$y, xlab = "Temperature", ylab = "O3")
```

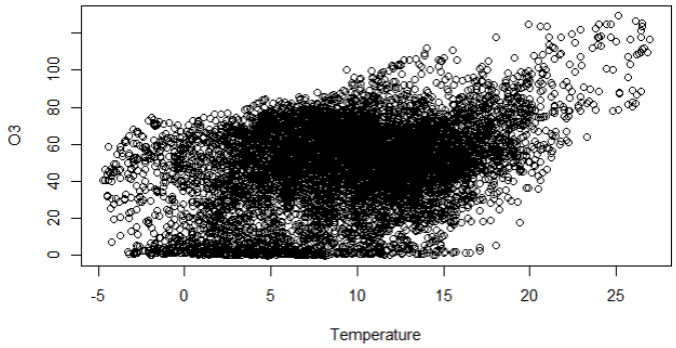


Figure 3.2: Concentration of ozone against atmospheric temperature.

To start the parametric approach for the 0.95-th conditional quantile estimate $\hat{\mu}_{0.95}(x)$ using the observations from Figure 3.2, we need to define the additive effect of our model. In which since we are using cubic regression spline function for goodness-of-fit, we apply `smoothCon` to create 20 cubic spline basis transformations of the observations `x`.

```
sm <- smoothCon(s(x, k = 20, bs = "cr"), data=data, knots=NULL)[[1]]
# We initialised the cubic spline basis transformations of x as
X <- sm$X
```

Then, we fit cubic regression spline model using `gam` function.

```
fit <- gam(y ~ X - 1, data = data)
plot(data$x, data$y, xlab = "Temperature", ylab = "O3", col = "grey")
lines(data$x[order(data$x)], X[order(data$x),]%*%fit$coefficients, col = "
    blue")
```

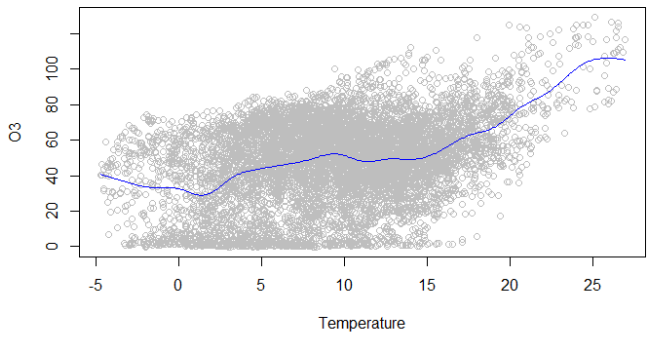


Figure 3.3: Cubic regression spline model of ozone concentration (blue).

Applying the procedures from Example 5, we compute the fitted values of `y` using `predict` function.

```
yPred <- predict(fit)
```

Then, we use normality assumption on the distribution of the conditional response $Y|X \sim N(\mu, \sigma^2)$ such that μ be the fitted values of `yPred` while σ^2 be estimated using Maximum Likelihood Estimation as in Example 2 to compute the parameters of the conditional distribution $Y|X$.

```
condQuan <- rep(0, length(yPred))
sigma <- sqrt(sum((data$y - yPred)^2) / length(yPred))
```

Finally, we compute the conditional quantile estimate $\hat{\mu}_{0.95}(x)$ by using R built-in function `qnorm`.

```
for (i in 1:length(yPred)){ condQuan[i] <- qnorm(0.95, yPred[i], sigma) }
plot(data$x, data$y, xlab = "Temperature", ylab = "O3", col = "grey")
lines(data$x[order(data$x)], condQuan[order(data$x)], col="red")
```

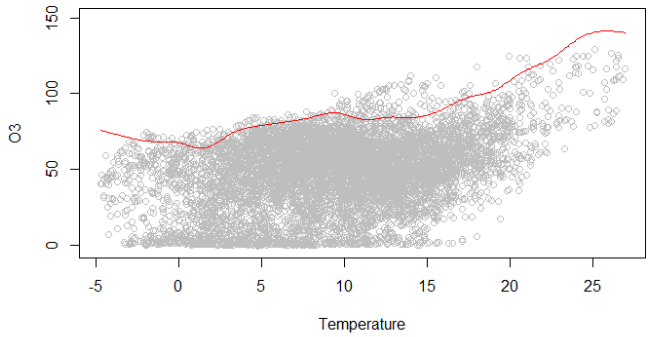


Figure 3.4: Parametric approach to estimate 0.95-th conditional quantile of ozone (red).

As we can see from the above diagram it is obvious that although we found quite a practical method to model the estimation of conditional quantile of the responses. It is still not fully represent the desired result for $\tau = 0.95$ quantile of the observations. This poor performance of estimation is further shown through the investigation on the relative comparison between the actual observations with the fitted conditional quantile estimates. Thus, in the next subsection we will introduce some feasible method in assessing the performance of the modelling approach in estimating the extreme 0.95-th conditional quantile of response `O3`.

3.3 Performance Analysis

In this subsection, we are interested to learn how well our modelling approach works in estimating the conditional quantile of real observations like in the dataset `X2021`. Hence, we are in need of practical exercise to assess the accuracy of our fitted conditional quantile estimate. One way to do this is by making individual comparison between the values of fitted conditional quantile using observations of univariate and the values of actual observations of response. In doing so, we recall back the definition of quantile from Definition 3.0.1 where for $\tau \in (0, 1)$ we have $\mu_\tau = \inf\{y : F(y) \geq \tau\}$. By such, we know that given observations of univariate $x \in X$, for every estimated conditional quantile of $\hat{\mu}_{0.95}(x)$, we expect that around 95 percent of the observations of response $y \in Y$ conditioned on univariate x will be smaller than $\hat{\mu}_{0.95}(x)$. Equipped with this knowledge, we can construct a hypothesis testing to validate the accuracy of the conditional quantile estimate.

3.3.1 Hypothesis test to evaluate the accuracy of conditional quantile estimate

Let $W \in \{0,1\}$ be a random variable of Bernoulli distribution with parameter $\boldsymbol{\theta} = \{p\}$. Where in this case $p \in (0,1)$ be the probability for a success in a trial. In which in this context we define the observations of $w_i \in W$ for all $i \in \{1,...,n\}$ as

$$w_i = \begin{cases} 1, & \text{if } y_i < \hat{\mu}_\tau(x_i) \\ 0, & \text{otherwise} \end{cases}, \tag{3.8}$$

where $\tau \in (0,1)$ and $w_i \overset{iid}{\sim} Bernoulli(p)$ for $p = \mathbb{P}\{Y < \hat{\mu}_\tau(X)\}$. By the above conditions, we can construct the hypothesis statement as follows,

$$H_0 : p = \tau, \qquad\qquad H_1 : p \neq \tau \tag{3.9}$$

Now that we have our hypothesis statement, we should be able to leverage a reliable statistic in order to analyse the performance of our quantile estimation model. One way of getting around defining the suitable statistic for hypothesis testing is by applying maximum likelihood estimation.

Proposition 3.3.1. Let $n > 0$ number of observations for random variable $W \overset{iid}{\sim} Bernoulli(p)$ with unknown parameter $p \in (0,1)$. Using maximum likelihood estimation we define the unbiased estimator for p to be denoted as $p = \hat{p}_{mle}$ (Cain, 2023 [Cai23]) such that

$$\hat{p}_{mle} = \frac{\sum_{i=1}^{n} w_i}{n}. \tag{3.10}$$

Proof. Taking into consideration for $n > 0$ number of independent and identically distributed $\{w_1,...,w_n\} \in W \overset{iid}{\sim} Bernoulli(p)$. We can defined the likelihood function $L_n(p|\cdot)$ as the product of the marginal probability distribution function of individual observations w_i for all $1 \leq i \leq n$,

$$L_n(p|\mathbf{w}) = \prod_{i=1}^{n} p^{w_i}(1-p)^{1-w_i}, \tag{3.11}$$

to simplify our problem, we use log transformation for the expression (3.11) to get

$$\begin{aligned} \ell_n(p|\mathbf{w}) &= \log\big\{L_n(p|\mathbf{w})\big\} \\ &= \log\left\{\prod_{i=1}^{n} p^{w_i}(1-p)^{1-w_i}\right\} \\ &= \sum_{i=1}^{n} w_i \log(p) + (1-w_i)\log(1-p), \end{aligned} \tag{3.12}$$

then by taking the first derivative of (3.12) in terms of p, we obtain

$$\frac{\delta \ell_n(p|\mathbf{w})}{\delta p} = \sum_{i=1}^{n}\left(\frac{w_i}{p} - \frac{1-w_i}{1-p}\right), \tag{3.13}$$

after that we equate (3.13) with zero for solution of stationary point

$$\begin{aligned}\frac{\delta \ell_n(p|\mathbf{w})}{\delta p} = \sum_{i=1}^{n}\left(\frac{w_i}{p} - \frac{1-w_i}{1-p}\right) &= 0 \\ \frac{1}{p}\sum_{i=1}^{n} w_i &= \frac{1}{1-p}\sum_{i=1}^{n}(1-w_i) \\ \frac{1-p}{p} &= \frac{n - \sum_{i=1}^{n} w_i}{\sum_{i=1}^{n} w_i} \\ \hat{p}_{mle} &= \frac{\sum_{i=1}^{n} w_i}{n}.\end{aligned} \tag{3.14}$$

Hence, proving expression (3.10). To show that (3.13) is indeed a maximiser we check the second derivative,

$$\frac{\delta^2 \ell_n(p|\mathbf{w})}{\delta p^2} = -\sum_{i=1}^{n}\left(\frac{w_i}{p^2} + \frac{1-w_i}{(1-p)^2}\right) \leq 0, \tag{3.15}$$

for all $0 \leq w_i \leq 1$ and $p \in \mathbb{R}$. To show that (3.10) is unbiased we consider the formula of bias using expectation of (3.10),

$$\begin{aligned}bias(\hat{p}_{mle}) = \mathbb{E}[\hat{p}_{mle}] - p = \mathbb{E}\left[\frac{\sum_{i=1}^{n} w_i}{n}\right] - p \\ = \frac{1}{n}\sum_{i=1}^{n}\mathbb{E}[w_i] - p \\ = \frac{1}{n}\sum_{i=1}^{n} p - p = 0.\end{aligned} \tag{3.16}$$

Therefore, $\hat{p}_{mle}$ is the unbiased estimator that maximises the likelihood function (3.11). □

In performing the hypothesis test stated in (3.9), we apply Proposition 3.3.1 for our test statistic. In this project, we are considering for $\alpha = 0.05$ significance level and two-sided confidence interval using Wilson Score Interval [Wil27]. The reason for using Wilson Score Interval is to constrain the upper boundary of the confidence interval in order to not exceed the maximum value of probability which is 1 since we are dealing with extreme quantile of $\tau = 0.95$ (Wilson, 1927).

Definition 3.3.1 (Wilson Score Interval ([Wil27]))**.** For $n > 0$ number of independent and identically distributed observations of random variable $\{w_1, ..., w_n\} \in W \overset{iid}{\sim} Bernoulli(p)$. Let $\hat{p}$ be the estimate parameter for p, and $Z \sim N(0,1)$ such that for $t \in (0,1)$ we denote $z_t = F^{-1}(t)$. Thus, we determine the $1-\alpha$ confidence interval for the true parameter $p \in (CI_{lower}, CI_{upper})$ for a given significance level $\alpha \in (0,1)$ as

$$CI_{lower} = \frac{\hat{p} + \frac{z^2_{\alpha/2}}{2n} - z_{\alpha/2}\sqrt{\frac{\hat{p}(1-\hat{p})}{n} + \frac{z^2_{\alpha/2}}{4n^2}}}{1 + \frac{z_{\alpha/2}}{n}}, \tag{3.17}$$

$$CI_{upper} = \frac{\hat{p} + \frac{z^2_{\alpha/2}}{2n} + z_{\alpha/2}\sqrt{\frac{\hat{p}(1-\hat{p})}{n} + \frac{z^2_{\alpha/2}}{4n^2}}}{1 + \frac{z_{\alpha/2}}{n}}. \tag{3.18}$$

Hence, we will reject the null hypothesis if and only if $\tau \notin [CI_{lower}, CI_{upper}]$ defined as in Definition 3.3.1. Using this strategy, we simulate the test on the whole observations of response `y` and the fitted conditional quantile `yPred` to study the performance of the parametric approach using R. First we fixed the significance level $\alpha = 0.05$ and zero vector to store the observations of $W \sim Bernoulli(p)$.

```
a <- 0.05
w <- rep(0, length(y))
```

Then, conditioned by constraint for successes of W as in equation (3.8) and the fitted values for conditional quantile estimate `condQuan` we define observations $w \in W$.

```
for (i in 1:length(y)){
    if (y[i] <= condQuan[i]){
        w[i] <- 1
    }
}
```

Using the observations w, we compute our statistic $\hat{p}_{mle}$ using (3.10).

```
p_hat <- sum(w) / length(y)
```

With the estimate `p_hat` and using `qnorm` function to compute the 0.95 quantile from standard normal distribution we compute the Wilson score interval for $p = \mathbb{P}\{Y < \hat{\mu}_\tau(X)\}$ using Definition 3.3.1.

```
z <- qnorm(1 - a/2, 0, 1)
r1 <- p_hat + z^2 / (2 * n)
r2 <- z * sqrt((p_hat * (1 - p_hat)) / n + z^2 / (4 * n^2))
r3 <- 1 + z^2 / n
CI <- c((r1 - r2) / r3, (r1 + r2) / r3)
print(paste("The confidence interval for p is (",CI[1],", ",CI[2],")"))
```

Output :

```
[1] "The confidence interval for p is (0.969890892342623,0.97685276575318)"
```

From the above results, we obtained $0.970 \leq p \leq 0.977$. Hence, the result shows that $\tau \notin [CI_{lower}, CI_{upper}]$ and indicates enough evidence to reject the null hypothesis H_0. So, we know by fact that the parametric approach is not a good technique to be implemented in this scenario. This is due to the high biasness in our modelling approach of making greedy parametric assumptions on the distribution of the observations. Therefore, we need a better approach that makes less parametric assumption to improve the accuracy of our estimation for extreme conditional quantile for `O3` with respect to `Temperature`.

3.4 Non-parametric approach to estimate conditional quantile

In this subsection we will explore non-parametric approach in modelling the estimation for conditional quantile using real observations. The concept of non-parametric approach in modelling involves implementation of statistical method without any prior assumption to the probability distribution of our data. As such, this approach is more reliable in concluding an inference towards real observations. This is due to having less restrictions on

the observations to conserve the nature of the assumed distribution unlike in parametric method. In regards to the context of our problem previously, in parametric method we initiate the modelling procedure by using cubic regression spline model with normality assumptions. Thus, the relevance of applying maximum likelihood estimation in the objective function from Definition 2.2.3 to estimate the predictor coefficients $\boldsymbol{\beta} \in \mathbb{R}^p$ of the generalised additive model. Moving from this, one may wonder how can we solve the linear problem of finding the suitable predictor coefficients $\boldsymbol{\beta}$ if we do not make assumptions on the probability distribution of the data. In doing so, we will look into new ways of constructing our model objective function by firstly define the loss function for our model.

3.4.1 Introduction to pinball loss function and its role in quantile regression

In conventional modelling procedure, loss function indicates the error in our model where our goal in constructing a good model is by minimising the expectation of the loss function. In the context of quantile estimation, the loss function that can be used to quantify the accuracy of our quantile estimate is the assymmetric loss function or pinball loss function.

Definition 3.4.1 (Pinball Loss Function (pp. 5, [Koe05]))**.** Given the observation of random variable $y \in Y$ and for some $\mu \in \mathbb{R}$. Let $z = y - \mu$ and $\tau \in (0, 1)$, the pinball loss function is defined as follows

$$\rho_\tau(z) = \begin{cases} (\tau - 1)z, & \text{if } z < 0 \\ \tau z, & \text{if } z \geq 0. \end{cases} \tag{3.19}$$

Proposition 3.4.1. Referring to the pinball loss function from Definition 3.4.1, it can be shown that by solving the optimisation problem of minimising the expected pinball loss will direct us to finding the estimation of the τ-th quantile μ_τ for $\tau \in (0, 1)$.

Proof. By using the definition of stationary point we will show that μ that minimises the expected pinball loss is indeed the quantile of random variable $Y \in \mathbb{R}$. First, we start off by defining the definition of expected pinball loss for $\tau \in (0, 1)$,

$$\mathbb{E}[\rho_\tau(y - \mu)] = \int_{-\infty}^{\mu} \rho_\tau(y - \mu) dF(y), \tag{3.20}$$

then we expand $\rho(\cdot)$ in (3.20). Since pinball loss is conditioned by the sign of the input values, we need to consider for both cases.

$$\mathbb{E}[\rho_\tau(y - \mu)] = \begin{cases} \int_{-\infty}^{\mu} (\tau - 1)(y - \mu) dF(y), & \text{if } y < \mu \\ \int_{\mu}^{\infty} \tau(y - \mu) dF(y), & \text{if } y \geq \mu. \end{cases} \tag{3.21}$$

Then, by differentiating $\mathbb{E}[\rho_\tau(y - \mu)]$ in (3.21) in terms of μ we have

$$\begin{aligned} \frac{\delta \mathbb{E}[\rho_\tau(y - \mu)]}{\delta \mu} &= \begin{cases} (\tau - 1) \int_{-\infty}^{\mu} 1 dF(y), & \text{if } y < \mu \\ \tau \int_{\mu}^{\infty} 1 dF(y), & \text{if } y \geq \mu \end{cases} \\ &= \begin{cases} (\tau - 1) \cdot [F(y)]_{-\infty}^{\mu}, & \text{if } y < \mu \\ \tau \cdot [F(y)]_{\mu}^{\infty}, & \text{if } y \geq \mu \end{cases} \\ &= \begin{cases} (\tau - 1)F(\mu), & \text{if } y < \mu \\ \tau(1 - F(\mu)), & \text{if } y \geq \mu, \end{cases} \end{aligned} \tag{3.22}$$

where the third equality in (3.22) is obtained by using the formal definition of cumulative distribution function $F(\infty) = 1$ and $F(-\infty) = 0$. Then, by using the definition of stationary point by equating the first derivative (3.22) to zero we solve

$$0 = \begin{cases} (\tau - 1)F(\mu), & \text{if } y < \mu \\ \tau(1 - F(\mu)), & \text{if } y \geq \mu, \end{cases} \tag{3.23}$$

where we obtained for all $y \in \mathbb{R}$

$$\begin{aligned} F(\mu) &= \tau \\ \mu &= F^{-1}(\tau), \end{aligned} \tag{3.24}$$

which by Definition 3.0.1, we have shown that μ is the τ-th quantile of Y. Furthermore, to check that $\mu = F^{-1}(\tau)$ is the minimiser we can check the second derivative of (3.22)

$$\frac{\delta^2 \mathbb{E}[\rho_\tau(y - \mu)]}{\delta \mu^2} = \begin{cases} (\tau - 1)f(\mu), & \text{if } y < \mu \\ \tau(1 - f(\mu)), & \text{if } y \geq \mu, \end{cases} \tag{3.25}$$

where $f : \mathbb{R} \to [0, 1]$ be the probability density function of Y. So, we have for all $\mathbb{R}$ such that $\frac{\delta^2 \mathbb{E}[\rho_\tau(y-\mu)]}{\delta \mu^2} \in [0, 1] \geq 0$. Therefore, $\mu = F^{-1}(\tau)$ minimises the expected loss. □

Using Definition 3.4.1 of pinball loss function and Proposition 3.4.1, we can now define our non-parametric objective function for the best conditional quantile estimate in terms of linear problem of generalised additive model $\hat{\mu}_\tau(\mathbf{X})$.

Definition 3.4.2. (Non-Parametric Conditional Quantile Estimator (pp. 7, [Koe05]) Let $n \in \mathbb{Z}^+$ observations of response $y \in Y$ and covariates $\mathbf{x} \in \mathbf{X}$ and $\rho_\tau(\cdot)$ be the pinball loss function for $\tau \in (0, 1)$. We define the predictor coefficients $\boldsymbol{\beta} \in \mathbb{R}^p$ for the best conditional quantile estimate $Y|\mathbf{X}$ using non-parametric approach as follows

$$\hat{\boldsymbol{\beta}} = \underset{\boldsymbol{\beta} \in \mathbb{R}^p}{\arg\min} \sum_{i=1}^{n} \rho_\tau(y_i - \mu_\tau(\mathbf{x}_i|\boldsymbol{\beta})), \tag{3.26}$$

where for $m \times r = p$, we defined the quantile estimator $\mu_\tau(\mathbf{X}|\boldsymbol{\beta}) = \sum_{j=1}^{m} \sum_{k=1}^{r} \beta_{jk} h_{jk}(\mathbf{X})$ such that $h_{jk}(\mathbf{X})$ be fixed linear transformation of covariates $\mathbf{X}$.

Using the Definition 3.4.2, we will attempt to simulate non-parametric approach to estimate the 0.95-th conditional quantile estimate of `O3` in dataset `X2021`. However, unlike before we will solve the optimisation problem manually by implementing numerical optimisation method using `optim` built-in function in R. Specifically, in this project we apply Broyden-Fletcher-Goldfarb-Shanno (BFGS) algorithm to promote computational efficiency since it leverages quasi-Newton method for optimisation (Chen, 2023 [Che23]).

Definition 3.4.3 (Quasi-Newton Method)**.** Let $f : \mathbb{R}^p \to \mathbb{R}$ be the function to be minimised. Quasi-Newton method is a numerical optimisation method that leverages both Newton method and gradient descent to approximate the optimal solution $\mathbf{x}_K \approx \mathbf{x}^* \in \mathbb{R}^p$ such that $f(\mathbf{x}^*) < f(\mathbf{x})$ for all $\mathbf{x} \in \mathbb{R}^p$ after $K > 0$ number of iterations.

Besides, since we will train our model iteratively using `optim` function, we need to initialised the prior predictor coefficients $\boldsymbol{\beta}$ and the first derivative of the pinball loss function. Hence, in this case we used the cubic regression spline coefficients from previous simulation

as the initial values for $\boldsymbol{\beta}$ which has been declared previously as `fit$coefficients`. Also, by conditional derivative we define the first derivative of the pinball loss function as follows

$$\frac{\delta[\rho_\tau(y-\mu)]}{\delta\mu} = \begin{cases} (\tau - 1), & \text{if } y < \mu \\ \tau, & \text{if } y \geq \mu. \end{cases} \tag{3.27}$$

Same as before, we let $\tau = 0.95$ and `X` to represent 20 cubic spline transformation of observations `Temperature` while y represent the observations for `O3`.

```
tau <- 0.95
x <- data$x
y <- data$y
sm <- smoothCon(s(x, k = 20, bs = "cr"), data=data, knots=NULL)[[1]]
X <- sm$X
iBeta <- fit$coefficients
```

First, we define functions `loss_pinball` to represent pinball loss function.

```
#takes inputs quantile, basis transformed covariates, response, parameter
loss_pinball <- function(tau, X, y, par){
    #define the quantile estimate mu
    mu <- X %*% par
    #define z as difference between response and estimate
    z <- y - mu
    p <- rep(0, length(y))
    #compute pinball loss in terms of z
    p[z >= 0] <- tau * z[z >= 0]
    p[z < 0] <- (tau - 1) * z[z < 0]
    return(sum(p))
}
```

Then, we create functions `grad_pinball` to represent the first derivative of the pinball loss function and `negGrad_pinball` to compute the directional derivative to update the value $\mathbf{x}_{k+1}$.

```
#takes inputs quantile, basis transformed covariates, response, parameter
grad_pinball <- function(tau, X, y, par){
    #define the quantile estimate mu
    mu <- X %*% par
    #define z as difference between response and estimate
    z <- y - mu
    g <- rep(0, length(y))
    #compute derivative of pinball loss in terms of z
    g[z >= 0] <- tau
    g[z < 0] <- tau - 1
    return(g)
}

#takes inputs quantile, basis transformed covariates, response, parameter
negGrad_pinball <- function(tau, X, y, par){
    #find the directional derivative for each components of X
    tmp <- lapply(1:length(y),
    function(ii){
        #use negative sign gradient since we want to minimise the function
        a <- -grad_pinball(tau, X[ii,], y[ii], par)
        return( a * X[ii, ])})
    out <- Reduce("+", tmp)
    return(out)
}
```

Next, we use `optim` function to iteratively compute the optimal solution for the cubic spline predictor coefficients $\boldsymbol{\beta}$ for the 0.95-th conditional quantile estimate $\hat{\mu}_{0.95}$

```
fit_pinball <- optim(par = iBeta, loss_pinball, gr = negGrad_pinball,
                     method = "BFGS", tau = tau, X = X, y = data$y)
```

Finally, we plot both the non-parametric model and parametric model onto the dataset `X2021`.

```
plot(x, y, xlab = 'Temperature', ylab = 'O3', col = 'grey', ylim=c(-2, 150)
    )
lines(x[order(x)], X[order(x),] %*% fit_pinball$par, col = 'blue', lwd = 2)
lines(data$x[order(data$x)], condQuan[order(data$x)], col='red', lwd = 2)
legend(-5, 130, legend=c("Parametric model", "Non-parametric model"),
       col=c("red", "blue"), lty=1:1, cex=0.8)
```

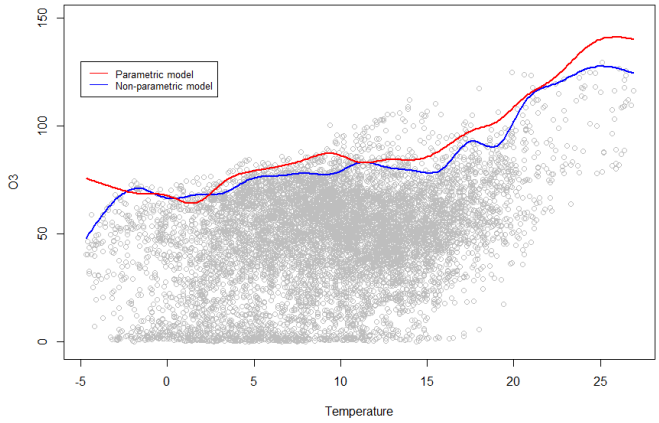


Figure 3.5: Non-parametric approach (blue) and parametric approach (red) to estimate the 0.9-th conditional quantile of `O3`.

Based on Figure 3.5, we can see how the non-parametric model differ to the previous parametric model. The most obvious difference is in the goodness-of-fit to which non-parametric possessed. Hence, this might indicates better performance in achieving our main objective of quantile estimation. In order to evaluate the quality of the non-parametric quantile regression model, we will simulate the hypothesis test from Subsection 3.3.1 in order to study the accuracy of our model estimation.

3.4.2 Assessing non-parametric quantile regression model estimation using hypothesis test

Once obtaining the fitted values for the conditional quantile estimates for `O3` like in Figure 3.5, we will investigate the performance of the non-parametric model using the 0.95 confidence level hypothesis test defined in (3.9) from Subsection 3.3.1. In this hypothesis test, we check the Wilson score interval for the statistic $\hat{p}_{mle}$ to deduce that $p = \mathbb{P}\{Y < \hat{\mu}_{0.95}(X)\} = 0.95$. This can be done using the same code from Subsection 3.3.

```
a <- 0.05
w <- rep(0, length(y))
for (i in 1:length(y)){
```

```
    if (y[i] <= X[i,] %*% fit_pinball$par){
        w[i] <- 1
    }
}

p_hat <- sum(w) / length(y)

z <- qnorm(1 - a, 0, 1)
r1 <- p_hat + z^2 / (2 * n)
r2 <- z * sqrt((p_hat * (1 - p_hat)) / n + z^2 / (4 * n^2))
r3 <- 1 + z^2 / n
CI <- c((r1 - r2) / r3, (r1 + r2) / r3)

print(paste("The confidence interval for p is (",CI[1],", ",CI[2],")"))
```

Output :

```
[1] "The confidence interval for p is ( 0.945885405202645 ,
    0.953817017291266 )"
```

By the above test, we obtained that $0.946 < p < 0.954$ which implies that there is no enough evidence to reject the null hypothesis that $p = \mathbb{P}\{Y < \hat{\mu}_{0.95}(X)\} = 0.95$. However, it is interesting to learn the accuracy of our conditional quantile estimate onto samples of data instead of the whole data like we are working on currently. This can be done by creating samples from the observations by partitioning them into bins. Therefore, we can simply run the hypothesis test onto each individual bin to check if the conditional quantile estimate $\hat{\mu}_{0.95}$ truly depicts the 0.95-th quantile of response `O3` (be reminded that in the modelling procedure we let y to represent `O3` while x represents `Temperature`).

To do such, we define function `create_bin` that takes arguments `data.frame` and the number of bins `k` to partitions our data into bins according to the range of values the univariate observation x falls into.

```
#takes input dataframe and number of bins to make
create_bin <- function(data, k){
    x <- data$x
    range <- seq(min(x), max(x))
    #set k number of percentiles to partition data
    percentile <- seq(0, 100, 100/k)
    #set the highest percentile to 100
    percentile[length(percentile)] <- 100
    #find k number of evenly spaced percentile using quantile
    breaks <- quantile(range, probs = percentile/100)
    #for the first percentile we minus the minimum observations by one
    #since cut consider open interval to define the lower bound of bin
    breaks[1] <- min(x) - 1
    #the last percentile be the maximum observations
    breaks[length(breaks)] <- max(x)
    #classify the observations into k bins using cut function
    bin <- cut(x, breaks = breaks)
    #append the bin ranges into the original dataframe
    data$bins <- bin
    return(data)
}
```

Then, we create function `quantile_bin` that takes arguments of `data.frame` and predictor coefficients `par` to compute the statistic or the estimate for $p = \mathbb{P}\{Y < \hat{\mu}_{0.95}(X)\}$ using observations in each bin by applying Maximum Likelihood Estimation from equation (3.10).

```
#takes input dataframe, basis function and predictor coefficient
quantile_bin <- function(data, sm, par){
    x <- data$x
    y <- data$y
    #initialised the bins for observations in dataframe data
    blocks <- sort(unique(data$bins))
    k <- length(blocks)
    #initialised k length of zero vector to store the estimate of p
    p <- rep(0, k)
    for(i in 1:length(blocks)){
        #collect observations from the same bin
        ranged <- data[data$bins==blocks[i],]
        sum_p <- 0
        for (j in 1:dim(ranged)[1]){
            #calculate the predicted conditional quantile of data in the
    bin
            pred <- PredictMat(sm, data.frame(x=ranged$x[j])) %*% par
            #if observation of response is smaller than predicted quantile
            if (ranged$y[j] < pred){
                #increment the count for sum_z
                sum_p <- sum_p + 1
            }
        }
        #define estimate of p as as sum_z divided by number of data in bin
        p[i] <- sum_p / dim(ranged)[1]
    }
    return (p)
}
```

Next, we create funtion `ci` that takes arguments `qb_estim` which is the statistics for all sample bins, `data.frame` and significance level of `a` which has default value of `a = 0.05`. Function `ci` returns the Wilson Score Interval for p at each bin of observations.

```
#takes input estimate of p, dataframe and significance level
ci <- function(qb_estim, data, a = 0.05){
    k <- length(qb_estim)
    #initialised the bins for observations in dataframe data
    blocks <- sort(unique(data$bins))
    #define the 1-a/2 quantile of standard normal distribution
    z <- qnorm(1 - a/2, 0, 1)
    #create dataframe to store the estimate of p, number of observations,
    #ranges of bins, lower bound interval, upper bound interval
    df <- data.frame(quantile_estimate = qb_estim, number = rep(0, k),
                     bins = blocks, ci_low = rep(0, k), ci_up = rep(0, k))
    for (i in 1:k){
        #compute the wilson score interval for every estimate of p
        p <- qb_estim[i]
        n <- dim(data[data$bins==blocks[i],])[1]
        r1 <- p + z^2 / (2 * n)
        r2 <- z * sqrt((p * (1 - p)) / n + z^2 / (4 * n^2))
        r3 <- 1 + z^2 / n
        low <- (r1 - r2) / r3
        up <- (r1 + r2) / r3

        #store the number of observations in bin
        df[i,]$number <- n
        #store the lower bound of wilson score
        df[i,]$ci_low <- low
        #store the upper bound of wilson score
        df[i,]$ci_up <- up
    }
    return (df)
}
```

After that, we create plotting function `ci_bin` that takes input in form of matrix that contains the confidence interval of p and the studied quantile $\tau \in (0, 1)$.

```
#takes input of wilson score interval and quantile
ci_bin <- function(ci_estim, tau){
    k <- dim(ci_estim)[1]
    range <- seq(1, k)
    #plot the estimate of p for each bins
    plot(range, ci_estim$quantile_estimate, ylim = c(min(ci_estim$ci_low),
        max(ci_estim$ci_up)), xlab = "Bins", ylab = "probability")
    abline(h = tau, col = "red")

    for (i in 1:k){
        #plot the wilson score interval for each bins
        arrows(range[i], ci_estim[i,]$ci_low, range[i], ci_estim[i,]$ci_up,
             angle = 90, code = 3, length = 0.1)
    }
    return (ci_estim)
}
```

Then, we redefine `data` to store bins' range of values using `create_bin` function and find the value for the statistics of $\hat{p}_{mle}$ for each bin using `quantile_bin` function. Then, using `ci` function, we store the estimated values for p at each bin in `data.frame` format along with their respective Wilson Score Interval at significance level of $\alpha = 0.05$.

```
#partition our data into 5 bins
data <- create_bin(data, 5)
#define the basis functions in terms of univariate x
x <- data$x
sm <- smoothCon(s(x, k = 20, bs = "cr"), data=data, knots=NULL)[[1]]
#calculate the statistics in each bin
qb_pinball <- quantile_bin(data, sm, fit_pinball$par)
#compute Wilson score interval for p
df_pinball <- ci(qb_pinball, data, 0.05)
#plot the results
ci_bin(df_pinball, tau)
```

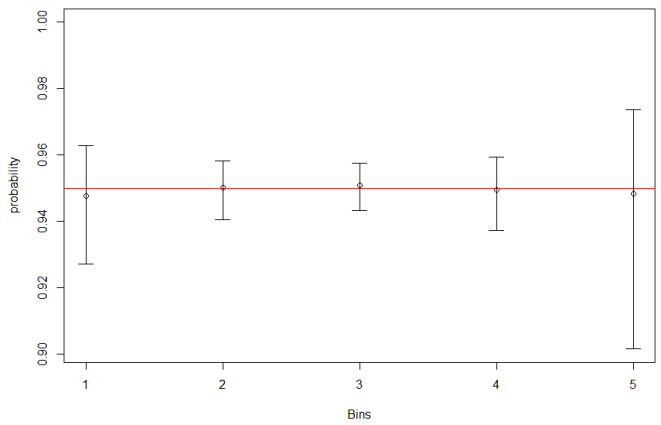


Figure 3.6: Plot for Wilson score interval for each estimate $\hat{p}_{mle}$ in 5 bin samples using non-parametric approach.

By observing Figure 3.6, we realise that for 5 number of bins, $\tau \in (CI_{i,lower}, CI_{i,upper})$ for all $i \in 1, ..., K$. Therefore, there is no significant evidence to reject the null hy-

pothesis which implies that $p = \mathbb{P}\{Y < \hat{\mu}_{0.95}(X)\} = \tau$. By that, we know that the non-parametric approach is effective in estimating the conditional quantile of `O3` using univariate `Temperature`. However, one question may arise before we decided to rely on non-parametric approach which is how well does the non-parametric approach performs on unseen data. To relieve ourselves from wondering about this matter, we will look into model validation procedure which involves two stages of modelling which is model training and model testing.

3.5 Model evaluation using train-test split data

In this subsection, we will explore one method of model validation which is by splitting our data into train and test samples. In particular, this method requires us to partition our data into two sets. In which we use one of the splitted dataset for training of our model to estimate the predictor coefficient $\boldsymbol{\beta} \in \mathbb{R}^p$ while the other dataset is used in order to test or assess the performance or the accuracy of the τ-th conditional quantile estimate $\mu_\tau(\mathbf{X}|\boldsymbol{\beta})$. Normally, we let the larger portion of the dataset to be used for model training whereas the lesser amount of dataset for model testing. In this project we fixed the ratio for the amount of observations in the training and testing dataset to be $70:30$ respectively. Taking the notion of this model validation process we will assess the performance of the non-parametric approach in modelling the 0.95-th quantile estimation for response `O3` given univariate `Temperature`.

Initially, we randomly defined a list `train` to store boolean arguments for data splitting using `sample` function in R. Ratio for the elements `TRUE:FALSE` is $70:30$.

```
train <- sample(1:dim(data)[1], size = 0.7 * length(x), replace = FALSE)
```

Using the list `train` we split the original dataset into two partitions. Besides, for the respective training and testing datasets we split into the respective observations for response $y \in Y$ and univariate $x \in X$.

```
data_train <- data[train,]
data_test <- data[-train,]

x_train <- data_train$x
y_train <- data_train$y

x_test <- data_test$x
y_test <- data_test$y
```

Then, by using `smoothCon` function we transform the training observations for the univariate $x \in X$ into cubic regression spline terms with basis $K = 20$.

```
sm <- smoothCon(s(x, k = 20, bs = "cr"), data=data_train, knots=NULL)[[1]]
X_train <- sm$X
```

Then, we use `optim` function for training of non-parametric model to estimate the predictor coefficients $\boldsymbol{\beta}$ using the training dataset. Like previous precedure, we fixed $\tau = 0.95$ and used `fit$coefficients` as the initial values for $\boldsymbol{\beta}$.

```
tau <- 0.95
iBeta <- fit$coefficients

```

```
train_pinball <- optim(par = iBeta, loss_pinball, gr = negGrad_pinball,
                       method = "BFGS", tau = tau, X = X_train, y = y_train)
```

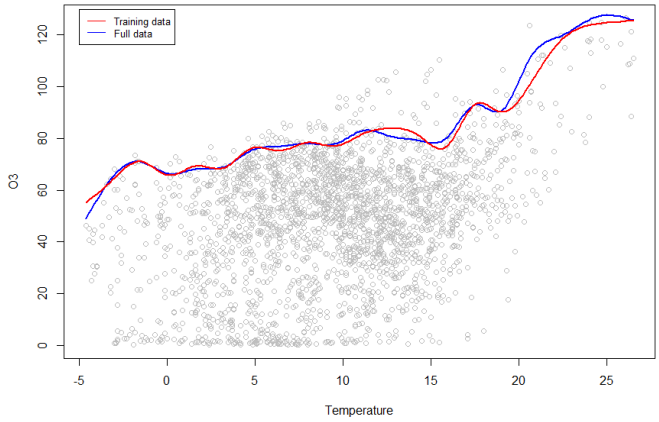


Figure 3.7: Non-parametric approach using full dataset (blue) and non-parametric approach using training dataset (red) to estimate the 0.95-th conditional quantile of `O3`.

Once obtaining the model estimator, we can test check the accuracy of non-parametric model `train_pinball` by computing the total pinball loss value between the quantile estimate fitted onto univariate `x_test` and the actual response in the test dataset `y_test`.

```
#to transform univariate in test set into cubic spline terms
X_test <- PredictMat(sm, data.frame(x=x_test))
#calculate the pinball loss between fitted values and actual response in
    test set
pinball_non <- loss_pinball(tau, X_test, y_test, train_pinball$par) /
    length(y_test)
#return the values
pinball_non
```

Output :

```
[1] 1.897811
```

Remark. Taking into consideration of our objective functions for non-parametric approach in Definition 3.4.2, we acknowledge the role of pinball loss in quantile regression model. In supervised machine learning process, the best selection of the predictor coefficients $\boldsymbol{\beta} \in \mathbb{R}^p$ are based on value that minimises the pinball loss. Therefore, it is reasonable to suggest pinball loss as the model performance metric to determine the best model for our quantile estimation problem.

To proceed with the assessment of non-parametric model performance, we implement the previous hypothesis test method from equation (3.9) in Subsection 3.3.1 onto the whole test set.

```
#set the number of bin samples to 1 since we want to check for whole test
    set
data_single_test <- create_bin(data_test, 1)
#quantile_bin function to compute the statistics
#ci function to compute the Wilson score interval
ci(quantile_bin(data_single_test, sm, train_pinball$par), data_single_test,
    0.05)
```

Output :

```
  quantile_estimate number        bins    ci_low     ci_up
1         0.9498164   2451 (-5.5,26.6] 0.9404474 0.9577776
```

Based on the above output, `quantile_estimate` is the statistic $\hat{p}_{mle}$, `number` is the number in the samples, `bins` is the range of our bin samples while `ci_low` and `ci_up` are both the respective lower and upper boundary of the Wilson score interval for $p = \mathbb{P}\{Y < \hat{\mu}_{0.95}(X_{test})\}$. By the above results, we have $\tau \in 0.95 \in [CI_{lower}, CI_{upper}]$ thus retaining the null hypothesis. Moving from this, we will challenge the performance of non-parametric model by running the hypothesis test onto 5 bin samples from the testing dataset. To do so, we defined 5 bins from the testing dataset using `create_bin` function. Then, for each bins we calculate the statistics $\hat{p}_{mle}$ for $p = \mathbb{P}\{Y < \hat{\mu}_{0.95}(X_{test})\}$ using `quantile_bin` function.

```
data_test <- create_bin(data_test, 5)
qbt_pinball <- quantile_bin(data_test, train_pinball$par)
```

Next, we compute the 0.95 Wilson score interval for p in each bin using `ci` function and plot the confidence interval graph using `ci_bin` function.

```
dft_pinball <- ci(qbt_pinball, data_test, 0.05)
ci_bin(dft_pinball, tau)
```

$\hat{p}_{mle}$	Number of samples	Bins	0.95 Wilson score interval
0.97660	171	$(-5.7, 1.5]$	$[0.94140, 0.99086]$
0.94312	756	$(1.5, 7.7]$	$[0.92426, 0.95750]$
0.94100	1051	$(7.7, 13.9]$	$[0.92509, 0.95371]$
0.94418	430	$(13.9, 20.1]$	$[0.91829, 0.96220]$
0.86046	43	$(20.1, 26.9]$	$[0.72736, 0.93444]$

Table 3.1: Table of statistic $\hat{p}_{mle}$, number of samples in each bin, range of values in each bin and 0.95 Wilson score interval for p.

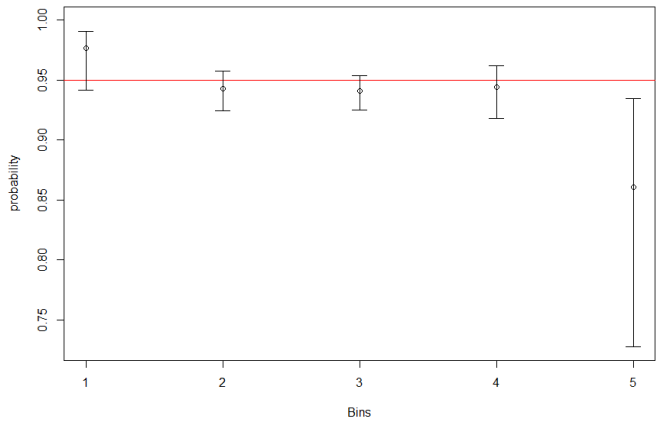


Figure 3.8: Plot for Wilson score interval for $p = \mathbb{P}\{Y < \hat{\mu}_{0.95}(X)\}$ in 5 bin samples using non-parametric approach onto testing dataset.

Based on the results from Table 3.1 and Figure 3.8, we observed how the conditional quantile estimate $\hat{\mu}_{0.95}(X)$ has such an obvious drop in performance when fitted on unseen data.

Although, non-parametric model works very well on trained data, it lacks the capability to estimate the conditional quantile of unseen data. Thus, results in higher uncertainty towards model's actual performance when we only have limited data. This may due to the overfitting tendency that increases the wiggliness of the model in non-parametric approach. Hence, reduces the accuracy of the quantile estimation. Therefore, in the next subsection, we will look into the alternative semi-parametric approach that has better control on the wiggliness of the conditional quantile estimation model.

3.6 Semi-parametric approach to estimate conditional quantile

Acquiring the result from non-parametric method to quantile estimation, we are confident that it will help us in estimating the conditional quantile of our response without making strong distribution assumptions. However, we should also be concerned of the feasibility of the solution. This is because based on Definition 3.4.1, we can see that the pinball loss function is a piecewise linear function with minimum of value zero at the point $y = \mu_\tau$. In addition to that, pinball loss function is asymmetry since it consists of two distinct slopes. This characteristic of pinball loss function influences the variability of our model estimator. This effect can be seen in Figure 3.9.

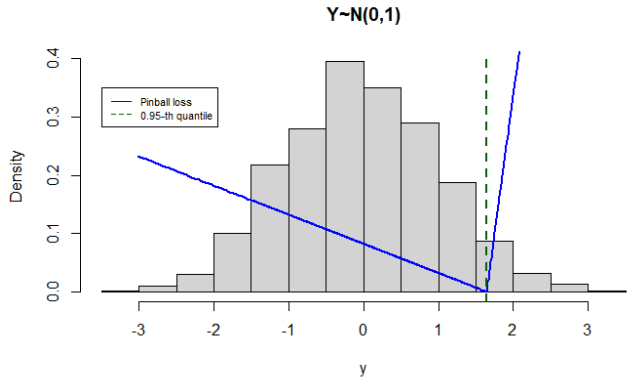


Figure 3.9: Pinball loss asymmetric effect on 1000 random observations of standard normal distribution (blue) with quantile $\mu_{0.95}$ (green).

From Figure 3.9, we observed how the values of pinball loss are heavily dominated by observations $z > \mu_{0.95}$ which are lesser in quantity compared to observations $z \geq \mu_{0.95}$. In modelling context, this can influence the variability of our quantile estimate especially when we are dealing with extreme quantile (e.g for $\tau \approx 0$ and $\tau \approx 1$). Moreover, this can further produce poor quantile estimator due to likeliness of having outliers in our observations. In an effort to reduce the variability of our model due to extreme values, we need to modify the pinball loss such that it asymmetric and piecewise behaviour can be compromised by introducing some weight parameters and transforming it into a continuous function. Hence, we are in need of a smooth form of pinball loss function which can regulate the goodness-of-fit of our model function. The idea to this has been proposed by (Fasiolo *et al.* 2017 [Fas+17]) where they introduced the extended log-F (ELF) density function to treat the input of the pinball loss as a random variable from the exponential family from Definition 2.1.3.

3.6.1 Construction of smooth form of pinball loss function

Definition 3.6.1 (Extended Log-F (ELF) density)**.** Given the observation of random variable $y \in Y$ and for some $\mu \in \mathbb{R}$. Let $z = y - \mu$ and $\tau \in (0, 1)$, the ELF density function is defined as follows

$$\tilde{\rho}_F(z|\tau, \lambda, \sigma) = \frac{e^{(1-\tau)\frac{z}{\sigma}} \left(1 + e^{\frac{z}{\lambda\sigma}}\right)^{-\lambda}}{\lambda\sigma \text{Beta}[\lambda(1-\tau), \lambda\tau]}, \tag{3.28}$$

where $\lambda > 0$ and $\sigma \in \mathbb{R}$ are predetermined tuning parameters while $\text{Beta}(\cdot, \cdot)$ is the beta function.

In the general case of leveraging ELF density function in modelling, we usually express ELF density function in terms of its logarithm. This is due to the computational complexity of exponential values that may be too small and sensitive to approximation error. By considering the bijective and strictly increasing properties of logarithm we simplify the ELF density function as the negative log ELF density function denoted by $\rho^*_\tau(z|\lambda, \sigma)$

$$\begin{aligned}
\rho^*_\tau(z|\lambda, \sigma) &= -\log\{\tilde{\rho}_F(z|\tau, \lambda, \sigma)\} \\
&= -\log\left\{\frac{e^{(1-\tau)\frac{z}{\sigma}} \left(1 + e^{\frac{z}{\lambda\sigma}}\right)^{-\lambda}}{\lambda\sigma \text{Beta}[\lambda(1-\tau), \lambda\tau]}\right\} \\
&= (\tau - 1)\frac{z}{\sigma} + \lambda \log(1 + e^{\frac{z}{\lambda\sigma}}) + \log\{\lambda\sigma \text{Beta}[\lambda(1-\tau), \lambda\tau]\}.
\end{aligned} \tag{3.29}$$

Using the log ELF density function as our generalisation of pinball loss function, we are able to reduce the variability of our quantile estimator. This is because the log ELF density function reduces the asymmetric effect in pinball loss function due to choices of extreme quantile and domination of large values by extreme observations or outliers. This can be seen in Figure 3.10 that shows the plot for negative log ELF density function onto 1000 observations from standard normal distribution.

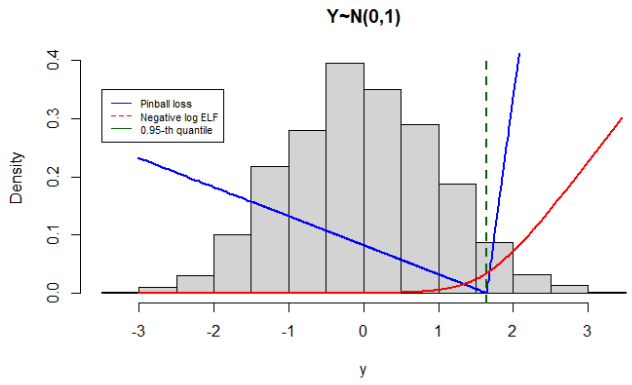


Figure 3.10: Pinball loss (blue) and Negative log ELF density function (red) on 1000 random observations of standard normal distribution with quantile $\mu_{0.95}$ (green).

From the plot in Figure 3.10, one realises the potential of using negative log ELF density function $\rho^*_\tau(\cdot)$ in managing the variability for our quantile estimation. However, there are questions regarding the implementation of it and the rules in deciding the optimal hyperparameters like λ and σ to provide the best conditional quantile estimators. On the

next subsection, we will focus on the construction of the model objective function using negative log ELF density functions and the notion of semi-parametric approach in quantile estimation which involves parametric assumption on the computation of one of the weight parameter λ.

3.6.2 Weak parametric assumption in determining model weight parameter

In the negative log ELF density function $\rho_\tau^*(\cdot)$ there are inclusions of additional hyperparameters apart from the predictor coefficients $\boldsymbol{\beta}$ which are σ and λ. This hyperparameters are tuned to reduce the wiggliness of our quantile estimator. Hence, choosing the most suitable values for σ and λ will help us to optimise the bias-variance tradeoff for our model. In this project due simplicity of modelling, we set $\sigma = 1$. Whereas the weight parameter $\lambda \in \mathbb{R}$ can be defined as a deterministic value that depends on our assumption on the probability distribution of the responses per suggested by [Fas+17]. The explicit form in deriving the deterministic value for the weight parameter λ is described in [CS06] which in the context of this project be defined as follows.

Definition 3.6.2 (Deterministic weight, λ (pp. 4-5 [CS06]))**.** Let $f(\cdot)$ be the assumed probability density function on the response variable. Then, the optimal weight $\lambda = \hat{\lambda}^*$ is defined as

$$\hat{\lambda}^* = \frac{1}{n}\left\{\frac{9f(\mu_\tau|\hat{\boldsymbol{\theta}}_{mle})}{\pi^4 f'(\mu_\tau|\hat{\boldsymbol{\theta}}_{mle})^2}\right\}^{\frac{1}{3}}, \tag{3.30}$$

where $\hat{\boldsymbol{\theta}}_{mle}$ be the parameters of the assumed probability distribution estimated by Maximum Likelihood Estimation.

Remark. The explicit form of deterministic weight $\hat{\lambda}^*$ in Definition 3.6.2 to be implemented in (3.29) is inspired by the results from *Corollary 3.1* from Shankar (1998) [Sha98] under the assumptions stated in *Chapter 1* of the paper that we considered holds for the assumed probability density function $f(\cdot|\boldsymbol{\theta})$. This assumptions require **(1)** $f(\cdot|\boldsymbol{\theta})$ to be differentiable and has bounded derivative while **(2)** $f'(\cdot|\boldsymbol{\theta})$ must be continuous in the close proximity of μ_τ and $f'(\mu_\tau|\boldsymbol{\theta}) \neq 0$.

Applying Definition 3.6.2, we can reduce the effort of training our model multiple times in order to choose the optimal weight parameter, λ by assuming the distribution of the response. In which, in this scenario we agreed to make a minor parametric assumption specifically on the tuning parameter instead of the conditional quantile estimator like in parametric approach. In order to efficiently derive $\hat{\lambda}^*$, we can follow the computation steps in Algorithm 1.

Algorithm 1: Computational method of determining the weight parameter $\hat{\lambda}^*$.

Input: $n > 0$ number of observations for response $\{y_1, ..., y_n\} \in Y$, $\tau \in (0, 1)$ and assumed distribution with probability density function $f(\cdot|\boldsymbol{\theta})$ and cumulative distribution function $F(\cdot|\boldsymbol{\theta})$.

Output: Deterministic weight parameter $\hat{\lambda}^*$.

1. Initialise some values for $\boldsymbol{\theta} = \boldsymbol{\theta}_0$.
2. Using quasi-Newton method compute $\hat{\boldsymbol{\theta}}_{mle} = \arg\min_{\boldsymbol{\theta}} \sum_{i=1}^{n} -\log\{f(y_i|\boldsymbol{\theta})\}$ using initial values $\boldsymbol{\theta}_0$.
3. Calculate $\mu_\tau = F^{-1}(\tau|\hat{\boldsymbol{\theta}}_{mle})$.
4. Calculate $f(\mu_\tau|\hat{\boldsymbol{\theta}}_{mle})$ and its derivative $f'(\mu_\tau|\hat{\boldsymbol{\theta}}_{mle})$.
5. Compute $\hat{\lambda}^* = \frac{1}{n}\left\{\frac{9f(\mu_\tau|\hat{\boldsymbol{\theta}}_{mle})}{\pi^4 f'(\mu_\tau|\hat{\boldsymbol{\theta}}_{mle})^2}\right\}^{\frac{1}{3}}$.

In practical scenario of using R, we can use `dnorm` to calculate the probability density function $f(\cdot)$ and `qnorm` to calculate the quantile $\mu_\tau = F^{-1}(\cdot)$. Whereas the Maximum Likelihood Estimation using quasi-Newton method in the second line of Algorithm 1 can be done using `optim` function. Equipped with the newly defined weight parameter $\hat{\lambda}^*$ we can now defined the model objective function for semi-parametric quantile regression model.

3.6.3 Construction of semi-parametric quantile regression model

By fixing $\sigma = 1$ and using Algorithm 1 in defining $\hat{\lambda}^*$, we can reduce the ELF density function from Definition 3.6.1 into a single term function which can be substituted by the difference of response variable with estimated quantile, i.e. $z = y - \mu_\tau$. In which by taking the application of generalised additive model we can expressed the estimated quantile as a linear transformation of smooth functions with input covariates like in Subsection 3.4 for non-parametric approach. Taking this notion into consideration we can redefine our non-parametric model objective function from Definition 3.4.2 to be as follows.

Definition 3.6.3 (Semi-Parametric Penalised Conditional Quantile Estimator)**.** Let $n \in \mathbb{Z}^+$ observations of response $y \in Y$ and covariates $\mathbf{x} \in \mathbf{X}$ with $\tilde{\rho}_F(\cdot)$ be the ELF density function. let $\hat{\lambda}^*$ be determined as in definition 3.6.2, the predictor coefficients $\boldsymbol{\beta} \in \mathbb{R}^p$ for the optimal τ-th conditional quantile estimate using semi-parametric approach is as follows

$$\hat{\boldsymbol{\beta}}_\gamma = \underset{\boldsymbol{\beta}\in\mathbb{R}^p}{\arg\min} \sum_{i=1}^{n} -\log\{\tilde{\rho}_F(y_i - \mu_\tau(\mathbf{x}_i|\boldsymbol{\beta})|\tau, \hat{\lambda}^*, \sigma)\} + \gamma\boldsymbol{\beta}^\intercal \boldsymbol{S}\boldsymbol{\beta}, \tag{3.31}$$

for some positive semidefinite matrix $\boldsymbol{S} \in \mathbb{R}^{r\times r}$ and a non-negative value $\gamma \in \mathbb{R}$. Whereas for $m \times r = p$, we defined the conditional quantile estimator $\mu_\tau(\mathbf{X}|\boldsymbol{\beta}) = \sum_{j=1}^{m}\sum_{k=1}^{r} \beta_{jk} h_{jk}(\mathbf{X})$ such that $h_{jk}(\mathbf{X})$ be fixed linear transformation of covariates $\mathbf{X}$.

In addition to the use of negative log ELF density function to regulate the variability of pinball loss, Definition 3.6.3 also consider the penalised version of generalised additive model from Definition 2.1.1 to penalise the goodness-of-fit of the semi-parametric model. Hence, we can better manage the wiggliness of our model by specifying the tuning hyper-parameter, $\gamma > 0$. Now, we will begin simulating semi-parametric approach to estimate

the conditional quantile for `O3` using penalised cubic regression spline model.

Firstly, we import libraries for model training and basis transformation.

```
library(qgam)
library(mgcv)
```

Then, we create function `dlf` to compute the log ELF density function with respect to observation of response Y and quantile estimate μ_τ.

```
#computes log ELF density and its derivatives w.r.t mu
dlf <- function(y, tau, mu, lam, log = FALSE, deriv = 0)
{
    #calculate the log ELF density function
    sig <- 1 #fixed sigma to 1
    z <- (y - mu) / sig
    out <- (1-tau) * z - lam * log1pexp( z / lam ) -
            log( sig * lam * beta(lam*(1-tau), tau*lam) )
    #if not log return the ELF density function
    if( !log ) out <- exp(out)
    #if deriv is positive compute the derivative of the log ELF density
    if( deriv > 0 )
    {
        out <- list("d" = out)
        dl <- dlogis(y, mu, lam*sig)
        pl <- plogis(y, mu, lam*sig)
        out$D <- sum((pl - (1-tau)) / sig)
        #if deriv is more than one compute the second derivative
        if(deriv > 1)
        {
            out$D2 <- sum( - dl / sig )
        }
    }
    return(out)
}
```

Next, we create function `penalty` that takes input penalised constant γ, predictor coefficients $\boldsymbol{\beta}$ and positive semidefinite matrix $\boldsymbol{S}$ to return the penalised component $\gamma\boldsymbol{\beta}^\intercal\boldsymbol{S}\boldsymbol{\beta}$ and its derivative.

```
#takes input hyperparameter, predictor coefficient, positive semidefinite
penalty <- function(k, par, S, deriv = 0)
{
    #compute the penalty component in the generalised pinball loss
    w <- k * t(par) %*% S %*% par
    w <- list("w" = w)
    #if deriv is positive compute the derivative of the penalty component
    if (deriv > 0)
    {
        w$D <- 2 * k * t(par) %*% S
    }
    return(w)
}
```

After that, we create `penalised_negllkFun` function to compute the model objective function defined in Definition 3.6.3 and `penalised_negGrad` function to compute the directional derivative of the model objective function.

```
#takes input predictor coefficient, quantile, weight, covariates,
#response observation, semi positive definite matrix and hyperparameter
penalised_negllkFun <- function(par, tau, lam, X, y, S, k = 0)
```

```
{
    #compute the conditional quantile estimate
    mu <- X %*% par
    #compute the semi-parametric objective function
    out <- - sum( dlf(y = y, tau = tau, mu = mu, lam = lam, log = TRUE))
            + penalty(k, par, S)$w
    return(out)
}

#takes input predictor coefficient, quantile, weight, covariates,
#response observation, semi positive definite matrix and hyperparameter
penalised_negGrad <- function(par, tau, lam, X, y, S, k = 0)
{
    #compute the conditional quantile estimate
    mu <- X %*% par
    #compute the descending gradient of semi-parametric objective function
    tmp <- lapply(1:length(mu),
                function(ii){
                    a <- - dlf(y = y[ii], tau = tau,
                    mu = mu[ii], lam = lam, log = TRUE, deriv = 1)$D

                    return( a * X[ii, ] + penalty(k, par, S, deriv = 1)$D )
    })
    out <- Reduce("+", tmp)
    return(out)
}
```

Now, we have our main model defining function to work on the optimal solutions for objective function in Definition 3.6.3. Using the `optim` function, we can find the solution for conditional quantile estimator $\mu_\tau(\mathbf{X}|\hat{\boldsymbol{\beta}}_\gamma)$ if we can decide on the best penalising hyperparameter $\gamma > 0$. One may wonder how can we decide on the most suitable tuning hyperparameter. One of the strategy suggested by Brownlee (2023) in doing so is by undergoing K-fold cross validation to test the performance of model with different values of hyperparameter onto different sets of unseen data [Bro23].

3.6.4 Implementation of K-fold cross validation in determining tuning hyperparameter for semi-parametric quantile regression model

Cross validation involves comparisons of statistical analysis on models trained using sets of different tuning hyperparameter values. K-fold cross validation requires the sampling of observations into $K > 0$ number of partitions in order to run the cross validation process for K multiple times for each choice of tuning hyperparameter (James et al., 2023[Jam+23]). Taking the inspiration from penalised least square problem, we construct the K-fold cross validation for the model objective function in Definition 3.6.3 as follows.

Definition 3.6.4 (K-Fold Cross Validation for Penalised Semi-Parametric Method)**.** Consider the case for $n > 0$ observations of response $y \in Y$ and univariate $x \in X$. Let $K > 0$ and $P_s \in \{1, ..., n\}$ such that $\bigcup_{s=1}^{K} P_s = \{1, ..., n\}$, the K-fold cross validation for model objective function in definition 3.6.3 is defined as

$$CV(K;\gamma) = \frac{1}{K}\sum_{s=1}^{K}\left\{\frac{1}{n_s}\sum_{i\in P_s}\rho_\tau^*\left(y_i - \hat{\mu}_\tau^{-P_s}(x_i|\boldsymbol{\beta},\gamma)|\hat{\lambda}^*\right)\right\}, \tag{3.32}$$

where $n_s = |P_s|$ and $\rho_\tau^*(\cdot) = -\log\{\tilde{\rho}_F(\cdot)\}$ be the negative log ELF density function. Hence,

considering set of $L \in \mathbb{Z}^+$ candidates for $\gamma \in (0, 1]$, we choose $\gamma = \gamma^*$ that minimises (3.32),

$$\gamma^* = \underset{\gamma_l \in (0,1], l \in \{1,..,L\}}{\arg\min} \; CV(K; \gamma_l). \tag{3.33}$$

Applying the K-fold cross validation process from Definition 3.6.4, we design the algorithm to evaluate and choose the optimal penalising constant $\gamma > 0$ that minimises the cross validation score $CV(\cdot)$.

Algorithm 2: K-Fold cross validation to choose $\gamma > 0$ in penalised semi-parametric method of quantile estimation model.

Input: Non-negative integer K, Non-negative hyperparameter candidates $\mathrm{A} = \{\gamma_1, ..., \gamma_L\}$, $\tau \in (0, 1)$, and $n > 0$ number of observations for response $\{y_1, ..., y_n\}$ and univariate $\{x_1, ..., x_n\}$.

Output: The optimal penalising hyperparameter $\gamma = \gamma^*$.

Initialised an $l > 0$ sized zero vector to store cross validation score $\mathrm{CV} = \{0, ..., 0\}$;
for $l \in \{1, ..., L\}$ **do**
  for $s \in \{1, ..., K\}$ **do**
    $\hat{\lambda}^* \leftarrow$ Using Algorithm 1 with input $\{y_i : i \notin P_s\}$, τ and assumed probability density function $f(\cdot|\boldsymbol{\theta})$ and cumulative distribution function $F(\cdot|\boldsymbol{\theta})$. $\hat{\boldsymbol{\beta}}_{\gamma_l} \leftarrow \arg\min_{\boldsymbol{\beta}} \sum_{i \notin P_s} \rho_\tau^* \left(y_i - \mu_\tau(x_i|\boldsymbol{\beta}) | \hat{\lambda}^*, \gamma_l \right) + \gamma_l \boldsymbol{\beta}^\intercal \boldsymbol{S} \boldsymbol{\beta}$;
    for $i \in P_s$ **do**
      $n_s \leftarrow |P_s|$
      $\hat{\mu}_\tau^{-P_s}(x_i|\hat{\boldsymbol{\beta}}_{\gamma_l}) \leftarrow \sum_{j=1}^{p} \hat{\beta}_{\gamma_l, j} h_j(x_i)$;
      $\mathrm{CV}[l] \leftarrow \mathrm{CV}[l] + \frac{1}{n_s} \rho_\tau^* \left(y_i - \hat{\mu}_\tau^{-P_s}(x_i|\hat{\boldsymbol{\beta}}_{\gamma_l}) | \hat{\lambda}^* \right)$;
$\gamma^* \leftarrow \mathrm{A}[\mathrm{min.index}(\mathrm{CV})]$;
return γ^*;

Leveraging both methods from Algorithm 1 and Algorithm 2, we can find the optimal penalising hyperparameter γ^* for different distribution assumptions on $\hat{\lambda}^*$. In this project, we will focus on two cases which are the normal distribution and gamma distribution assumptions. The result of the K-fold cross validation for K = 10 is documented in Figure 3.11 and Table 3.2.

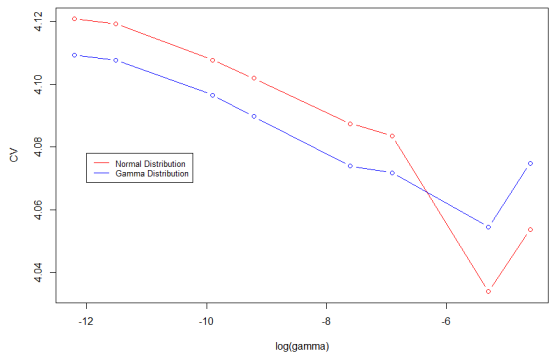


Figure 3.11: Plot of cross validation score against $\log(\gamma)$ for Gamma distribution assumption (red) and Normal distribution assumption (blue) for weight parameter $\hat{\lambda}^*$.

	$CV(K=5;\gamma)$	
γ	Normal	Gamma
5.0×10^{-6}	4.1208	4.1093
1.0×10^{-5}	4.1192	4.1076
5.0×10^{-5}	4.1077	4.0965
1.0×10^{-4}	4.1019	4.0897
5.0×10^{-4}	4.0874	4.0739
1.0×10^{-3}	4.0835	4.0717
5.0×10^{-3}	4.0338	4.0545
1.0×10^{-2}	4.0535	4.0747

Table 3.2: Table of hyperparameter candidates γ and cross validation score for K = 5 with gamma distribution and normal distribution assumption for weight parameter $\hat{\lambda}^*$.

Based on the above results, we choose the penalising hyperparameter to be $\gamma^* = 5.0 \times 10^{-3}$ for both cases of distribution assumptions for the weight parameter λ. In the case for gamma distribution assumption, we compute the weight parameter to be $\hat{\lambda}^*_{gamma} = 0.9802916$ while for normal distribution assumption, we compute the weight parameter to be $\hat{\lambda}^*_{normal} = 0.8005079$. In which, using the optimal value of γ^* and both the deterministic weights, we can now model the 0.95-th conditional quantile estimates for O3 using semi-parametric approach.

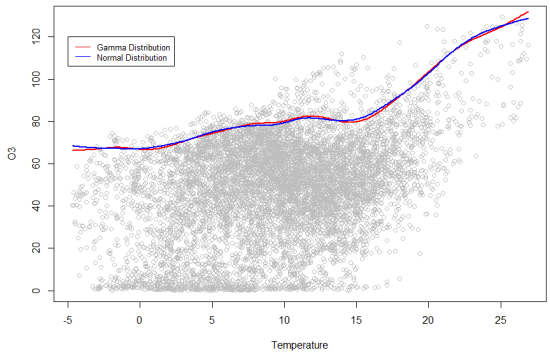


Figure 3.12: Semi-parametric model using tuning hyperparameter $\gamma^* = 5.0 \times 10^{-3}$ for gamma distribution assumption (red) and normal distribution assumption (blue).

Based on Figure 3.12, we can observe a slight difference in the fitted model of semi-parametric quantile regression due to different distribution assumptions. Next, we will study the performance of the quantile estimation for semi-parametric approach besides comparing the accuracy between different distribution assumption onto the weight parameter λ. Like in previous cases, we will perform hypothesis test constructed in Subsection 3.3.1 in order to detemine whether the results from semi-parametric method are valid for quantile estimation.

3.6.5 Assessing semi-parametric quantile regression model estimation using hypothesis test

Firstly, we will look into implementing the 0.95 confidence level hypothesis test defined in Subsection 3.3.1 onto the whole dataset `data`. In order to do so, we used initialised a single bin onto our data using `create_bin` function.

```
data_single <- create_bin(data, 1)
```

Then, we used `quantile_bin` function to estimate the values for the statistic $\hat{p}_{mle}$ and using `ci` to compute the 0.95 Wilson score interval for $p = \mathbb{P}\{Y < \hat{\mu}_{0.95}(X)\}$. Since we consider for two distribution assumptions, we need to perform the hypothesis test on both the gamma distribution assumption and normal distribution assumption semi-parametric models.

```
#best_gamma$par is the semi-parametric model coefficients for gamma
    assumption
ci(quantile_bin(data_single, best_gamma$par), data_single, 0.05)
#best_normal$par is the semi-parametric model coefficients for normal
    assumption
ci(quantile_bin(data_single, best_normal$par), data_single, 0.05)
```

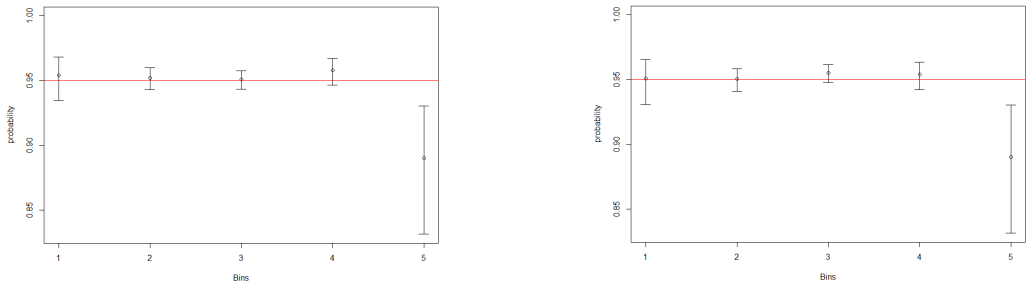
Distribution assumption	$\hat{\lambda}$	$\hat{p}_{mle}$	0.95 Wilson score interval
Normal	0.80050	0.95176	$[0.94690, 0.95620]$
Gamma	0.98029	0.95201	$[0.94716, 0.95644]$

Table 3.3: Table of distribution assumptions, weight $\hat{\lambda}$, statistic $\hat{p}_{mle}$, and 0.95 Wilson score interval for $\hat{p}_{mle}$.

From the above output, the results shows that $0.95 \in [CI_{lower}, CI_{upper}]$ for both cases implying there is no clear evidence to reject null hypothesis h_0 which states that $p = \mathbb{P}\{Y < \hat{\mu}_{0.95}(X)\} = 0.95$. This shows that semi-parametric model are reliable in estimating the 0.95-th quantile of `O3` with respect to `Temperature` when we considered the overall estimation onto the whole dataset. Moving from this, we will run the hypothesis test onto samples partitioned from the dataset. Like in Subsection 3.4.2, we leverages bin sampling to partitioned our dataset into 5 samples. The results from the hypothesis test are recorded below in Table 3.4 and Figure 3.13.

Bins	number of samples	0.95 Wilson score interval	
		$\hat{\lambda}^*_{normal} = 0.80050$	$\hat{\lambda}^*_{gamma} = 0.98029$
$(-5.7, 1.5]$	184	$[0.93467, 0.96815]$	$[0.93088, 0.96545]$
$(1.5, 7.7]$	745	$[0.94284, 0.96002]$	$[0.94102, 0.95849]$
$(7.7, 13.9]$	1051	$[0.94342, 0.95775]$	$[0.94771, 0.96147]$
$(13.9, 20.1]$	437	$[0.94669, 0.96695]$	$[0.94233, 0.96346]$
$(20.1, 26.9]$	34	$[0.83137, 0.93038]$	$[0.83137, 0.93038]$

Table 3.4: Table of range values in each bins, number of samples in each bins and 0.95 Wilson score interval of p for semi-parametric model with normal distribution assumption on $\hat{\lambda}^*_{normal}$ and gamma distribution assumption on $\hat{\lambda}^*_{gamma}$.



(a) $\hat{\lambda}^*_{normal} = 0.8005079$ (b) $\hat{\lambda}^*_{gamma} = 0.9802916$

Figure 3.13: Wilson score interval for the estimation of $p = \mathbb{P}\{Y < \hat{\mu}_{0.95}(X)\}$.

Based on the above results, we realised that the semi-parametric model fit very well in estimating 0.95-th quantile for the first four bins of the dataset. But the results show that in both cases of distribution assumption, semi-parametric model lack the capability to accurately estimate the 0.95-th quantile of `O3` for the final bin sample. This may be due to the significance drop in the number of samples in the bin relative to its predecessors. However, by the results of the hypothesis test in Table 3.3 we know that in consideration for overall performance for the whole dataset, this method is still relevant in estimating the 0.95-th conditional quantile for `O3` with respect to `Temperature`. Next, we will look into how semi-parametric approach in quantile regression performs on unseen data.

3.6.6 Train-test split to assess the performance of semi-parametric model

In this subsection we will implement train-test split of our observations in order to analyse the performance of the respective semi-parametric models on unseen data. Like in Subsection 3.5, we split our observations into ratio of 70 : 30, where 70 percent of the observations will be used for training of semi-parametric model and the other 30 percents of the observations will be used for performance analysis. As stated before we will model two semi-parametric model using two different distribution assumptions onto the weight parameter $\hat{\lambda}^*$ with the same tuning hyperparameter of $\gamma^* = 5.0 \times 10^{-3}$. The results for the respective models are as follows.

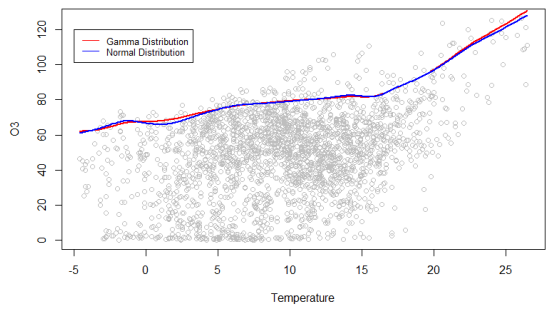


Figure 3.14: Semi-parametric model trained by training dataset using tuning hyperparameter $\gamma^* = 5.0 \times 10^{-3}$ for gamma distribution assumption (red) and normal distribution assumption (blue) fitted onto testing dataset.

Once obtaining the semi-parametric model for both distribution assumptions trained using the training dataset, we fitted the model onto testing dataset to attain the fitted 0.95-th quantile estimates $\hat{\mu}_{0.95}$ for `O3`. Using $\hat{\mu}_{0.95}$, we can now run the hypothesis test to study the performance of semi-parametric model on unseen data. Just to be reminded, we are testing for null hypothesis, $H_0 : p = \mathbb{P}\{Y < \hat{\mu}_{0.95}(X_{test})\} = 0.95$ against $H_1 : p \neq 0.95$.

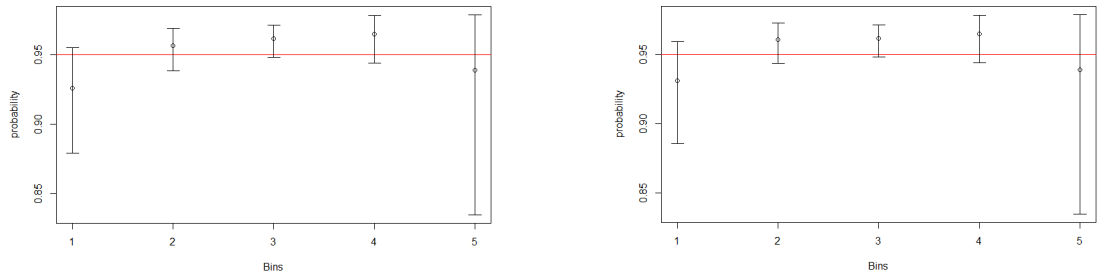
Distribution assumption	$\hat{\lambda}$	$\hat{p}_{mle}$	0.95 Wilson score interval
Normal	0.90410	0.95756	$[0.94884, 0.96485]$
Gamma	1.1016	0.95920	$[0.95062, 0.96634]$

Table 3.5: Table of distribution assumptions, weight $\hat{\lambda}$, statistic $\hat{p}_{mle}$, and 0.95 Wilson score interval for $\hat{p}_{mle}$.

Table 3.5 shows that for both distribution assumptions, we have $\tau = 0.95 \in [CI_{lower}, CI_{upper}]$ implying that we retains the null hypothesis. This shows that semi-parametric model has good estimation for 0.95-th quantile of `O3` with respect to `Temperature` on unseen data. Next, we will run the hypothesis testing onto 5 bin samples from the testing dataset to observe in details on how semi-parametric model performs on unseen data at different ranges of values. Below are the results containing the statistics $\hat{p}_{mle}$, and the respective 0.95 Wilson score interval for $p = \mathbb{P}\{Y < \hat{\mu}_{0.95}(X_{test})\}$

Bins	Number of samples	0.95 Wilson score interval	
		$\hat{\lambda}^*_{normal} = 0.90410$	$\hat{\lambda}^*_{gamma} = 1.1016$
$(-5.7, 1.5]$	189	$[0.87951, 0.95536]$	$[0.88588, 0.95936]$
$(1.5, 7.7]$	688	$[0.93843, 0.96928]$	$[0.94350, 0.97289]$
$(7.7, 13.9]$	1066	$[0.94823, 0.97152]$	$[0.94823, 0.97152]$
$(13.9, 20.1]$	459	$[0.94413, 0.97843]$	$[0.94413, 0.97843]$
$(20.1, 26.9]$	49	$[0.83479, 0.97895]$	$[0.83479, 0.97895]$

Table 3.6: Table of range of values in each bins, number of samples in each bins and 0.95 Wilson score interval of p for semi-parametric model with normal distribution assumption on $\hat{\lambda}^*_{normal}$ and gamma distribution assumption on $\hat{\lambda}^*_{gamma}$.



(a) $\hat{\lambda}^*_{normal} = 0.9049924$ (b) $\hat{\lambda}^*_{gamma} = 1.101615$

Figure 3.15: Wilson score interval for the estimation of $p = \mathbb{P}\{Y < \hat{\mu}_{0.95}(X)\}$ for semi-parametric model trained using training dataset with normal distribution assumption on $\hat{\lambda}^*_{normal}$ and gamma distribution assumption on $\hat{\lambda}^*_{gamma}$.

From the results in Table 3.6 and Figure 3.15, we can observe that both cases of semi-

parametric models perform better in modelling 0.95-th quantile on unseen data relative to the non-parametric model in Figure 3.8. In addition, compared to the training of full dataset in Subsection 3.6.5, the models train on proportion of the data seems to perform better in estimating for 0.95-th quantile of `O3` with respect to `Temperature`. As such, we know by our choice of hypothesis testing, semi-parametric model best suited for estimating the extreme quantile of ozone concentration depending on input temperature of air. In addition to this we can also compare the minimum cross validation score and the total pinball loss values between fitted quantile estimate each model with their respective actual responses from the test datasets. These metrics are recorded in Table 3.7 along with their statistics $\hat{p}_{mle}$ and 0.95 Wilson score interval of $p = \mathbb{P}\{Y < \hat{\mu}_{0.95}(X_{test})\}$ evaluated using unseen test dataset.

	Non-parametric model	Semi-parametric model	
		$\hat{\lambda}^*_{normal} = 0.90410$	$\hat{\lambda}^*_{gamma} = 1.1016$
$\hat{p}_{mle}$	0.94981	0.95756	0.95920
CI_{lower}	0.94044	0.94884	0.95062
CI_{upper}	0.95777	0.96485	0.95062
$\sum \rho_{0.95}(y - \hat{\mu}_{0.95}(x))$	1.8850	1.8412	1.8399
$\min\{CV(K = 5; \gamma)\}$	N/A	4.0338	4.0545

Table 3.7: Statistics $\hat{p}_{mle}$, lower boundary and upper boundary of 0.95 Wilson score interval, total pinball loss values and minimum of cross validation score values for 0.95-th non-parametric model 0.95-th semi-parametric model with $\hat{\lambda}^*_{normal} = 0.90410$ and $\hat{\lambda}^*_{gamma} = 1.1016$

.

Based on the results from Table 3.7, we observe that although every model pass the hypothesis test for $p = 0.95$ but the values for the total pinball loss of both semi-parametric model are smaller than non-parametric model implying better performance in semi-parametric model. However, the cross validation score for normal distribution model is slightly smaller than that of gamma distribution model. Due to this and the results from Table 3.6, it is justifiable to choose semi-parametric model with weight parameter $\hat{\lambda}^*_{normal} = 0.90410$ and tuning hyperparameter $\gamma^* = 5.0 \times 10^{-3}$ as our model to estimate the 0.95-th quantile of `O3` with respect to univariate `Temperature`. Moving from this, we challenge the performance of semi-parametric approach by performing the simulation for different quantile estimation $\tau = \{0.90, 0.85, 0.80, 0.75\}$. The modelling frameworks for these problems are solely based on the procedures we had done for $\tau = 0.95$. The results of these simulations are recorded in Appendix B where we store the plots for the fitted values, cross-validation score ad Wilson score intervals for our test statistics. Besides, Appendix B provides records of the performance metrics for each model in order for us to make comparisons for the better choice of quantile regression model.

Remark. In the source code attached in Appendix A, the semi-parametric model that is trained using the input training data `data_train` are the instances of `optim` function outputs declared as `train_normal` and `train_gamma`.

3.7 Summary

Focusing on the main objective in this project, Chapter 3 provides thorough explanations for the hands on simulations of modelling procedures to study the extreme quantile of ozone concentration with respect to the temperature of air in Bristol city. Recalling the three main approaches in quantile regression model, we are able to construct three main model evaluation techniques for model selection. These model evaluation techniques involve hypothesis testing, train-test split evaluation and relative comparison of performance metrics. By such we acknowledge the weaknesses of strong assumption in the distribution of response observations that increases the bias of parametric model mentioned in Section 3.2 hence making it less reliable in this scenario due to lack of information about the true population of data. Furthemore, by implementing train-test split evaluation we learn the tendencies of overtraining in non-parametric approach due to asymmetrical property of piecewise pinball loss function that is used in the supervised machine learning process for the optimisation of predictor coefficients in quantile regression. As such, we solved this solution by introducing weight parameter and applying penalised generalised additive model to manage the variability of our model. In which besides being a robust model for extreme quantile estimation, semi-parametric model also proves its quality by having the best performance with respect to our choice of metrics.

Chapter 4

Conclusion

4.1 Reflection

As the final reminder to readers, this essay demonstrates the application of generalised additive model in the modelling framework of conditional quantile estimation using a very specific method. For instance, this project exploits the convergence property of negative log ELF density function proposed by Fasiolo *et al.* (2017) [Fas+17], to define a smooth version of pinball loss function. Thus, there are many standards and rules for modelling that may work, perhaps better than the one suggested here in modelling conditional quantile of a real life data. Critically speaking, throughout this project there are some realisations towards the efficiency of the modelling procedures. The most notable issue is in the computational cost of the cross-validation procedures. This indicates the need for more efficient coding in its routine instead of the basic implementation of double for loops. Besides, there should be some reconsiderations in the choices of tuning hyperparameters for penalised generalised additive models in semi-parametric approaches. One way of improving the current method is by considering Tree-structured Parzen estimator (TPE) which leverages Bayesian optimisation for the selection of hyperparameter based on the expected improvement function (Watanabe, 2023 [Wat23]). Besides, we observe quite insignificant difference in the semi-parametric model with different distribution assumptions. Hence, questioning the reliability of our assumptions in applying Definition 3.6.2 in determining the values for our weight parameter. However, the discussion in regards to defining the optimal bandwidth for sample quantiles like in [Sha98] exceeds the standard level of comprehension of undergraduates (Shankar, 1998). Thus, for now we may only consider to possibly explore it as the extension of this project. Other than that, in aiming for minimal bias in modelling the simulation lacks robustness and reproducibility potential due to the randomisation effect in train-test sampling. Other than that, although the source code in Appendix A presents the full script for the simulations, it is not fully transferrable by copying and pasting due to the letter formatting of Latex. Hence, the reason for providing shared access to the copy of the original source code via GitHub to retrieve the fully functioning R script for those who are interested [foo24].

Bibliography

[And23] Christophe Andrieu. *Advanced linear modelling and classification (TB2, 23-24)*. Jan. 2023, pp. 41–42.

[Ava20] Anand Avati. *Bias-Variance Analysis: Theory and Practice*. 2020, pp. 6–8. URL: https://cs229.stanford.edu/summer2020/BiasVarianceAnalysis.pdf.

[Bro23] Jason Brownlee. *A Gentle Introduction to k-fold Cross-Validation*. Oct. 2023. URL: https://machinelearningmastery.com/k-fold-cross-validation/.

[BT22] Marton Balazs and Balint Toth. *Continuous conditional distributions*. University of Bristol, 2022, pp. 1–2.

[Cai23] Jerry Cain. "Maximum Likelihood Estimation". In: Stanford University, 2023. Chap. 20, pp. 20–26. URL: https://web.stanford.edu/class/archive/cs/cs109/cs109.1234/lectures/20_mle_annotated.pdf.

[Che23] Yudong Chen. *Lecture 22: Quasi-Newton: The BFGS and SR1 Methods*. University of Wisconsin-Madison, 2023, pp. 1–2.

[CS06] Ming-Yen Cheng and Shan Sun. "Bandwidth selection for kernel quantile estimation". In: *Journal of the Chinese Statistical Association* 44(3) (2006), pp. 271–295.

[Fas+17] Matteo Fasiolo et al. "Fast Calibrated Additive Quantile Regression". In: *Journal of the American Statistical Association* 116 (July 2017). DOI: 10.1080/01621459.2020.1725521.

[foo24] foo-art. *group_project_gam*. Tech. rep. 2024. URL: https://github.com/foo-art/group_project_gam.

[GS] Hani Goodarzi and Jafarpour Sina. "LECTURE 11: EXPONENTIAL FAMILY AND GENERALIZED LINEAR MODELS". In: Princeton University. Chap. 1. URL: https://www.cs.princeton.edu/courses/archive/spr09/cos513/scribe/lecture11.pdf.

[HT86] Trevor Hastie and Robert Tibshirani. "Generalized Additive Models". In: *Statistical Science* 1 (1986), p. 297.

[Iye22] Vasalakshi Iyer. *Bristol, UK Air Quality Continuous Data*. Tech. rep. UK Air Information Resource (AIR), 2022. URL: https://www.kaggle.com/datasets/visalakshiiyer/air-quality-uk-bristol?select=2021.csv.

[Jac04] Nicholas Jackie. "The first derivative and stationary points". In: University of Sydney, 2004, p. 3.

[Jam+23] Gareth James et al. "Linear Model Selection and Regularization". In: *An Introduction to Statistical Learning: with Applications in Python*. Cham: Springer International Publishing, 2023, pp. 232–233. ISBN: 978-3-031-38747-0. DOI: 10.1007/978-3-031-38747-0_6. URL: https://doi.org/10.1007/978-3-031-38747-0_6.

[Joy14] D. Joyce. “Conditional distributions Math 217 Probability and Statistics”. In: (2014), p. 1.

[Koe05] Roger Koenker. “Introduction”. In: *Quantile Regression*. Econometric Society Monographs. Cambridge University Press, 2005, pp. 1–25.

[Mar+17] L.D. Martins et al. “Extreme value analysis of air pollution data and their comparison between two large urban regions of South America”. In: *Weather and Climate Extremes* 18 (2017), pp. 44–54. URL: https://www.sciencedirect.com/science/article/pii/S2212094717300610.

[NC22] Peter Neal and Daniel Cavey. *Foundations of Statistics : Chapter 12 Conditional Distribution and Conditional Expectation*. 2022. URL: https://bookdown.org/peter_neal/math4081_notes/CondDis.html.

[Sha98] Balachander Shankar. “An optimal choice of bandwidth for perturbed sample quantiles”. In: (1998). URL: https://hdl.handle.net/2346/82414.

[SW24] Christopher Stover and Eric W. Weisstein. “"Quantile." From MathWorld–A Wolfram Web Resource.” In: (2024). URL: https://mathworld.wolfram.com/Quantile.html.

[Wat23] Shuhei Watanabe. “Tree-Structured Parzen Estimator: Understanding Its Algorithm Components and Their Roles for Better Empirical Performance”. In: Department of Computer Science, University of Freiburg, Germany, 2023, pp. 1–5.

[Whi+24] Michael Whitehouse et al. *MATH30028: Statistical Machine Learning*. University of Bristol, 2024.

[Wil27] Edwin B. Wilson. “Probable Inference, the Law of Succession, and Statistical Inference”. In: *Journal of the American Statistical Association* 22(158) (1927), pp. 209–212.

[Woo17] Simon N. Wood. “Generalized Additive Models : An Introduction with R”. In: *Texts in Statistical Science* 2 (2017), pp. 1–497.

[WPS16] Simon N. Wood, Natalya Pya, and Benjamin Safken. “Smoothing Parameter and Model Selection for General Smooth Models”. In: *Journal of the American Statistical Association* 111 (516 2016), pp. 1548–1563. URL: https://www.tandfonline.com/doi/full/10.1080/01621459.2016.1180986.

[Yu22] Feng Yu. *Statistics 1*. 2022, p. 42.

[ZWF19] Junfeng Zhang, Yongjie Wei, and Zhangfu Fang. “Ozone Pollution: A Major Health Hazard Worldwide”. In: 10 (Oct. 2019).

Appendix A

R Source Code

```
library(qgam)
library(mgcv)
library(qgam)
library(glmnet)
library(readr)

#------------------------------------------------------------------------

#Section 2.2

x <- seq(1, 10, length.out=100)
y1 <- x + rnorm(100)
y2 <- sin(x) + rnorm(100)

#linear effect of simple linear regression on Y1
linear <- glm(y1 ~ x - 1)
linear_mse <- sum(linear$residuals^2) / length(x)
linear_mse

#Figure 2.1
plot(linear)
plot(x, y1, xlab=expression(X), ylab=expression(Y1))
lines(x, linear$coefficients * x, lwd=2, col="blue")

#linear effect of simple linear regression on Y2
nonlin <- glm(y2 ~ x - 1)
nonlin_mse <- sum(nonlin$residuals^2) / length(x)
nonlin_mse

#Figure 2.1
plot(nonlin)
plot(x, y2, xlab=expression(X), ylab=expression(Y1))
lines(x, x * nonlin$coefficients, lwd=2, col="blue")

#------------------------------------------------------------------------

#Subsection 2.3.1

X <- cbind(x, x^2, x^3)
cubic <- glm(y2 ~ X)
cubic_mse <- sum(cubic$residuals^2) / length(x)
cubic_mse

plot(cubic)

```

```
#Figure 2.2
plot(x, y2, xlab = "X", ylab = "Y2")
lines(x, t(cubic$coefficients) %*% t(cbind(1, X)), lwd = 2, col="blue")

#Figure 2.3
plot(x, rep(1,length(x)) * cubic$coefficients[1], type = "l", col="blue",
    xlab = "x", ylab = expression(beta[0]))
plot(x, x * cubic$coefficients[2], type = "l", col="blue", xlab = "x", ylab
     = expression(beta[1] * x))
plot(x, x^2 * cubic$coefficients[3], type = "l", col="blue", xlab = "x",
    ylab = expression(beta[2] * x^2))
plot(x, x^3 * cubic$coefficients[4], type = "l", col="blue", xlab = "x",
    ylab = expression(beta[3] * x^3))

#-----------------------------------------------------------------------

#Subsection 2.3.2

#store x and y2 in dataframe format
df <- data.frame( X = x , Y2 = y2 )
#fit cubic spline
fitSpline <- gam(Y2 ~ s(X , bs ="cr"), data = df)

#Figure 2.4
plot( fitSpline, col="blue", xlab="X", ylab="Y2", ylim=c(min(y2), max(y2)))
points (x , y2 )

#mean squared error of model
fitSpline_mse <- sum(fitSpline$residuals^2) / length(x)
fitSpline_mse

#-----------------------------------------------------------------------

#Subsection 2.3.4

#load dataset from loon package
library(loon.data)
data("diabetes")

#define matrix to store 10 sets of 10 predictor coefficients
beta <- matrix(0, 10, 10)
for (i in 1:10){
  #initialise the hyperparameter
  gamma <- 0.01 * i
  #train cubic regression spline model using hyperparameter
  fit <- gam(RelativeWeight ~ s(FastingPlasmaGlucose, k = 10, bs = "cr"),
              data = diabetes, scale = gamma )
  #store the predictor coefficients
  beta [i,] <- fit$coefficients
}

#Figure 2.5
plot(seq(1, 10), rep(0, 10), type="b", xlab=expression(gamma), ylab=
    expression(beta), ylim=c(-0.10, 0.10))
for (i in 2:10){
  lines(seq(1, 10), beta[,i], type="b", col=i)
}

#-----------------------------------------------------------------------

#Section 3.2

df <- data.frame( X = x , Y2 = y2 )
```

```
#fit cubic regression spline onto data
fitSpline <- gam(Y2 ~ s(X , bs ="cr"), data = df)
#output the fitted values
y2pred <- predict(fitSpline)

#compute the 0.95-th conditional quantile for each fitted values
cond_quan <- rep(0, length(y2pred))
for (i in 1:length(y2pred)){
  cond_quan[i] <- qnorm(0.95, y2pred[i], 1)
}

#Figure 3.1
plot( fitSpline, col="blue", xlab="X", ylab="Y2", ylim=c(min(y2), max(y2)))
points(x, y2)
lines(x, cond_quan, col = "red")

#-------------------------------------------------------------------------

#Subsection 3.2.1

#load the dataset
#may consider changing the location of file
X2021 <- read_delim("C:/Users/Fuaddin/Documents/2021.csv", delim = ";",
                    escape_double = FALSE, trim_ws = TRUE)

#data manipulation process
data <- na.omit(data.frame(y=X2021$O3, x=X2021$Temperature))
data <- data[data$y > 0,]
n <- dim(data)[1]
#observe the data
head(data)

#Figure 3.2
plot(data$x, data$y, ylab = "Temperature", xlab = "O3")

x <- data$x
y <- data$y
#create 20 cubic spline basis transformations of the observations x
sm <- smoothCon(s(x, k = 20, bs = "cr"), data=data, knots=NULL)[[1]]
X <- sm$X

#fit the cubic regression spline model
fit <- gam(y ~ X - 1, data=data)

#Figure 3.3
plot(data$x, data$y, xlab='Temperature', ylab='O3', col='grey')
lines(data$x[order(data$x)], X[order(data$x),]%*%fit$coefficients, lwd=2,
    col='blue')

#calculating the parameter for the quantile estimation
yPred <- predict(fit)
condQuan <- rep(0, length(yPred))
sigma <- sqrt(sum((data$y - yPred)^2) / length(yPred))
#estimate the quantile using parametric method
for (i in 1:length(yPred)){
  condQuan[i] <- qnorm(0.95, yPred[i], sigma)}
plot(data$x, data$y, xlab='Temperature', ylab='O3', col='grey', ylim=c(-2,
    150))
lines(data$x[order(data$x)], condQuan[order(data$x)], lwd=2, col='red')

#-------------------------------------------------------------------------

#Subsection 3.3.1
```

```
165
166 #significance level
167 a <- 0.05
168
169 #calculate the statistics and store as p_hat
170 w <- rep (0 , length (y))
171 for (i in 1:length(y)){
172   if (y[i] < condQuan[i]){
173     w[i] <- 1
174   }
175 }
176 p_hat <- sum(w) / length(y)
177
178 #calculate the 0.95 wilson score interval
179 z <- qnorm(1 - a/2, 0, 1)
180 r1 <- p_hat + z^2 / (2 * n)
181 r2 <- z * sqrt((p_hat * (1 - p_hat)) / n + z^2 / (4 * n^2))
182 r3 <- 1 + z^2 / n
183 CI <- c((r1 - r2) / r3, (r1 + r2) / r3)
184
185 print(paste("The confidence interval for p is (",CI[1],", ",CI[2],")"))
186
187 #------------------------------------------------------------------------
188
189 #Subsection 3.4.1
190
191 tau <- 0.95
192 x <- data$x
193 y <- data$y
194 sm <- smoothCon(s(x, k = 20, bs = "cr"), data=data, knots=NULL)[[1]]
195 X <- sm$X
196 n <- length(y)
197 iBeta <- fit$coefficients
198
199 # takes inputs quantile , basis transformed covariates , response ,
      parameter
200 loss_pinball <- function(tau, X, y, par){
201   # define the quantile estimate mu
202   mu <- X %*% par
203   # define z as difference between response and estimate
204   z <- y - mu
205   p <- rep(0, length(y))
206   # compute pinball loss in terms of z
207   p[z >= 0] <- tau * z[z >= 0]
208   p[z < 0] <- (tau - 1) * z[z < 0]
209   return(sum(p))
210 }
211
212 # takes inputs quantile , basis transformed covariates , response ,
      parameter
213 grad_pinball <- function(tau, X, y, par){
214   # define the quantile estimate mu
215   mu <- X %*% par
216   # define z as difference between response and estimate
217   z <- y - mu
218   g <- rep(0, length(y))
219   # compute derivative of pinball loss in terms of z
220   g[z >= 0] <- tau
221   g[z < 0] <- tau - 1
222   return(g)
223 }
224
```

```
# takes inputs quantile , basis transformed covariates , response , parameter
negGrad_pinball <- function(tau, X, y, par){
  # find the directional derivative for each components of X
  tmp <- lapply(1:length(y),
  function(ii){
      # use negative sign gradient since we want to minimise the function
      a <- -grad_pinball(tau, X[ii,], y[ii], par)
      return( a * X[ii, ])})
  out <- Reduce("+", tmp)
  return(out)
}

fit_pinball <- optim(par = iBeta, loss_pinball, gr = negGrad_pinball, method ="BFGS", tau = tau, X = X, y = data$y)

#Figure 3.5
plot(x, y, xlab = 'Temperature', ylab = 'O3', col = 'grey', ylim=c(-2, 150))
lines(x[order(x)], X[order(x),] %*% fit_pinball$par, col = 'blue', lwd = 2)
lines(data$x[order(data$x)], condQuan[order(data$x)], col='red', lwd = 2)
legend(-5, 130, legend=c("Parametric model", "Non-parametric model"),
       col=c("red", "blue"), lty=1:1, cex=0.8)

#-------------------------------------------------------------------------

#Subsection 3.4.2

#compute the statistic and Wilson score interval for hypothesis test
a <- 0.05
w <- rep(0, length(y))
for (i in 1:length(y)){
  if (y[i] <= X[i,] %*% fit_pinball$par){
    w[i] <- 1
  }
}

p_hat <- sum(w) / length(y)

z <- qnorm(1 - a, 0, 1)
r1 <- p_hat + z^2 / (2 * n)
r2 <- z * sqrt((p_hat * (1 - p_hat)) / n + z^2 / (4 * n^2))
r3 <- 1 + z^2 / n
CI <- c((r1 - r2) / r3, (r1 + r2) / r3)

print(paste("The confidence interval for p is (",CI[1],", ",CI[2],")"))


# takes input dataframe and number of bins to make
create_bin <- function (data, k){
  x <- data$x
  range <- seq(min (x), max(x))
  # set k number of percentiles to partition data
  percentile <- seq(0 , 100 , 100/k)
  # set the highest percentile to 100
  percentile[length(percentile)] <- 100
  # find k number of evenly spaced percentile using quantile
  breaks <- quantile (range , probs = percentile/100)
  # for the first percentile we minus the minimum observations by one
  # since cut consider open interval to define the lower bound of bin
  breaks [1] <- min(x) - 1
  # the last percentile be the maximum observations
  breaks[length(breaks)] <- max(x)
```

```
  # classify the observations into k bins using cut function
  bin <- cut(x, breaks = breaks)
  # append the bin ranges into the original dataframe
  data$bins <- bin
  return(data)
}

# takes input dataframe and predictor coefficient
quantile_bin <- function(data, sm, par){
  y <- data$y
  x <- data$x
  # initialised the bins for observations in dataframe data
  blocks <- sort(unique(data$bins))
  k <- length(blocks)
  # initialised k length of zero vector to store the estimate of p
  p <- rep(0, k)
  for(i in 1:length(blocks)){
    # collect observations from the same bin
    ranged <- data[data$bins == blocks[i],]
    sum_p <- 0
    for(j in 1:dim(ranged)[1]){
      # calculate the predicted conditional quantile of data in the bin
      pred <- PredictMat(sm, data.frame(x = ranged$x[j]))%*%par
      # if observation of response is smaller than predicted quantile
      if(ranged$y[j] < pred){
        # increment the count for sum_z
        sum_p <- sum_p + 1
        }
      }
    # define estimate of p as as sum_z divided by number of data in bin
    p[i] <- sum_p / dim(ranged)[1]
    }
  return (p)
}

# takes input estimate of p , dataframe and significance level
ci <- function(qb_estim, data, a = 0.05){
  k <- length(qb_estim)
  # initialised the bins for observations in dataframe data
  blocks <- sort(unique(data$bins))
  # define the 1 - a/2 quantile of standard normal distribution
  z <- qnorm(1 - a/2, 0, 1)
  # create dataframe to store the estimate of p , number of observations ,
  # ranges of bins , lower bound interval , upper bound interval
  df <- data.frame(quantile_estimate = qb_estim, number = rep(0, k),
                   bins = blocks, ci_low = rep (0, k), ci_up = rep(0, k))
  for(i in 1:k){
    # compute the wilson score interval for every estimate of p
    p <- qb_estim[i]
    n <- dim(data[data$bins == blocks[i],])[1]
    r1 <- p + z^2 / (2 * n)
    r2 <- z * sqrt ((p * (1 - p)) / n + z^2 / (4 * n^2))
    r3 <- 1 + z^2 / n
    low <- (r1 - r2) / r3
    up <- (r1 + r2) / r3

    # store the number of observations in bin
    df[i,] $number <- n
    # store the lower bound of wilson score
    df[i,] $ci_low <- low
    # store the upper bound of wilson score
    df[i,] $ci_up <- up
    }
```

```
  return (df)
}

# takes input of wilson score interval and quantile
ci_bin <- function(ci_estim, tau){
  k <- dim(ci_estim)[1]
  range <- seq(1, k)
  # plot the estimate of p for each bins
  plot(range, ci_estim$quantile_estimate, ylim = c(min(ci_estim$ci_low),
    max(ci_estim$ci_up)), xlab = "Bins", ylab = "probability")
  abline(h = tau, col = "red")

  for (i in 1:k){
    # plot the wilson score interval for each bins
    arrows(range[i], ci_estim[i,]$ci_low, range[i], ci_estim[i,]$ci_up,
    angle = 90, code = 3, length = 0.1)
  }
  return (ci_estim)
}

#partition our data into 5 bins
data <- create_bin(data, 5)
# define the basis functions in terms of univariate x
x <- data$x
sm <- smoothCon(s(x, k = 20, bs = "cr"), data = data , knots = NULL)[[1]]
# calculate the statistics in each bin
qb_pinball <- quantile_bin(data, sm, fit_pinball$par)
# compute Wilson score interval for p
df_pinball <- ci(qb_pinball, data, 0.05)
# plot the results
ci_bin(df_pinball, tau)

#------------------------------------------------------------------------

#Section 3.5

#define index for training samples
train <- sample(1:dim(data)[1], size = 0.7 * length(x), replace = FALSE)

#train-test split
data_train <- data[train,]
data_test <- data[-train,]

x_train <- data_train$x
y_train <- data_train$y

#initialised test set
x_test <- data_test$x
y_test <- data_test$y

#tranform x into cubic spline basis term
sm <- smoothCon(s(x, k = 20, bs = "cr"), data=data_train, knots=NULL)[[1]]
X_train <- sm$X

tau <- 0.95
#initialiser for predictor coefficients
iBeta <- fit$coefficients

#train model using training data
train_pinball <- optim(par = iBeta, loss_pinball, gr = negGrad_pinball,
                       method = "BFGS", tau = tau, X = X_train, y = y_train
    )

```

```
#to transform univariate in test set into cubic spline terms
X_test <- PredictMat(sm, data.frame(x=x_test))
#calculate the pinball loss between fitted values and actual response in
    test set
pinball_non <- loss_pinball(tau, X_test, y_test, train_pinball$par) /
    length(y_test)
#return the values
pinball_non

#Figure 3.7
plot(x_test, y_test, xlab="Temperature", ylab="O3", col = "grey")
lines(x_test[order(x_test)], X_test[order(x_test),] %*% fit_pinball$par,
    type = 'l', col = "blue", lwd = 2)
lines(x_test[order(x_test)], X_test[order(x_test),] %*% train_pinball$par,
    type = 'l', col = "red", lwd = 2)
legend(-5, 130, legend=c("Training data", "Full data"),
       col=c("red", "blue"), lty=1:1, cex=0.8)

#set the number of bin samples to 1 since we want to check for whole test
    set
data_single_test <- create_bin(data_test, 1)
#quantile_bin function to compute the statistics
#ci function to compute the Wilson score interval
ci(quantile_bin(data_single_test, sm, train_pinball$par), data_single_test,
     0.05)

#hypothesis test for bin samples
qbt_pinball <- quantile_bin(data_test, sm, train_pinball$par)
dft_pinball <- ci(qbt_pinball, data_test, 0.05)
#Table 3.1 and Figure 3.8
ci_bin(dft_pinball, tau)


#-------------------------------------------------------------------------

#Subsection 3.6.1

y <- rnorm(1000, 0, 1)
tau <- 0.95
true_quantile <- qnorm(tau, 0, 1)
pinball <- rep(0, length(y))

for(i in 1:length(y)){
  z <- y[i] - true_quantile
  if(z < 0){
    pinball[i] <- -(1 - tau) * z
  }
  else{
    pinball[i] <- tau * z
  }
}

sig <- 3
lam <- 0.1
log_elf <- rep(0, length(y))
for(i in 1:length(y)){
  z <- (y[i] - true_quantile) / sig
  log_elf[i] <- (1-tau) *  - lam * log1pexp( z / lam )
}

#Figure 3.10
hist(y, probability = TRUE, main = "Y~N(0,1)")
lines(y[order(y)], pinball[order(y)], lwd = 2, col = 'blue')
```

```
lines(y[order(y)], - log_elf[order(y)] * 10, lwd = 2, col = 'red')
abline(v = true_quantile, lty = 2, lwd = 2, col = 'darkgreen')
legend(-3, 0.3, legend=c("Pinball loss", "Negative log ELF", "0.95-th
    quantile"),
       col=c("blue", "red", "darkgreen"), lty = 1:1:2, cex = 0.7)

#------------------------------------------------------------------------

#Subsection 3.6.3

#computes ELF density and its derivatives w.r.t mu
dlf <- function(y, tau, mu, lam, log = FALSE, deriv = 0)
{
  #calculate the log of ELF density function
  sig <- 1 #fixed sigma to 1
  z <- (y - mu) / sig
  out <- (1-tau) * z - lam * log1pexp( z / lam ) -
    log( sig * lam * beta(lam*(1-tau), tau*lam) )
  #if not log return the ELF density function
  if( !log ) out <- exp(out)
  #if deriv is positive compute the derivative of the log ELF density
  if( deriv > 0 )
  {
    out <- list("d" = out)
    dl <- dlogis(y, mu, lam*sig)
    pl <- plogis(y, mu, lam*sig)
    out$D <- sum((pl - (1-tau)) / sig)
    #if deriv is more than one compute the second derivative
    if(deriv > 1)
    {
      out$D2 <- sum( - dl / sig )
    }
  }
  return(out)
}


#takes input hyperparameter, predictor coefficient, positive semidefinite
penalty <- function(k, par, S = sm$S[[1]], deriv = 0)
{
  #compute the penalty component in the generalised pinball loss
  w <- k * t(par) %*% S %*% par
  w <- list("w" = w)
  #if deriv is positive compute the derivative of the penalty component
  if (deriv > 0)
  {
    w$D <- 2 * k * t(par) %*% S
  }
  return(w)
}


#takes input predictor coefficient, quantile, weight, covariates,
#response observation, semi positive definite matrix and hyperparameter
penalised_negllkFun <- function(par, tau, lam, X, y, S = sm$S[[1]], k = 0)
{
  #compute the conditional quantile estimate
  mu <- X %*% par
  #compute the semi-parametric objective function
  out <- - sum( dlf(y = y, tau = tau, mu = mu, lam = lam, log = TRUE))
  + penalty(k, par, S)$w
  return(out)
}
```

```
527
528 #takes input predictor coefficient, quantile, weight, covariates,
529 #response observation, semi positive definite matrix and hyperparameter
530 penalised_negGrad <- function(par, tau, lam, X, y, S = sm$S[[1]], k = 0)
531 {
532   #compute the conditional quantile estimate
533   mu <- X %*% par
534   #compute the descending gradient of semi-parametric objective function
535   tmp <- lapply(1:length(mu),
536                 function(ii){
537                   a <- - dlf(y = y[ii], tau = tau,
538                              mu = mu[ii], lam = lam, log = TRUE, deriv = 1)
    $D
539
540                   return( a * X[ii, ] + penalty(k, par, S, deriv = 1)$D )})
541   out <- Reduce("+", tmp)
542   return(out)
543 }
544
545 #-----------------------------------------------------------------------------
546
547 #Algorithm 1 (normal distribution)
548
549 # to compute the negative log likelihood function for normal assumption
550 negative_log_likelihood_normal <- function(params, y){
551   mu <- params[1]
552   sigma <- params[2]
553   # Calculate negative log-likelihood
554   neg_log <- -sum(dnorm(y, mean = mu, sd = sigma, log = TRUE))
555
556   return(neg_log)
557 }
558
559 # Function to estimate ML parameters of normal distribution using optim
560 estimate_normal_mle <- function(y){
561   # Initial guess for parameters
562   par_0 <- c(mu = 1, sigma = 1)
563   # Minimize negative log-likelihood using optim
564   result <- optim(par = par_0, fn = negative_log_likelihood_normal, y = y,
      method = "L-BFGS-B")
565   # Extract estimated parameters
566   par_mle <- result$par
567
568   return(par_mle)
569 }
570
571 # to compute the weight parameter of the ELF loss function by normal
      assumption
572 lambda_normal <- function(tau, y){
573   par <- estimate_normal_mle(y)
574   n <- length(y)
575   # a <- 0.5(mean(y) - mean(log(y)))
576   mu_tau <- qnorm(tau, par[1], par[2])
577   f_mu0 <- dnorm(mu_tau, par[1], par[2])
578   f_mu1 <- dnorm(mu_tau + 1e-10, par[1], par[2])
579   #using numerical method to estimate parameter of distribution in the pdf
580   f_gau_prime <- (f_mu0 - f_mu1) / 1e-10
581   out <- (9 / (n * pi^4) * f_mu0 / f_gau_prime^2 ) ^ (1/3)
582
583   return(out)
584 }
585
586 #-----------------------------------------------------------------------------
```

```
587
588 #Algorithm 1 (gamma distribution)
589
590 # to compute the negative log likelihood function for gamma assumption
591 negative_log_likelihood_gamma <- function(params, y) {
592   a <- params[1]
593   b <- params[2]
594   # Calculate negative log-likelihood
595   neg_log <- -sum(dgamma(y, shape = a, rate = b, log = TRUE))
596
597   return(neg_log)
598 }
599
600 # Function to estimate ML parameters of gamma distribution using optim
601 estimate_gamma_mle <- function(y) {
602   # Initial guess for parameters
603   par_0 <- c(a = 10, b = 10)
604   # Minimize negative log-likelihood using optim
605   result <- optim(par = par_0, fn = negative_log_likelihood_gamma, y = y,
      method = "L-BFGS-B", lower=c(0.1,0.1))
606   # Extract estimated parameters
607   par_mle <- result$par
608
609   return(par_mle)
610 }
611
612 # to compute the weight parameter of the ELF loss function by gamma
      assumption
613 lambda_gamma <- function(tau, y){
614   par <- estimate_gamma_mle(y)
615   n <- length(y)
616   mu_tau <- qgamma(tau, par[1], par[2])
617   f_mu0 <- dgamma(mu_tau, par[1], par[2])
618   f_mu1 <- dgamma(mu_tau + 1e-10, par[1], par[2])
619   #using numerical method to estimate parameter of distribution in the pdf
620   f_gau_prime <- (f_mu0 - f_mu1) / 1e-10
621   out <- (9 / (n * pi^4) * f_mu0 / f_gau_prime^2 ) ^ (1/3)
622
623   return(out)
624 }
625
626 #-----------------------------------------------------------------------------
627
628 #Subsection 3.6.4
629
630 #Algorithm 2 (gamma distribution)
631
632 #refer to pseudo code of Algorithm 2
633 #takes input covariates, response, initial predictors, hyperparameters
      candidates
634 cvFive_gamma <- function(X, y, tau, par, hp_can){
635   #line 1
636   cv <- rep(1e10, length(hp_can))
637   #determine 5-fold cv
638   loop <- seq(1, dim(X)[1], floor(dim(X)[1]/5))
639   loop <- append(loop, dim(X)[1])
640   #line 2
641   for(k in 1:length(hp_can)){
642     sum_cv <- 0
643     #line 3
644     for (i in 1:(length(loop) - 1)){
645       index <- seq(-loop[i], -loop[i + 1])
646       #line 4
```

```
      l_gamma <- lambda_gamma(tau, y[index])
      #line 7
      fit <- optim(par = par, penalised_negllkFun, gr = penalised_negGrad,
    method = "BFGS", tau = tau,
                   lam = l_gamma, X = X[index, ], y = y[index], k = hp_can
    [k])
      #line 8
      sum_cv <- sum_cv - sum(dlf(y[-index], tau, X[-index, ] %*% fit$par,
    l_gamma, log = TRUE)) / length(y[-index])
    }
    #line 9
    cv[k] <- sum_cv / length(loop)
  }
  #line 10
  return(cv)
}


#Algorithm 2 (normal distribution)

#takes input covariates, response, initial predictors, hyperparameters
    candidates
cvFive_normal <- function(X, y, tau, par, hp_can){
  cv <- rep(1e10, length(hp_can))
  loop <- seq(1, dim(X)[1], floor(dim(X)[1]/5))
  loop <- append(loop, dim(X)[1])
  for(k in 1:length(hp_can)){
    sum_cv <- 0
    for (i in 1:(length(loop) - 1)){
      index <- seq(-loop[i], -loop[i + 1])
      l_normal <- lambda_normal(tau, y[index])
      fit <- optim(par = par, penalised_negllkFun, gr = penalised_negGrad,
    method = "BFGS", tau = tau,
                 lam = l_normal, X = X[index, ], y = y[index], k = hp_can
    [k])
      sum_cv <- sum_cv - sum(dlf(y[-index], tau, X[-index, ] %*% fit$par,
    l_normal, log = TRUE)) / length(y[-index])
    }
    cv[k] <- sum_cv / length(loop)
  }
  return(cv)
}

#-------------------------------------------------------------------------

#initialised important values
tau <- 0.95
y <- data$y
x <- data$x
sm <- smoothCon(s(x, k = 20, bs = "cr"), data=data, knots=NULL)[[1]]
X <- sm$X
#tuning hyperparameter candidates
hp_can <- c(5*1e-6, 1e-5, 5*1e-5, 1e-4, 5*1e-4, 1e-3, 5*1e-3, 1e-2)

#five-fold cv to choose hyperparameter for gamma distribution
cv_gamma <- cvFive_gamma(X, y, tau, fit$coefficients, hp_can)
#five-fold cv to choose hyperparameter for normal distribution
cv_normal <- cvFive_normal(X, y, tau, fit$coefficients, hp_can)
total_cv <- c(cv_gamma, cv_normal)

#Figure 3.11
plot(log(hp_can), cv_normal, type = "b", xlab = "log(gamma)", ylab = "CV",
    col="red", ylim = c(min(total_cv), max(total_cv)))
```

```
lines(log(hp_can), cv_gamma, type = "b", col="blue")
legend(-12, 4.078, legend=c("Normal Distribution", "Gamma Distribution"),
       col=c("red", "blue"), lty=1:1, cex=0.8)

#compute the weight parameter for gamma assumption
l_gamma <- lambda_gamma(tau, y)
#compute the weight parameter for normal assumption
l_normal <- lambda_normal(tau, y)

#extract the best tuning hyperparameter
best_hp_gamma <- hp_can[which.min(cv_gamma)]
#fit semi-parametric model using the hyperparameter
best_gamma <- optim(par = fit$coefficients, penalised_negllkFun, gr =
    penalised_negGrad, method = "BFGS", tau = tau,
                    lam = l_gamma, X = X, y = y, k = best_hp_gamma)
best_hp_normal <- hp_can[which.min(cv_normal)]
best_normal <- optim(par = fit$coefficients, penalised_negllkFun, gr =
    penalised_negGrad, method = "BFGS", tau =tau,
                     lam = l_normal, X = X, y = y, k = best_hp_normal)

#Figure 3.12
plot(x[order(x)], y[order(x)], xlab="Temperature", ylab="O3", col = "grey")
#plotting the quantile estimate
lines(x[order(x)], X[order(x),]%*%best_gamma$par, type = 'l', col = "red",
    lwd=2)
lines(x[order(x)], X[order(x),]%*%best_normal$par, type = 'l', col = "blue
    ", lwd=2)
legend(-5, 120, legend=c("Gamma Distribution", "Normal Distribution"),
       col=c("red", "blue"), lty=1:1, cex=0.8)

#-------------------------------------------------------------------------

#Subsection 3.6.5

data_single <- create_bin(data, 1)
#best_gamma$par is the semi-parametric model coefficients for gamma
    assumption
ci(quantile_bin(data_single, sm, best_gamma$par), data_single, 0.05)
#best_normal$par is the semi-parametric model coefficients for normal
    assumption
ci(quantile_bin(data_single, sm, best_normal$par), data_single, 0.05)

#hypothesis test using gamma assumption
gamma_best_qb <- quantile_bin(data, sm, best_gamma$par)
df_gamma <- ci(gamma_best_qb, data, 0.05)
#Figure 3.13(b)
ci_bin(df_gamma, tau)

#hypothesis test using normal assumption
normal_best_qb <- quantile_bin(data, sm, best_normal$par)
df_normal <- ci(normal_best_qb, data, 0.05)
#Figure 3.13(a)
ci_bin(df_normal, tau)

#-------------------------------------------------------------------------

#Subsection 3.6.6

#train-test split data
train <- sample(1:dim(data)[1], size = 0.7 * length(x), replace = FALSE)

data_train <- data[train,]
data_test <- data[-train,]
```

```
data_single_test <- data_single[-train,]

x_train <- data_train$x
y_train <- data_train$y

x_test <- data_test$x
y_test <- data_test$y

sm <- smoothCon(s(x, k = 20, bs = "cr"), data=data_train, knots=NULL)[[1]]
X_train <- sm$X

#compute weight parameter for both distribution assumptions
l_gamma <- lambda_gamma(tau, y_train)
l_normal <- lambda_normal(tau, y_train)

#train semi-parametric model using training data
train_gamma <- optim(par = fit$coefficients, penalised_negllkFun, gr =
    penalised_negGrad, method = "BFGS", tau = tau,
                     lam = l_gamma, X = X_train, y = y_train, k =
    best_hp_gamma)
train_normal <- optim(par = fit$coefficients, penalised_negllkFun, gr =
    penalised_negGrad, method = "BFGS", tau =tau,
                      lam = l_normal, X = X_train, y = y_train, k =
    best_hp_normal)

#Figure 3.14
X_test <- PredictMat(sm, data.frame(x=x_test))
plot(x_test[order(x_test)], y_test[order(x_test)], xlab="Temperature", ylab
    ="O3", col = "grey", ylim=c(-2, 140))
#plotting the quantile estimate
lines(x_test[order(x_test)], X_test[order(x_test),]%*%train_gamma$par, type
     = 'l', col = "red", lwd=2)
lines(x_test[order(x_test)], X_test[order(x_test),]%*%train_normal$par,
    type = 'l', col = "blue", lwd=2)
legend(-5, 120, legend=c("Gamma Distribution", "Normal Distribution"),
       col=c("red", "blue"), lty=1:1, cex=0.8)


#train_gamma$par is the semi-parametric model coefficients for gamma
    assumption
ci(quantile_bin(data_single_test, sm, train_gamma$par), data_single_test,
    0.05)
#train_normal$par is the semi-parametric model coefficients for normal
    assumption
ci(quantile_bin(data_single_test, sm, train_normal$par), data_single_test,
    0.05)

#hypothesis test for bin samples
gamma_train_qb <- quantile_bin(data_test, sm, train_gamma$par)
df_test_gamma <- ci(gamma_train_qb, data_test, 0.05)
ci_bin(df_test_gamma, tau)

normal_train_qb <- quantile_bin(data_test, sm, train_normal$par)
df_test_normal <- ci(normal_train_qb, data_test, 0.05)
ci_bin(df_test_normal, tau)

#total pinball loss values
pinball_gamma <- loss_pinball(tau, X_test, y_test, train_gamma$par) /
    length(y_test)
pinball_gamma
pinball_normal <- loss_pinball(tau, X_test, y_test, train_normal$par) /
    length(y_test)
pinball_normal
```

```
809
810 #find the minimum cross validation score
811 min(cv_gamma)
812 min(cv_normal)
813
814 #-------------------------------------------------------------------------
815
816 #Appendix B
817
818 #Simulation for 0.90-th semi-parametric quantile regression model
819 tau1 <- 0.9
820 y <- data$y
821 x <- data$x
822 sm <- smoothCon(s(x, k = 20, bs = "cr"), data=data, knots=NULL)[[1]]
823 X <- sm$X
824 hp_can1 <- c(5*1e-5, 1e-4, 5*1e-4, 1e-3, 5*1e-3, 1e-2, 5e-2, 1e-1)
825 loop <- seq(1, dim(X)[1], floor(dim(X)[1]/5))
826 loop <- append(loop, dim(X)[1])
827
828 cv_gamma1 <- cvFive_gamma(X, y, tau1, fit$coefficients, hp_can1)
829 cv_normal1 <- cvFive_normal(X, y, tau1, fit$coefficients, hp_can1)
830 total_cv1 <- c(cv_gamma1, cv_normal1)
831
832 #Figure B.1
833 plot(log(hp_can1), cv_normal1, type = "b", xlab = "log(gamma)", ylab = "CV
      ", col="red", ylim = c(min(total_cv1), max(total_cv1)))
834 lines(log(hp_can1), cv_gamma1, type = "b", col="blue")
835 legend(-5, 6.90, legend=c("Normal Distribution", "Gamma Distribution"),
836        col=c("red", "blue"), lty=1:1, cex=0.8)
837
838 best_hp_gamma1 <- hp_can1[which.min(cv_gamma1)]
839 best_hp_gamma1
840 best_hp_normal1 <- hp_can1[which.min(cv_normal1)]
841 best_hp_normal1
842
843 set.seed(1)
844 train1 <- sample(1:dim(data)[1], size = 0.7 * length(x), replace = FALSE)
845
846 data_train1 <- data[train1,]
847 data_test1 <- data[-train1,]
848 data_single_test1 <- data_single[-train1,]
849
850 x_train1 <- data_train1$x
851 y_train1 <- data_train1$y
852
853 x_test1 <- data_test1$x
854 y_test1 <- data_test1$y
855
856 sm <- smoothCon(s(x, k = 20, bs = "cr"), data=data_train1, knots=NULL)[[1]]
857 X_train1 <- sm$X
858
859 l_gamma1 <- lambda_gamma(tau1, y_train1)
860 l_normal1 <- lambda_normal(tau1, y_train1)
861
862
863 train_gamma1 <- optim(par = fit$coefficients, penalised_negllkFun, gr =
      penalised_negGrad, method = "BFGS", tau = tau1,
864                       lam = l_gamma1, X = X_train1, y = y_train1, k =
      best_hp_gamma1)
865 train_normal1 <- optim(par = fit$coefficients, penalised_negllkFun, gr =
      penalised_negGrad, method = "BFGS", tau =tau1,
866                        lam = l_normal1, X = X_train1, y = y_train1, k =
      best_hp_normal1)
```

```
867
868 #Figure B.1
869 X_test1 <- PredictMat(sm, data.frame(x=x_test1))
870 plot(x_test1[order(x_test1)], y_test1[order(x_test1)], xlab="Temperature",
        ylab="O3", col = "grey", ylim=c(-2, 140))
871 #plotting the quantile estimate
872 lines(x_test1[order(x_test1)], X_test1[order(x_test1),]%*%train_gamma1$par,
         type = 'l', col = "red", lwd=2)
873 legend(-5, 120, legend="Gamma Distribution", col="red", lty=1, cex=0.8)
874
875 plot(x_test1[order(x_test1)], y_test1[order(x_test1)], xlab="Temperature",
        ylab="O3", col = "grey", ylim=c(-2, 140))
876 lines(x_test1[order(x_test1)], X_test1[order(x_test1),]%*%train_normal1$par
        , type = 'l', col = "blue", lwd=2)
877 legend(-5, 120, legend="Normal Distribution", col="blue", lty=1, cex=0.8)
878
879 #Figure B.2
880 gamma_train_qb1 <- quantile_bin(data_test1, sm, train_gamma1$par)
881 df_test_gamma1 <- ci(gamma_train_qb1, data_test1, 0.05)
882 ci_bin(df_test_gamma1, tau1)
883
884 #Figure B.2
885 normal_train_qb1 <- quantile_bin(data_test1, sm, train_normal1$par)
886 df_test_normal1 <- ci(normal_train_qb1, data_test1, 0.05)
887 ci_bin(df_test_normal1, tau1)
888
889
890 l_gamma1
891 l_normal1
892
893 best_hp_gamma1
894 best_hp_normal1
895
896 #train_gamma$par is the semi-parametric model coefficients for gamma
        assumption
897 ci(quantile_bin(data_single_test1, sm, train_gamma1$par), data_single_test1
        , 0.05)
898 #train_normal$par is the semi-parametric model coefficients for normal
        assumption
899 ci(quantile_bin(data_single_test1, sm, train_normal1$par),
        data_single_test1, 0.05)
900
901 pinball_gamma1 <- loss_pinball(tau1, X_test1, y_test1, train_gamma1$par) /
        length(y_test1)
902 pinball_gamma1
903 pinball_normal1 <- loss_pinball(tau1, X_test1, y_test1, train_normal1$par)
        / length(y_test1)
904 pinball_normal1
905
906 min(cv_gamma1)
907 min(cv_normal1)
908 #------------------------------------------------------------------------
909
910 #Simulation for 0.85-th semi-parametric quantile regression model
911
912 tau2 <- 0.85
913 y <- data$y
914 x <- data$x
915 sm <- smoothCon(s(x, k = 20, bs = "cr"), data=data, knots=NULL)[[1]]
916 X <- sm$X
917 hp_can2 <- c(5*1e-5, 1e-4, 5*1e-4, 1e-3, 5*1e-3, 1e-2, 5e-2, 1e-1)
918 loop <- seq(1, dim(X)[1], floor(dim(X)[1]/5))
919 loop <- append(loop, dim(X)[1])
```

```
920
921 cv_gamma2 <- cvFive_gamma(X, y, tau2, fit$coefficients, hp_can2)
922 cv_normal2 <- cvFive_normal(X, y, tau2, fit$coefficients, hp_can2)
923 total_cv2 <- c(cv_gamma2, cv_normal2)
924
925 #Figure B.3
926 plot(log(hp_can2), cv_normal2, type = "b", xlab = "log(gamma)", ylab = "CV
    ", col="red", ylim = c(min(total_cv2), max(total_cv2)))
927 lines(log(hp_can2), cv_gamma2, type = "b", col="blue")
928 legend(-10, 5.42, legend=c("Normal Distribution", "Gamma Distribution"),
929        col=c("red", "blue"), lty=1:1, cex=0.8)
930
931
932 best_hp_gamma2 <- hp_can2[which.min(cv_gamma2)]
933 best_hp_normal2 <- hp_can2[which.min(cv_normal2)]
934
935 set.seed(2)
936 train2 <- sample(1:dim(data)[1], size = 0.7 * length(x), replace = FALSE)
937
938 data_train2 <- data[train2,]
939 data_test2 <- data[-train2,]
940 data_single_test2 <- data_single[-train2,]
941
942 x_train2 <- data_train2$x
943 y_train2 <- data_train2$y
944
945 x_test2 <- data_test2$x
946 y_test2 <- data_test2$y
947
948 sm <- smoothCon(s(x, k = 20, bs = "cr"), data=data_train2, knots=NULL)[[1]]
949 X_train2 <- sm$X
950
951 l_gamma2 <- lambda_gamma(tau2, y_train2)
952 l_normal2 <- lambda_normal(tau2, y_train2)
953
954 train_gamma2 <- optim(par = fit$coefficients, penalised_negllkFun, gr =
    penalised_negGrad, method = "BFGS", tau = tau2,
955                       lam = l_gamma2, X = X_train2, y = y_train2, k =
    best_hp_gamma2)
956 train_normal2 <- optim(par = fit$coefficients, penalised_negllkFun, gr =
    penalised_negGrad, method = "BFGS", tau =tau2,
957                        lam = l_normal2, X = X_train2, y = y_train2, k =
    best_hp_normal2)
958
959 #Figure B.3
960 X_test2 <- PredictMat(sm, data.frame(x=x_test2))
961 plot(x_test2[order(x_test2)], y_test2[order(x_test2)], xlab="Temperature",
    ylab="O3", col = "grey", ylim=c(-2, 140))
962 #plotting the quantile estimate
963 lines(x_test2[order(x_test2)], X_test2[order(x_test2),]%*%train_gamma2$par,
     type = 'l', col = "red", lwd=2)
964 legend(-5, 120, legend="Gamma Distribution", col="red", lty=1, cex=0.8)
965
966 plot(x_test2[order(x_test2)], y_test2[order(x_test2)], xlab="Temperature",
    ylab="O3", col = "grey", ylim=c(-2, 140))
967 lines(x_test2[order(x_test2)], X_test2[order(x_test2),]%*%train_normal2$par
    , type = 'l', col = "blue", lwd=2)
968 legend(-5, 120, legend="Normal Distribution", col="blue", lty=1, cex=0.8)
969
970 #Figure B.4
971 gamma_train_qb2 <- quantile_bin(data_test2, sm, train_gamma2$par)
972 df_test_gamma2 <- ci(gamma_train_qb2, data_test2, 0.05)
973 ci_bin(df_test_gamma2, tau2)
```

```
974
975 #Figure B.4
976 normal_train_qb2 <- quantile_bin(data_test2, sm, train_normal2$par)
977 df_test_normal2 <- ci(normal_train_qb2, data_test2, 0.05)
978 ci_bin(df_test_normal2, tau2)
979
980
981 l_gamma2
982 l_normal2
983
984 best_hp_gamma2
985 best_hp_normal2
986
987 #train_gamma$par is the semi-parametric model coefficients for gamma
    assumption
988 ci(quantile_bin(data_single_test2, sm, train_gamma2$par), data_single_test2
    , 0.05)
989 #train_normal$par is the semi-parametric model coefficients for normal
    assumption
990 ci(quantile_bin(data_single_test2, sm, train_normal2$par),
    data_single_test2, 0.05)
991
992 pinball_gamma2 <- loss_pinball(tau2, X_test2, y_test2, train_gamma2$par) /
    length(y_test2)
993 pinball_gamma2
994 pinball_normal2 <- loss_pinball(tau2, X_test2, y_test2, train_normal2$par)
    / length(y_test2)
995 pinball_normal2
996
997 min(cv_gamma2)
998 min(cv_normal2)
999 #------------------------------------------------------------------------
1000
1001 #Simulation for 0.80-th semi-parametric quantile regression model
1002
1003 tau3 <- 0.8
1004 y <- data$y
1005 x <- data$x
1006 sm <- smoothCon(s(x, k = 20, bs = "cr"), data=data, knots=NULL)[[1]]
1007 X <- sm$X
1008 hp_can3 <- c(5*1e-5, 1e-4, 5*1e-4, 1e-3, 5*1e-3, 1e-2, 5e-2, 1e-1)
1009 loop <- seq(1, dim(X)[1], floor(dim(X)[1]/5))
1010 loop <- append(loop, dim(X)[1])
1011
1012 cv_gamma3 <- cvFive_gamma(X, y, tau3, fit$coefficients, hp_can3)
1013 cv_normal3 <- cvFive_normal(X, y, tau3, fit$coefficients, hp_can3)
1014 total_cv3 <- c(cv_gamma3, cv_normal3)
1015
1016 #Figure B.5
1017 plot(log(hp_can3), cv_normal3, type = "b", xlab = "log(gamma)", ylab = "CV
    ", col="red", ylim = c(min(total_cv3), max(total_cv3)))
1018 lines(log(hp_can3), cv_gamma3, type = "b", col="blue")
1019 legend(-10, 5.9, legend=c("Normal Distribution", "Gamma Distribution"),
1020        col=c("red", "blue"), lty=1:1, cex=0.8)
1021
1022 best_hp_gamma3 <- hp_can3[which.min(cv_gamma3)]
1023 best_hp_normal3 <- hp_can3[which.min(cv_normal3)]
1024
1025 set.seed(3)
1026 train3 <- sample(1:dim(data)[1], size = 0.7 * length(x), replace = FALSE)
1027
1028 data_train3 <- data[train3,]
1029 data_test3 <- data[-train3,]
```

```
data_single_test3 <- data_single[-train3,]

x_train3 <- data_train3$x
y_train3 <- data_train3$y

x_test3 <- data_test3$x
y_test3 <- data_test3$y

sm <- smoothCon(s(x, k = 20, bs = "cr"), data=data_train3, knots=NULL)[[1]]
X_train3 <- sm$X

l_gamma3 <- lambda_gamma(tau3, y_train3)
l_normal3 <- lambda_normal(tau3, y_train3)

train_gamma3 <- optim(par = fit$coefficients, penalised_negllkFun, gr =
    penalised_negGrad, method = "BFGS", tau = tau3,
                      lam = l_gamma3, X = X_train3, y = y_train3, k =
    best_hp_gamma3)
train_normal3 <- optim(par = fit$coefficients, penalised_negllkFun, gr =
    penalised_negGrad, method = "BFGS", tau =tau3,
                       lam = l_normal3, X = X_train3, y = y_train3, k =
    best_hp_normal3)

#Figure B.5
X_test3 <- PredictMat(sm, data.frame(x=x_test3))
plot(x_test3[order(x_test3)], y_test3[order(x_test3)], xlab="Temperature",
    ylab="O3", col = "grey", ylim=c(-2, 140))
#plotting the quantile estimate
lines(x_test3[order(x_test3)], X_test3[order(x_test3),]%*%train_gamma3$par,
     type = 'l', col = "red", lwd=2)
legend(-5, 120, legend="Gamma Distribution", col="red", lty=1, cex=0.8)

plot(x_test3[order(x_test3)], y_test3[order(x_test3)], xlab="Temperature",
    ylab="O3", col = "grey", ylim=c(-2, 140))
lines(x_test3[order(x_test3)], X_test3[order(x_test3),]%*%train_normal3$par
    , type = 'l', col = "blue", lwd=2)
legend(-5, 120, legend="Normal Distribution", col="blue", lty=1, cex=0.8)

#Figure B.6
gamma_train_qb3 <- quantile_bin(data_test3, sm, train_gamma3$par)
df_test_gamma3 <- ci(gamma_train_qb3, data_test3, 0.05)
ci_bin(df_test_gamma3, tau3)

#Figure B.6
normal_train_qb3 <- quantile_bin(data_test3, sm, train_normal3$par)
df_test_normal3 <- ci(normal_train_qb3, data_test3, 0.05)
ci_bin(df_test_normal3, tau3)


l_gamma3
l_normal3

best_hp_gamma3
best_hp_normal3

#train_gamma$par is the semi-parametric model coefficients for gamma
    assumption
ci(quantile_bin(data_single_test3, sm, train_gamma3$par), data_single_test3
    , 0.05)
#train_normal$par is the semi-parametric model coefficients for normal
    assumption
ci(quantile_bin(data_single_test3, sm, train_normal3$par),
    data_single_test3, 0.05)
```

```
1081
1082 pinball_gamma3 <- loss_pinball(tau3, X_test3, y_test3, train_gamma3$par) /
     length(y_test3)
1083 pinball_gamma3
1084 pinball_normal3 <- loss_pinball(tau3, X_test3, y_test3, train_normal3$par)
     / length(y_test3)
1085 pinball_normal3
1086
1087 min(cv_gamma3)
1088 min(cv_normal3)
1089
1090 #-------------------------------------------------------------------------
1091
1092 #Simulation for 0.75-th semi-parametric quantile regression model
1093
1094 tau4 <- 0.75
1095 y <- data$y
1096 x <- data$x
1097 sm <- smoothCon(s(x, k = 20, bs = "cr"), data=data, knots=NULL)[[1]]
1098 X <- sm$X
1099 hp_can4 <- c(5*1e-5, 1e-4, 5*1e-4, 1e-3, 5*1e-3, 1e-2, 5e-2, 1e-1)
1100 loop <- seq(1, dim(X)[1], floor(dim(X)[1]/5))
1101 loop <- append(loop, dim(X)[1])
1102
1103 cv_gamma4 <- cvFive_gamma(X, y, tau4, fit$coefficients, hp_can4)
1104 cv_normal4 <- cvFive_normal(X, y, tau4, fit$coefficients, hp_can4)
1105 total_cv4 <- c(cv_gamma4, cv_normal4)
1106
1107 #Figure B.7
1108 plot(log(hp_can4), cv_normal4, type = "b", xlab = "log(gamma)", ylab = "CV
     ", col="red", ylim = c(min(total_cv4), max(total_cv4)))
1109 lines(log(hp_can4), cv_gamma4, type = "b", col="blue")
1110 legend(-10, 6.70, legend=c("Normal Distribution", "Gamma Distribution"),
1111        col=c("red", "blue"), lty=1:1, cex=0.8)
1112
1113 best_hp_gamma4 <- hp_can4[which.min(cv_gamma4)]
1114 best_hp_normal4 <- hp_can4[which.min(cv_normal4)]
1115
1116 set.seed(4)
1117 train4 <- sample(1:dim(data)[1], size = 0.7 * length(x), replace = FALSE)
1118
1119 data_train4 <- data[train4,]
1120 data_test4 <- data[-train4,]
1121 data_single_test4 <- data_single[-train4,]
1122
1123 x_train4 <- data_train4$x
1124 y_train4 <- data_train4$y
1125
1126 x_test4 <- data_test4$x
1127 y_test4 <- data_test4$y
1128
1129 sm <- smoothCon(s(x, k = 20, bs = "cr"), data=data_train4, knots=NULL)[[1]]
1130 X_train4 <- sm$X
1131
1132 l_gamma4 <- lambda_gamma(tau4, y_train4)
1133 l_normal4 <- lambda_normal(tau4, y_train4)
1134
1135 train_gamma4 <- optim(par = fit$coefficients, penalised_negllkFun, gr =
     penalised_negGrad, method = "BFGS", tau = tau4,
1136                       lam = l_gamma4, X = X_train4, y = y_train4, k =
     best_hp_gamma4)
1137 train_normal4 <- optim(par = fit$coefficients, penalised_negllkFun, gr =
     penalised_negGrad, method = "BFGS", tau =tau4,
```

```
                        lam = l_normal4, X = X_train4, y = y_train4, k =
    best_hp_normal4)

#Figure B.7
X_test4 <- PredictMat(sm, data.frame(x=x_test4))
plot(x_test4[order(x_test4)], y_test4[order(x_test4)], xlab="Temperature",
    ylab="O3", col = "grey", ylim=c(-2, 140))
#plotting the quantile estimate
lines(x_test4[order(x_test4)], X_test4[order(x_test4),]%*%train_gamma4$par,
     type = 'l', col = "red", lwd=2)
legend(-5, 120, legend="Gamma Distribution", col="red", lty=1, cex=0.8)

plot(x_test4[order(x_test4)], y_test4[order(x_test4)], xlab="Temperature",
    ylab="O3", col = "grey", ylim=c(-2, 140))
lines(x_test4[order(x_test4)], X_test4[order(x_test4),]%*%train_normal4$par
    , type = 'l', col = "blue", lwd=2)
legend(-5, 120, legend="Normal Distribution", col="blue", lty=1, cex=0.8)

#Figure B.8
gamma_train_qb4 <- quantile_bin(data_test4, sm, train_gamma4$par)
df_test_gamma4 <- ci(gamma_train_qb4, data_test4, 0.05)
ci_bin(df_test_gamma4, tau4)

#Figure B.8
normal_train_qb4 <- quantile_bin(data_test4, sm, train_normal4$par)
df_test_normal4 <- ci(normal_train_qb4, data_test4, 0.05)
ci_bin(df_test_normal4, tau4)


l_gamma4
l_normal4

best_hp_gamma4
best_hp_normal4

#train_gamma$par is the semi-parametric model coefficients for gamma
    assumption
ci(quantile_bin(data_single_test4, sm, train_gamma4$par), data_single_test4
    , 0.05)
#train_normal$par is the semi-parametric model coefficients for normal
    assumption
ci(quantile_bin(data_single_test4, sm, train_normal4$par),
    data_single_test4, 0.05)

pinball_gamma4 <- loss_pinball(tau4, X_test4, y_test4, train_gamma4$par) /
    length(y_test4)
pinball_gamma4
pinball_normal4 <- loss_pinball(tau4, X_test4, y_test4, train_normal4$par)
    / length(y_test4)
pinball_normal4

min(cv_gamma4)
min(cv_normal4)
```

Appendix B

Semi-Parametric Quantile Regression Examples

0.90-th Semi-Parametric Quantile Regression Model Examples

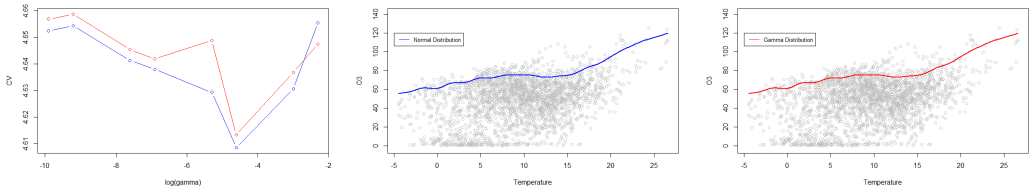


Figure B.1: (*left*) Plot of cross validation score against $\log(\gamma)$ for gamma distribution assumption (red) and normal distribution assumption (blue) for weight parameter $\hat{\lambda}^*$. (*middle*) Plot of semi-parametric 0.90-th quantile regression using weight parameter $\hat{\lambda}^*_{normal} = 0.89991$ with tuning hyperparameter $\gamma^* = 0.01$. (*right*) Plot of semi-parametric 0.90-th quantile regression using weight parameter $\hat{\lambda}^*_{gamma} = 0.96332$ and tuning hyperparameter $\gamma^* = 0.01$.

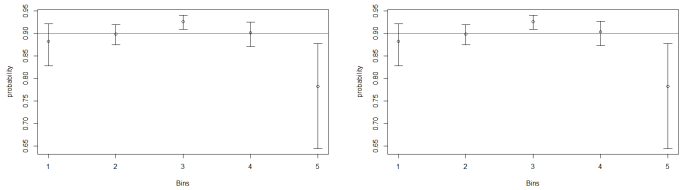


Figure B.2: (*left*) Hypothesis test for semi-parametric 0.90-th quantile regression using weight parameter $\hat{\lambda}^*_{normal} = 0.89991$ with tuning hyperparameter $\gamma^* = 0.01$. (*right*) Hypothesis test for semi-parametric 0.90-th quantile regression using weight parameter $\hat{\lambda}^*_{gamma} = 0.96332$ and tuning hyperparameter $\gamma^* = 0.01$.

	$\hat{\lambda}^*_{normal} = 0.89991$	$\hat{\lambda}^*_{gamma} = 0.96332$
$\hat{p}_{mle}$	0.90779	0.90820
CI_{lower}	0.89569	0.89612
CI_{upper}	0.91861	0.91900
$\sum \rho_{0.95}(y - \hat{\mu}_{0.95}(x))$	3.1140	3.1143
$\min\{CV(K = 5; \gamma)\}$	4.6133	4.6084

Table B.1: Statistics $\hat{p}_{mle}$, lower boundary and upper boundary of 0.95 Wilson score interval, total pinball loss values and minimum of cross validation score values for 0.90-th semi-parametric model with $\hat{\lambda}^*_{normal} = 0.89991$ and $\hat{\lambda}^*_{gamma} = 0.96332$.

0.85-th Semi-Parametric Quantile Regression Model

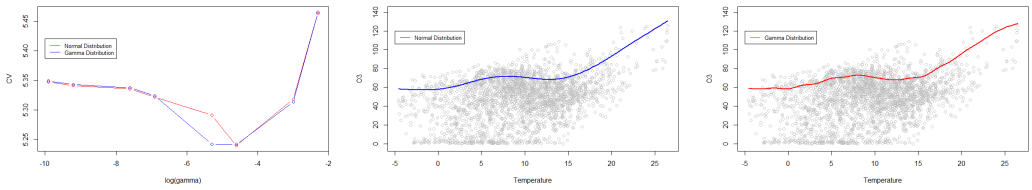


Figure B.3: (*left*) Plot of cross validation score against $\log(\gamma)$ for gamma distribution assumption (red) and normal distribution assumption (blue) for weight parameter $\hat{\lambda}^*$. (*middle*) Plot of semi-parametric 0.85-th quantile regression using weight parameter $\hat{\lambda}^*_{normal} = 0.93594$ with tuning hyperparameter $\gamma^* = 0.01$. (*right*) Plot of semi-parametric 0.85-th quantile regression using weight parameter $\hat{\lambda}^*_{gamma} = 0.91433$ and tuning hyperparameter $\gamma^* = 0.01$.

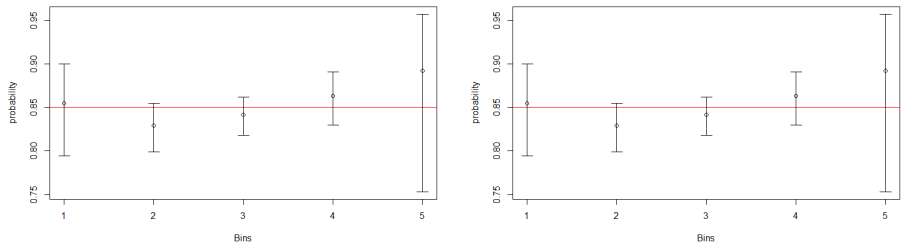


Figure B.4: (*left*) Hypothesis test for semi-parametric 0.80-th quantile regression using weight parameter $\hat{\lambda}^*_{normal} = 0.93594$ with tuning hyperparameter $\gamma^* = 0.01$. (*right*) Hypothesis test semi-parametric 0.80-th quantile regression using weight parameter $\hat{\lambda}^*_{gamma} = 0.91433$ and tuning hyperparameter $\gamma^* = 0.01$.

	$\hat{\lambda}^*_{normal} = 0.93594$	$\hat{\lambda}^*_{gamma} = 0.91433$
$\hat{p}_{mle}$	0.84251	0.84210
CI_{lower}	0.82755	0.82713
CI_{upper}	0.85639	0.85600
$\sum \rho_{0.95}(y - \hat{\mu}_{0.95}(x))$	4.3164	4.3164
$\min\{CV(K = 5; \gamma)\}$	5.2400	5.2421

Table B.2: Statistics $\hat{p}_{mle}$, lower boundary and upper boundary of 0.95 Wilson score interval, total pinball loss values and minimum of cross validation score values for 0.85-th semi-parametric model with $\hat{\lambda}^*_{normal} = 0.93594$ and $\hat{\lambda}^*_{gamma} = 0.91433$.

0.80-th Semi-Parametric Quantile Regression Model

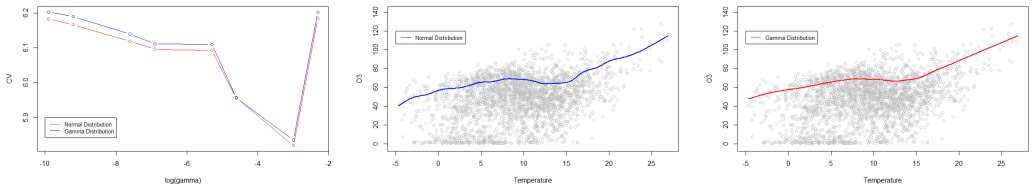


Figure B.5: (*left*) Plot of cross validation score against $\log(\gamma)$ for gamma distribution assumption (red) and normal distribution assumption (blue) for weight parameter $\hat{\lambda}^*$. (*middle*) Plot of semi-parametric 0.80-th quantile regression using weight parameter $\hat{\lambda}^*_{normal} = 1.0137$ with tuning hyperparameter $\gamma^* = 0.05$. (*right*) Plot of semi-parametric 0.80-th quantile regression using weight parameter $\hat{\lambda}^*_{gamma} = 0.90369$ and tuning hyperparameter $\gamma^* = 0.05$.

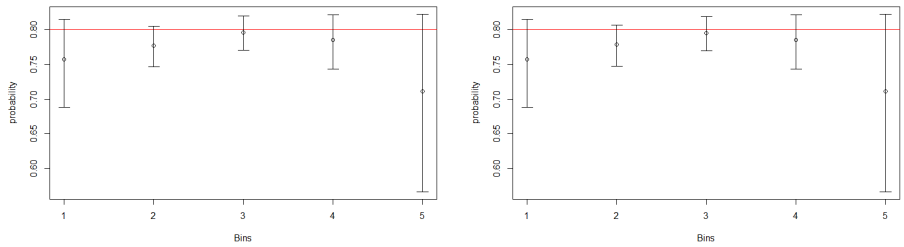


Figure B.6: (*left*) Hypothesis test for semi-parametric 0.80-th quantile regression using weight parameter $\hat{\lambda}^*_{normal} = 1.0137$ with tuning hyperparameter $\gamma^* = 0.05$. (*right*) Hypothesis test for semi-parametric 0.80-th quantile regression using weight parameter $\hat{\lambda}^*_{gamma} = 0.90369$ (red) and tuning hyperparameter $\gamma^* = 0.05$.

	$\hat{\lambda}^*_{normal} = 1.0137$	$\hat{\lambda}^*_{gamma} = 0.90369$
$\hat{p}_{mle}$	0.78416	0.78416
CI_{lower}	0.76744	0.76744
CI_{upper}	0.80001	0.80001
$\sum \rho_{0.95}(y - \hat{\mu}_{0.95}(x))$	5.4706	5.4705
$\min\{CV(K = 5; \gamma)\}$	5.8192	5.8356

Table B.3: Statistics $\hat{p}_{mle}$, lower boundary and upper boundary of 0.95 Wilson score interval, total pinball loss values and minimum of cross validation score values for 0.80-th semi-parametric model with $\hat{\lambda}^*_{normal} = 1.0137$ and $\hat{\lambda}^*_{gamma} = 0.90369$.

0.75-th Semi-Parametric Quantile Regression Model

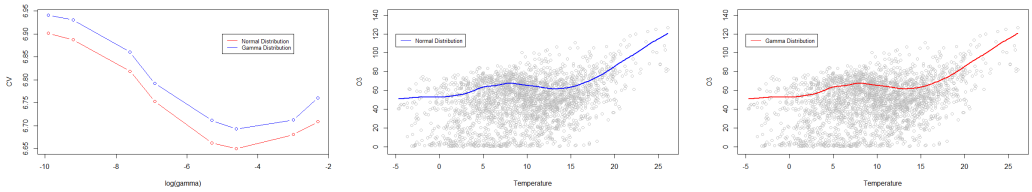


Figure B.7: (*left*) Plot of cross validation score against $\log(\gamma)$ for gamma distribution assumption (red) and normal distribution assumption (blue) for weight parameter $\hat{\lambda}^*$. (*middle*) Plot of semi-parametric 0.75-th quantile regression using weight parameter $\hat{\lambda}^*_{normal} = 1.1177$ with tuning hyperparameter $\gamma^* = 0.01$. (*right*) Plot of semi-parametric 0.75-th quantile regression using weight parameter $\hat{\lambda}^*_{gamma} = 0.90831$ and tuning hyperparameter $\gamma^* = 0.01$.

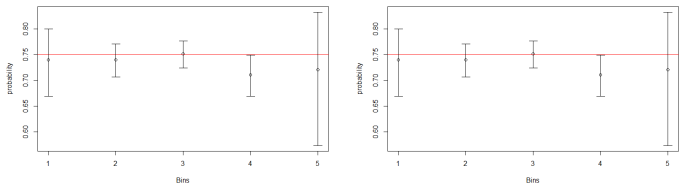


Figure B.8: (*left*) Hypothesis test for semi-parametric 0.75-th quantile regression using weight parameter $\hat{\lambda}^*_{normal} = 1.1177$ with tuning hyperparameter $\gamma^* = 0.01$. (*right*) Hypothesis test for semi-parametric 0.75-th quantile regression using weight parameter $\hat{\lambda}^*_{gamma} = 0.90831$ and tuning hyperparameter $\gamma^* = 0.01$.

	$\hat{\lambda}^*_{normal} = 1.1177$	$\hat{\lambda}^*_{gamma} = 0.90831$
$\hat{p}_{mle}$	0.73847	0.73847
CI_{lower}	0.72071	0.72071
CI_{upper}	0.75548	0.75548
$\sum \rho_{0.95}(y - \hat{\mu}_{0.95}(x))$	6.3974	6.3978
$\min\{CV(K = 5; \gamma)\}$	6.6499	6.6926

Table B.4: Statistics $\hat{p}_{mle}$, lower boundary and upper boundary of 0.95 Wilson score interval, total pinball loss values and minimum of cross validation score values for 0.5-th semi-parametric model with $\hat{\lambda}^*_{normal} = 1.1177$ and $\hat{\lambda}^*_{gamma} = 0.90831$.