Extreme Quantile Regression: Estimation and Model Selection

by

Heidi Barriault

Thesis submitted to the
University of Ottawa
In partial fulfillment of the requirements
For the M.Sc. degree in
Mathematics and Statistics[1]

Department of Mathematics and Statistics
Faculty of Science
University of Ottawa

[1]The M.Sc. program is a joint program with Carleton University, administered by the Ottawa-Carleton Institute of Mathematics and Statistics.

Abstract

Extreme quantile regression is a statistical method that focuses on estimating conditional quantiles in the tails of a distribution. Solving such problems requires tools from statistics, extreme value theory, optimization and is particularly relevant in high-dimensional settings, such as finance and health care. The objective of this thesis is to develop and evaluate methodologies for extreme quantile regression in high-dimensional contexts. To achieve this, we first review existing methods for solving penalized linear regression and quantile regression problems and then build on this foundation to extend penalization to extreme quantile regression.

Acknowledgments

I would like to express my sincere gratitude to Dr. Rafal Kulik for his guidance, support, and constructive feedback throughout the course of this project.

I am also deeply grateful to my parents, Julie and Louis, to my siblings, Daniel, William, Kari, and Emily, and to my grandparents, for their unwavering support and encouragement. Finally, I thank my friends for their continued motivation and understanding during the completion of this project.

This project would not have been possible without your contributions and support.

Contents

Abstract . . . ii
Acknowledgments . . . iii

1 Introduction **3**
1.1 Background . . . 3
1.2 Objectives . . . 3
1.3 Structure of the Thesis. . . . 4
1.4 Author's contribution . . . 5

2 Quantile Regression **7**
2.1 Theoretical and sample quantiles . . . 7
2.2 Conditional quantiles . . . 9
2.2.1 Inference . . . 12

3 Optimization problem **15**
3.1 Lagrange multipliers . . . 16
3.1.1 Lagrange multipliers and subgradients . . . 18
3.2 Regression problem . . . 19
3.2.1 Regression with ridge . . . 20
3.2.2 Regression with LASSO . . . 24
3.3 Quantile regression . . . 28
3.3.1 Linear programming for quantile regression . . . 30
3.3.2 Quantile regression and ridge . . . 31
3.3.3 Quantile regression and LASSO . . . 34

4 Extreme Quantile Regression **39**
4.1 Regular Variation . . . 40
4.2 Estimation of extreme unconditional quantiles . . . 42
4.2.1 Estimation of the tail index . . . 43
4.3 Estimation of extreme conditional quantiles using penalized likelihood . . . 47
4.3.1 Penalized likelihood with ridge penalty . . . 50
4.3.2 Penalized likelihood with LASSO penalty . . . 52
4.3.3 Numerical considerations . . . 55

4.4 Estimation of extreme conditional quantiles using extremal quantile regression . . . 59

5 Real data examples **69**
5.1 Experiment 1: Linear regression . . . 70
5.2 Experiment 2: Linear regression with LASSO penalty . . . 70
5.3 Experiment 3: Quantile regression . . . 70
5.4 Experiment 4: Quantile regression with LASSO . . . 71
5.5 Experiment 5: Quantile boxplots . . . 72
5.6 Experiment 6: Quantile boxplots for a modified NFL dataset . . . 72
5.7 Summary . . . 72

6 Summary and future directions **75**
6.1 Summary of results . . . 75
6.2 Contributions . . . 75
6.3 Strengths and limitations . . . 76
6.3.1 Limitations . . . 76
6.3.2 Future directions . . . 76

7 R codes **77**
7.1 Codes for Chapter 2 . . . 77
7.1.1 Example 2.1.4 . . . 77
7.2 Codes for Chapter 3 . . . 78
7.2.1 Example 3.2.3 . . . 78
7.2.2 Example 3.3.1 . . . 79
7.2.3 Example 3.3.2 . . . 79
7.3 Codes for Chapter 4 . . . 80
7.3.1 Example 4.3.1 . . . 80
7.3.2 Example 4.3.2 . . . 82
7.3.3 Example 4.3.3 . . . 83
7.3.4 Example 4.3.4 . . . 85
7.3.5 Example 4.3.5 . . . 87
7.4 Codes for Chapter 5 . . . 89
7.4.1 Section 5.1 . . . 89
7.4.2 Section 5.2 . . . 90
7.4.3 Section 5.3 . . . 90
7.4.4 Section 5.4 . . . 90
7.4.5 Section 5.5 . . . 91

Chapter 1

Introduction

1.1 Background

Quantile regression is a robust method that models the relationship between predictors and quantiles of an outcome. The foundational book by Koenker [5] demonstrates how quantile regression can be applied in various fields such as health care and finance, highlighting its benefits and providing solutions. Important applications of quantile regression often involve extreme quantiles, such as the 99^{th} percentile of infant birthweights. Extreme quantile regression, developed by Chernozhukov [4], focuses on the estimation of conditional quantiles in the tails of a distribution using extreme value theory. This approach is particularly relevant in real-world applications that involve rare events or outlier behaviour, including finance, climate studies, and health care.

Extreme quantile regression is a method that is commonly used in fields that involve high-dimensional data. As such, it requires optimization techniques to be able to perform meaningful analyses. However, the current methodologies for extreme quantile regression do not extend well to high-dimensional cases. Thus, this emphasizes the need for new methodologies that can be used in multivariate, possibly high-dimensional settings.

1.2 Objectives

The primary objective of this thesis is to develop and evaluate new methodologies for multivariate, possibly high-dimensional extreme quantile regression. For this, we provide an extensive review and assessment of existing methods for solving quantile regression problems and explore how optimization techniques can be applied to multivariate data cases using quantiles. It is important to note that in this thesis we do not make any assumptions on the predictors of the models used. There are however some specific assumptions on the response variable. The response variable has to be heavy tailed (to be specified later). Also, we do not really consider a truly high-dimensional case, since the dimension p of the predictor is fixed, it does not grow to infinity.

1.3 Structure of the Thesis.

Chapter 2 This chapter explores the theoretical and foundational aspects of quantile regression. We begin by studying the theoretical **quantile function** and sample quantile function. This is followed by **conditional quantiles** and how to apply them in the context of regression. Finally, we conclude with inference for sample quantiles. This chapter lays the groundwork for the thesis by introducing key concepts that form the basis of **quantile regression**. These foundational principles will be further expanded upon in the subsequent chapters. The work that is shown in this chapter is based on the book written by Koenker, [5].

Chapter 3. In order to deal with high dimensional predictors and building on our foundations of quantile regression, we incorporate **regularization techniques** such as ridge and LASSO penalties. These regularization techniques require the application of different **optimization methods**. Thus, first, we establish the necessary theoretical groundwork for optimization. We begin by introducing **Lagrange multipliers**, a common tool for solving optimization problems. However, as the complexity increases significantly when dealing with non-differentiable loss or penalty function, we explore alternative approaches, such as **subgradients**. The subgradient technique is combined with the **coordinate descent** in order to do optimization in dimensions larger than 1. However, although such an approach works well in the case of linear or logistic regression, it fails in the case of (penalized) quantile regression. Thus, we discuss **linear programming**.

In order to explain everything in detail, we first introduce a standard linear regression problem with ridge and LASSO penalization. In particular, we demonstrate how ridge regression can be solved using Lagrange multipliers and coordinate descent. On the other hand, for LASSO, the non-differentiability of the penalty function necessitates the use of subgradients along with the coordinate descent.

Then, we address quantile regression, which has a non-differentiable loss function. We investigate the feasibility of using the coordinate descent method for quantile regression with ridge or LASSO penalties, the same way we did for the penalized linear regression. We demonstrate that this approach fails, requiring a different optimization method. As such, we show that the optimization problem for quantile regression (without regularization) can be formulated and solved as a linear programming problem. Extending this framework, we reformulate the ridge-penalized quantile regression problem as a quadratic program, while the LASSO-penalized case results in a more complex version of the original quantile regression linear program.

The material for non-penalized quantile regression is classical and is based mostly on the book by Koenker, [5], with an excellent explanation given in [2]. The idea to transform quantile regression with penalties to a linear program came from [9] and [7].

Chapter 4. In this chapter, we integrate quantile regression and optimization techniques into an **extreme value** setting. Before we can apply ridge and LASSO penalization to **extreme quantile regression**, we must understand some fundamentals of extreme value theory and methods for estimating extreme quantiles. As such we begin this chapter with an introduction to **regular variation**, a classical and crucial concept for extreme value theory. This is based on [6]. Following this introduction, we present a method to estimate **extreme unconditional quantiles**. This method, inspired by [10], utilizes the Hill estimator of the **tail index**.

The first estimation method for extreme conditional quantiles assumes a **parametric form**. This allows us to calculate the **log-likelihood**, with a penalty. This likelihood is approximated by a quadratic function. As such, the optimal coefficients can be obtained by maximizing a quadratic function, with some challenges arising in the case of LASSO penalty. This is combined with a coordinate descent.

The second estimation method for extreme conditional quantiles assumes a **semi-parametric** form, incorporating a regularly varying behaviour. This approach originates from [4]. In particular, we consider the penalized regression under the **common slope** assumption. This approach stems from [10].

Chapter 5. In this chapter we analyze a dataset involving NFL statistics. We apply different regression and quantile regression methods to understand the relationship between different aspects of the game on the total number of points scored by a specific team.

Chapter 7. The last chapter contains all the codes.

1.4 Author's contribution

Most of the theoretical and methodological foundations presented in the thesis stem from the aforementioned literature. However:

- All the theoretical examples were fully developed by the author;
- All the numerical studies were developed by the author;
- The specific implementation of the penalized quantile regression, based on linear programming, was developed by the author;
- The entire methodology of both the penalized likelihood approach and the penalized common slope method for extreme conditional quantiles were developed by the author, correcting some issues present in the literature.

Chapter 2

Quantile Regression

In this chapter, we explore the theoretical and foundational aspects of quantile regression. We begin by studying the theoretical quantile function and sample quantile function, see Section 2.1. This is followed by conditional quantiles and how to apply them in the context of regression, see Section 2.2. Finally, we conclude with inference involving sample quantiles, see Section 2.2.1. This chapter lays the groundwork for the thesis by introducing key concepts that form the basis of quantile regression. These foundational principles will be further expanded in the subsequent chapters.

The work presented in this chapter is based on the book by Koenker,[5]. Some examples were written by the author of the thesis.

2.1 Theoretical and sample quantiles

To do quantile regression, we must first break down its different aspects. Quantile regression uses a more complex version of the quantile function of a random variable. Assume Y is a random variable and $F_Y(y) = \mathbb{P}(Y \leq y)$. We introduce the following notions.

Definition 2.1.1 (Quantile function of random variable)**.** *Let* $\tau \in [0, 1]$. *Define*

$$Q_Y(\tau) = F_Y^{\leftarrow}(\tau) = \inf\{y : F_Y(y) \geq \tau\} . \tag{2.1}$$

Note that if a random variable Y has an infinite support (e.g. the normal distribution), then $Q_Y(0) = -\infty$, $Q_Y(1) = +\infty$. When clear, we will simply use Q instead of Q_Y to refer to the quantile function.

Definition 2.1.2 (Loss function)**.** *Let* $x \in \mathbb{R}$ *and* $\tau \in [0, 1]$. *Define*

$$\rho_\tau(x) = x(\tau - \mathbb{1}(x < 0)) . \tag{2.2}$$

If we are able to find an analytical formula for the quantile function, we can calculate the desired quantiles. Alternatively, to find a specific quantile, we can minimize the expected loss function applied to $Y - u$ with respect to u. We note that

$$\begin{aligned} Q_Y(\tau) &= \text{argmin}_u\{\mathbb{E}[\rho_\tau(Y-u)]\} \\ &= \text{argmin}_u\{(\tau-1)\mathbb{E}[(Y-u)\mathbb{1}(Y-u<0)] + \tau\mathbb{E}[(Y-u)\mathbb{1}(Y-u\geq 0)]\}\ . \end{aligned} \tag{2.3}$$

Now, we are going to consider empirical versions of Q_Y, stemming from both (2.1) and (2.3). For a random sample Y_i, $i = 1, \ldots, n$, coming from the same distribution as Y, we can define the empirical version of the distribution function F_Y, which is referred to as the empirical cumulative distribution function (ECDF) and is given by

$$\widehat{F}_Y(y) = \frac{1}{n}\sum_{i=1}^{n} \mathbb{1}\{Y_i \leq y\}.$$

Definition 2.1.3 (Sample quantile function)**.**

$$\widehat{Q}_Y(\tau) = \widehat{F}_Y^{\leftarrow}(y) = \inf\{y : \widehat{F}_Y(y) \geq \tau\}, \quad \tau \in (0,1). \tag{2.4}$$

In particular, if $\tau \in ((k-1)/n, k/n]$, then $\widehat{Q}_Y(\tau) = Y_{(k)}$, where $Y_{(k)}$ are the order statistics corresponding to the random sample $Y_1, \ldots, Y_n$:

$$Y_{(1)} \leq Y_{(2)} \leq \cdots \leq Y_{(n)}.$$

Alternatively, sample quantiles are the empirical minimizers of

$$\widetilde{Q}_Y(\tau) := \text{argmin}_u \sum_{i=1}^{n} \rho_\tau(Y_i - u) = \text{argmin}_u \left\{ (\tau-1)\sum_{Y_i<u}(Y_i-u) + \tau\sum_{Y_i\geq u}(Y_i-u) \right\}\ . \tag{2.5}$$

Note that it is not obvious to see that the empirical quantiles obtained from (2.4) are the same as those obtained from (2.5). This is illustrated in the next example.

Example 2.1.4. In this example, we simulated a univariate dataset of size $n = 100$ and calculated sample medians using (2.4) and (2.5). We repeated it $N = 1000$ times. The results are displayed in Figure 2.1. We note that the obtained sample quantiles are almost identical.

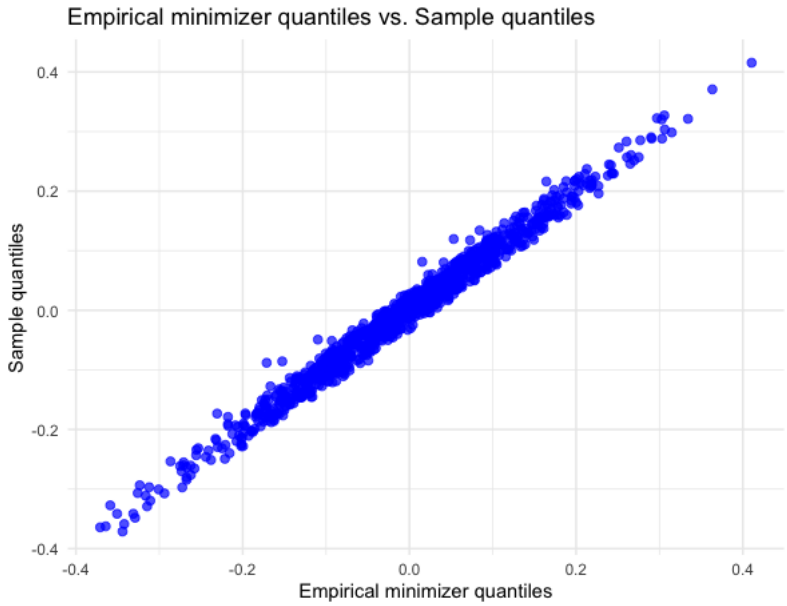

Figure 2.1: Quantile comparison.

2.2 Conditional quantiles

Assume now that we want to model a relationship between two variables X and Y. Consider the conditional distribution function

$$F_{Y|X}(y \mid x) = \mathbb{P}(Y \leq y \mid X = x).$$

In the spirit of Equation (2.1), we can define the conditional quantile as follows:

Definition 2.2.1 (Conditional quantile function)**.** *Let $\tau \in [0, 1]$. Define*

$$Q_{Y|X}(\tau \mid x) = F^{\leftarrow}_{Y|X}(\tau \mid x) = \inf\{y : F_{Y|X}(y \mid x) \geq \tau\} \, . \tag{2.6}$$

In particular, we can consider the conditional quantile taking a specific linear form:

Definition 2.2.2 (Linear conditional quantile)**.** *Let $x \in \mathbb{R}$ and $\tau \in (0, 1)$. For $\beta_0, \beta_1 \in \mathbb{R}$ consider the model*

$$Q_{Y|X}(\tau \mid x) = Q_\varepsilon(\tau) + \beta_0 + \beta_1 x \, , \tag{2.7}$$

where Q_ε is a quantile function of an auxiliary random variable ε.

This form of the conditional quantile is very general and there are several scenarios that lead to such expression. Note also that the parameters β_0, β_1 may be different depending on τ. So, formally, $\beta_0 = \beta_0(\tau)$, $\beta_1 = \beta_1(\tau)$.

For example, consider the linear quantile regression

$$Y_i = \widetilde{\beta}_0 + \widetilde{\beta}_1 X_i + \varepsilon_i \ ,$$

where ε_i are i.i.d. random variables with the quantile function Q_ε. Then

$$\begin{aligned} F_{Y|X}(y \mid x) &= \mathbb{P}(Y \le y \mid X = x) \\ &= \mathbb{P}(\widetilde{\beta}_0 + \widetilde{\beta}_1 X + \varepsilon \le y \mid X = x) \\ &= \mathbb{P}(\widetilde{\beta}_0 + \widetilde{\beta}_1 x + \varepsilon \le y) \\ &= F_\varepsilon(y - \widetilde{\beta}_0 - \widetilde{\beta}_1 x) \ . \end{aligned}$$

Hence

$$Q_\varepsilon(\tau) = y - \widetilde{\beta}_0 - \widetilde{\beta}_1 x$$

which leads to the following formula for the conditional quantiles:

$$y = Q_\varepsilon(\tau) + \widetilde{\beta}_0 + \widetilde{\beta}_1 x \ .$$

These conditional quantiles are of the form (2.7) where $\beta_0(\tau) = Q_\varepsilon(\tau) + \widetilde{\beta}_0$ and $\beta_1 = \widetilde{\beta}_1$.

Note: While traditional regression methods estimate $\widetilde{\beta}_0$, the parameter of the original regression model, quantile regression methods estimate β_0, the parameter of the quantile regression model.

We will go over two other scenarios that lead to the expression in (2.7). Consider the following quantile regression model,

$$Y_i = \widetilde{\beta}_0 + \widetilde{\beta}_1 \log X_i + \varepsilon_i$$

where ε_i are i.i.d. random variables with the quantile function Q_ε. Then

$$\begin{aligned} F_{Y|X}(y \mid x) &= \mathbb{P}(Y \le y \mid X = x) \\ &= \mathbb{P}(\widetilde{\beta}_0 + \widetilde{\beta}_1 \log X + \varepsilon \le y \mid X = x) \\ &= \mathbb{P}(\widetilde{\beta}_0 + \widetilde{\beta}_1 \log x + \varepsilon \le y) \\ &= \mathbb{P}(\varepsilon \le y - \widetilde{\beta}_0 - \widetilde{\beta}_1 \log x) \\ &= F_\varepsilon(y - \widetilde{\beta}_0 - \widetilde{\beta}_1 \log x) \ . \end{aligned}$$

Hence

$$Q_\varepsilon(\tau) = y - \widetilde{\beta}_0 - \widetilde{\beta}_1 \log x$$

and therefore we obtain the following formula for the conditional quantiles:

$$y = Q_\varepsilon(\tau) + \widetilde{\beta}_0 + \widetilde{\beta}_1 \log x \ .$$

Now, let's consider the heteroscedastic quantile regression model,

$$Y_i = \widetilde{\beta}_0 + \widetilde{\beta}_1 X_i + \varepsilon_i \sigma(X_i) \ ,$$

where ε_i are i.i.d. random variables with the quantile function Q_ε and $\sigma : \mathbb{R} \to \mathbb{R}_+$. Then

$$\begin{aligned} F_{Y|X}(y \mid x) &= \mathbb{P}(Y \leq y \mid X = x) \\ &= \mathbb{P}(\widetilde{\beta}_0 + \widetilde{\beta}_1 X + \varepsilon\sigma(X) \leq y \mid X = x) \\ &= \mathbb{P}(\widetilde{\beta}_0 + \widetilde{\beta}_1 x + \varepsilon\sigma(x) \leq y) \\ &= \mathbb{P}(\varepsilon\sigma(x) \leq y - \widetilde{\beta}_0 - \widetilde{\beta}_1 x) \\ &= \mathbb{P}\left(\varepsilon \leq \frac{y - \widetilde{\beta}_0 - \widetilde{\beta}_1 x}{\sigma(x)}\right) \\ &= F_\varepsilon\left(\frac{y - \widetilde{\beta}_0 - \widetilde{\beta}_1 x}{\sigma(x)}\right) \ . \end{aligned}$$

Hence

$$Q_\varepsilon(\tau) = \frac{y - \widetilde{\beta}_0 - \widetilde{\beta}_1 x}{\sigma(x)}$$

and therefore we obtain the following formula for the conditional quantiles:

$$y = \widetilde{\beta}_0 + \{Q_\varepsilon(\tau)\sigma(x) + \beta_1 x\} \ .$$

These examples demonstrate that even when the quantile regression model is non-linear, the conditional quantiles can still be linear.

Equation (2.7) is the starting point for the estimation of conditional quantiles. Assume that we have data (X_i, Y_i), $i = 1, \ldots, n$, coming from the model (2.7). Quantile regression, in its most basic form, is a minimization of loss function to find the values of beta. As such the equation (2.5) becomes:

$$\widehat{\beta} = \text{argmin}_\beta \sum_{i=1}^{n} \rho_\tau(Y_i - X_i\beta) = \text{argmin}_\beta \left\{ (\tau - 1) \sum_{Y_i < X_i\beta} (Y_i - X_i\beta) + \tau \sum_{Y \geq X_i\beta} (Y_i - X_i\beta) \right\} \ . \tag{2.8}$$

Where X_i is a vector of size $n \times 1$. To solve such an equation, there are various methods that can be used.

2.2.1 Inference

To be able to do inference using quantile regression, and later using extreme quantile regression, we must break down the inference into three distinct parts: the Central Limit Theorem (CLT), the CLT for Empirical Distribution Function (ECDF), and the CLT for quantiles.

To begin, we will look at the CLT for random variables.

Theorem 2.2.3 (Central Limit Theorem)**.** *Assume that $Y_1, \ldots, Y_n$ are i.i.d random variables with $\mathbb{E}[Y_1] = \mu$ and $Var(Y_1) = \sigma^2$. Let $\overline{Y}_n = \frac{1}{n}\sum_{i=1}^n Y_i$. Then,*

$$\sqrt{n}(\overline{Y}_n - \mu) \xrightarrow{d} \mathcal{N}(0, \sigma^2). \tag{2.9}$$

Proof. To proceed with the proof, let us use the characteristic functions. Recall the properties of the characteristic function:

$$\begin{aligned}
\varphi_Y(t) &= \mathbb{E}[e^{itY}], \varphi_Y(0) = 1, \\
\frac{d}{dt}\varphi_Y(t) &= \mathbb{E}[iYe^{itY}], \frac{d}{dt}\varphi_Y(0) = i\mathbb{E}[Y], \\
\frac{d^2}{dt^2}\varphi_Y(t) &= \mathbb{E}[(iY)^2 e^{itY}] = -\mathbb{E}[Y^2 e^{itY}], \\
\frac{d^2}{dt^2}\varphi_Y(0) &= -\mathbb{E}[Y^2].
\end{aligned}$$

Let $S_n = \sqrt{n}(\overline{Y}_n - \mu)$. Let $\mu = 0$ and so $\mathbb{E}[Y^2] = \sigma^2$. Then,

$$\begin{aligned}
\mathbb{E}[e^{itS_n}] &= \mathbb{E}[e^{it\sqrt{n}(\overline{Y}_n - \mu)}] \\
&= \mathbb{E}[e^{it\sqrt{n}(\frac{1}{n}\sum_{i=1}^n Y_i - \mu)}] = \mathbb{E}[e^{it\frac{1}{\sqrt{n}}\sum_{i=1}^n Y_i}] \\
&= \mathbb{E}[e^{it\frac{1}{\sqrt{n}}Y_1} \cdots e^{it\frac{1}{\sqrt{n}}Y_n}] = \prod_{j=1}^n \mathbb{E}[e^{it\frac{1}{\sqrt{n}}Y_j}], \text{by independence} \\
&= \prod_{j=1}^n \varphi_Y\left(\frac{1}{\sqrt{n}}t\right) = \varphi_Y^n\left(\frac{t}{\sqrt{n}}\right) \\
&= \left(1 + \frac{t}{\sqrt{n}}\varphi'(0) + \frac{t^2}{2n}\varphi''(0)^n + \cdots\right), \text{since } \mu = 0 \\
&= \left(1 + 0 - \frac{t^2\sigma^2}{2n}\right)^n \\
&= \left(1 - \frac{t^2\sigma^2}{2n}\right)^n \xrightarrow{n\to\infty} e^{-\sigma^2 t^2/2}.
\end{aligned}$$

We know that the characteristic function of a standard normal random variable $\mathcal{N}(0,1)$ is $e^{-t^2/2}$ and the limiting characteristic function of S_n is $e^{-\sigma^2 t^2/2}$. So, $\sqrt{n}(\overline{Y}_n - \mu) \xrightarrow{d} \mathcal{N}(0,\sigma^2)$. □

We are going to apply the theorem above to random variables $Y_i = \mathbb{1}\{X_i \leq x\}$. This way, we will obtain a CLT for the empirical cumulative distribution function (ECDF) evaluated at a single point x. In order to treat ECDF as a function of x, one needs to consider a functional convergence. This is beyond the scope of this thesis.

Corollary 2.2.4 (CLT of the ECDF)**.** *Assume that $X_1, \cdots, X_n$ are i.i.d. random variables with CDF F_X. Consider ECDF*

$$\frac{1}{n}\sum_{i=1}^{n} \mathbb{1}\{X_i \leq x\} = \frac{1}{n}\sum_{i=1}^{n} Y_i.$$

Then $\mathbb{E}[Y_i] = F_X(x)$ and $Var(Y_i) = F_X(x)(1 - F_X(x))$ and the following CLT holds:

$$\sqrt{n}\left(\frac{1}{n}\sum_{i=1}^{n} \mathbb{1}\{X_i \leq x\} - F_X(x)\right) \xrightarrow{d} \mathcal{N}(0, F_X(x)(1 - F_X(x)))$$

for each fixed x.

Recall the definition of the quantile function of a random variable Y (see Q_Y in (2.1)) and of associated sample quantiles $\widehat{Q}_Y$ in (2.4). Thus, sample quantiles are the function of the ECDF:

$$\widehat{Q}_X(\tau) = \widehat{F}_X^{\leftarrow}(\tau).$$

Consequently, to obtain CLT for sample quantiles, one needs to look first at the delta method.

Theorem 2.2.5 (Delta method)**.** *Let $Y_1, \ldots, Y_n$ be an arbitrary sequence of random variables. Let $\overline{Y}_n = \frac{1}{n}\sum_{i=1}^{n} Y_i$. Assume that $\sqrt{n}(\overline{Y}_n - \mu) \xrightarrow{d} \mathcal{N}(0,\sigma^2)$. Let g be a smooth function. Then,*

$$\sqrt{n}(g(\overline{Y}_n) - g(\mu)) \xrightarrow{d} \mathcal{N}(0, g'(\mu)^2\sigma^2). \tag{2.10}$$

Above, "smooth" refers to the existence and finiteness of the second-order derivative. Now, we are ready to state CLT for sample quantiles. From Corollary 2.2.4 we know that

$$\sqrt{n}(\widehat{F}_X(x) - F_X(x)) \xrightarrow{d} \mathcal{N}(0, F_X(x)(1 - F_X(x))).$$

Setting $g(y) = Q_X(y)$ and so $g'(y) = \frac{1}{f_X(Q_X(y))}$, see [5], we obtain the following result for sample quantiles defined in (2.4).

Corollary 2.2.6 (CLT for sample quantiles)**.** *Assume that $X_1, \cdots, X_n$ are i.i.d. random variables with CDF F_X. Then for $\tau \in (0,1)$, as $n \to \infty$,*

$$\sqrt{n}(\widehat{Q}_X(\tau) - Q_X(\tau)) \xrightarrow{d} \mathcal{N}\left(0, \frac{\tau(1-\tau))}{(f_X(Q_X(\tau))^2}\right).$$

Chapter 3

Optimization problem

Building on our foundation of quantile regression, we incorporate regularization techniques such as ridge and LASSO penalties. Prior to applying these methods to quantile regression, we establish the necessary theoretical groundwork. We begin by introducing Lagrange multipliers and subgradients, common tools for solving optimization problems, see Section 3.1. However, as the complexity of these techniques increases significantly when dealing with non-differentiable loss or penalty function, we explore alternative approaches. We first introduce the notation for a standard linear regression problem and extend it to ridge and LASSO, see Section 3.2. In Section 3.2, we also demonstrate how ridge regression can be solved using Lagrange multipliers and coordinate descent. For LASSO regression, the non-differentiability of the penalty function necessitates the use of subgradients, though this method does not generalize well to higher dimensions. Instead, we demonstrate how the optimization problem for LASSO regression can be solved using the coordinate descent method.

In Section 3.3, we address quantile regression, which has a non-differentiable loss function. We investigate the feasibility of using the coordinate descent method for quantile regression with ridge or LASSO penalties, similarly to penalized linear regression. We demonstrate that this approach fails for quantile regression, requiring a different optimization method. We show that the optimization problem for quantile regression (without regularization) can be formulated and solved as a linear programming problem. Extending this framework, we reformulate the ridge-penalized quantile regression problem as a quadratic program, while the LASSO-penalized case results in a more complex version of the original quantile regression linear program.

The material for non-penalized quantile regression is classical and is based mostly on the book by Koenker [5], with an excellent explanation given in [2]. The idea to transform quantile regression with penalties into a linear program came from [9] and [7], but the specific implementation for LASSO and ridge in this thesis are an original author's contributions.

Finally, we end this chapter with an introduction to an alternative method that uses a smoothed quantile loss function to solve these problems. The method was taken from [8].

3.1 Lagrange multipliers

A general optimization framework is as follows. Assume we have a constrained optimization problem such as:

$$\text{argmin}_{\beta \in \mathbb{R}^p} L(\beta) \text{ subject to } g_j(\beta) \leq 0, j = 1, \ldots, m \ , \tag{3.1}$$

where $L : \mathbb{R}^p \to \mathbb{R}$ and $g_j : \mathbb{R}^p \to \mathbb{R}$ are convex functions. The problem is usually difficult to solve by direct computation and hence special techniques are required.

A popular technique used to solve problems such as (3.1) is called the Lagrangian approach. This method rewrites the constrained problem into an unconstrained problem by introducing a new variable. If both L and g_j's are differentiable, by taking the derivatives and setting them to 0, we are able to solve the problem and find the value of the variable we were initially optimizing.

Let's begin by introducing the notation. Let $\beta = (\beta_1, \ldots, \beta_p)^T \in \mathbb{R}^p$ and $\lambda = (\lambda_1, \ldots, \lambda_m)^T \in \mathbb{R}^m_+$. The Lagrangian notation for (3.1) is:

$$H(\beta; \lambda) = L(\beta) + \sum_{j=1}^{m} \lambda_j g_j(\beta),$$

where λ_j are nonnegative weights known as Lagrange multipliers. The Lagrange multiplier is used to impose a penalty when the constraint $g_j(\beta)$ is violated.

Let us set β^* as the optimal solution. Then, we have,

$$L(\beta^*) = \inf_{\beta \in \mathbb{R}^p} \sup_{\lambda \geq 0} H(\beta; \lambda).$$

As this equation can be complex to solve, we use Lagrange duality. This guarantees the existence of an optimal vector $\lambda^* \geq 0$ of Lagrange multipliers such that

$$L(\beta^*) = \min_{\beta \in \mathbb{R}^p} H(\beta; \lambda^*).$$

Thus, any optimal solution β^* of the problem (3.1), must satisfy $g_j(\beta^*) \leq 0$, and

$$0 = \nabla_\beta H(\beta^*; \lambda^*) = \nabla L(\beta^*) + \sum_{j=1}^{m} \lambda_j^* \nabla g_j(\beta^*), \tag{3.2}$$

where $\nabla_\beta H(\beta; \lambda)$ is the gradient of the function H calculated with respect to β.

Example. To get a better understanding of the application of Lagrange multipliers, we will go over a simple example before solving the ridge and LASSO problems. Consider the following problem:

$$L(x, y) = x^2 + 4y^2 - 2x + 8y$$

subject to

$$x + 2y = 7.$$

We can rewrite the constraint as $g(x, y) = 0$, where $g(x, y) = x + 2y - 7$. As such, we can now find the gradients of L and g:

- $\nabla L(x, y) = \begin{bmatrix} 2x - 2 \\ 8y + 8 \end{bmatrix}$;
- $\nabla g(x, y) = \begin{bmatrix} 1 \\ 2 \end{bmatrix}$.

Let (x^*, y^*) be the solution of the optimization problem. To find the solution, we use the gradients and write them as linear equations. As such, we have the following 3 equations:

$$\begin{aligned} 2x - 2 &= \lambda, \\ 8y + 8 &= 2\lambda, \\ x + 2y &= 7. \end{aligned}$$

Firstly, we find x^*:

$$\begin{aligned} 2x^* - 2 &= \lambda \\ 2x^* &= \lambda + 2 \\ x^* &= \frac{\lambda + 2}{2}\,. \end{aligned}$$

Now, we find y^*:

$$\begin{aligned} 8y^* + 8 &= 2\lambda \\ 8y^* &= 2\lambda - 2 \\ 4y^* &= \lambda - 4 \\ y^* &= \frac{\lambda - 4}{4}\,. \end{aligned}$$

As both x^* and y^* depend on λ, we solve for it as well.

$$\begin{aligned} x^* + 2y^* &= 7 \\ \frac{\lambda + 2}{2} + \frac{\lambda - 4}{4} &= 7 \\ \frac{2\lambda - 2}{2} &= 7 \\ \lambda - 1 &= 7 \\ \lambda &= 8\,. \end{aligned}$$

Therefore, we have that:

$$x^* = \frac{8+2}{2} = 5$$

and

$$y^* = \frac{8-4}{4} = 1 \, .$$

Thus, the minimum of L, subject to the constraint, is achieved at $(x^*, y^*) = (5, 1)$. Furthermore, from the calculation above (by setting $\lambda = 0$) we can also obtain that the solution to the unconstrained problem is $(1, -1)$.

3.1.1 Lagrange multipliers and subgradients

Definition 3.1.1 (Subgradients)**.** *Given a convex function $f : \mathbb{R}^p \to \mathbb{R}$, a vector $z \in \mathbb{R}^p$ is called a subgradient of f at β if*

$$f(\gamma) \geq f(\beta) + <z, \gamma - \beta>$$

for all $\gamma \in \mathbb{R}^p$, where $<z, u>$ is the inner product in $\mathbb{R}^p$.

- We denote the set of all subgradients of f at β, also called the subdifferential, by $df(\beta)$.
- When f is differentiable at β, the subdifferential reduces to a single vector, the gradient. Thus, $df(\beta) = \nabla f(\beta)$.
- At points of non-differentiability, the subdifferential is a convex set that englobes all possible subgradients.

Example 3.1.2. The absolute value function $f(\beta) = |\beta|$ is non-differentiable at $\beta = 0$. As such, we have the following:

$$df(\beta) = \begin{cases} +1 & \text{if } \beta > 0 \\ -1 & \text{if } \beta < 0 \\ (-1, 1) & \text{if } \beta = 0 \end{cases} .$$

The condition specified in (3.2) changes when dealing with possibly non-differentiable functions L and g_j. In fact, we get

$$0 \in dL(\beta^*) + \sum_{j=1}^{m} \lambda_j^* dg_j(\beta^*) \, . \tag{3.3}$$

In particular, if L is differentiable, $m = 1$ and $g(\beta) = |\beta|$, then (3.3) becomes

$$\nabla L(\beta) + \lambda s_j = 0, j = 1, \ldots, p,$$

where s_j is equal to $\text{sign}(\beta)$ if $\beta_j \neq 0$ and a value between $(-1, 1)$ if $\beta_j = 0$.

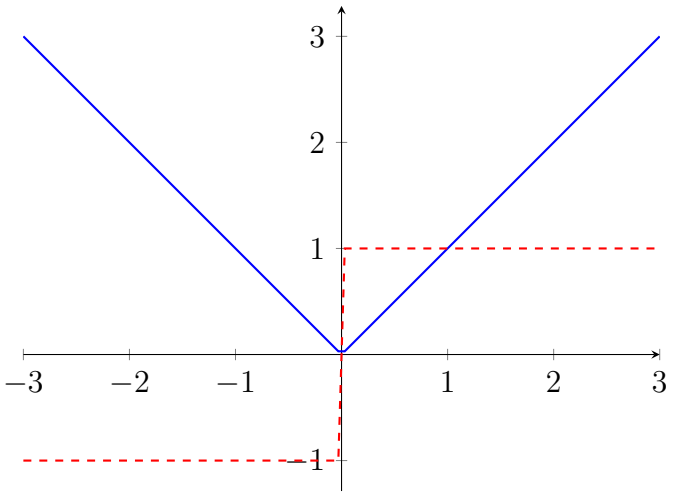

Figure 3.1: Graph of $f(\beta) = |\beta|$ and its "derivative"

3.2 Regression problem

In statistical applications the constrained optimization problem appears in a natural way, especially in the context of regression. In what follows, the vectors are column vectors. Assume we have the data $(\mathbf{X}_i, Y_i)$, $i = 1, \ldots, n$ and the following linear model is fitted to the data

$$Y_i = \beta_0 + \mathbf{X}_i^T \beta + \varepsilon_i,$$

where

- $\mathbf{X}_{i\cdot} = (X_{i1}, \ldots, X_{ip})^T$ are p-dimensional predictors, $i = 1, \ldots, n$;
- $\mathbf{X}_{\cdot j} = (X_{1j}, \ldots, X_{nj})^T$ are all observations for jth predictor, $j = 1, \ldots, p$.
- $\mathbf{X}$ is $(p+1) \times n$-dimensional design matrix, with the first row being 1 and the next rows being $\mathbf{X}_{i\cdot}$
- $\mathbf{Y} = (Y_1, \ldots, Y_n)^T$ is an n-dimensional response vector;
- $\beta = (\beta_1, \ldots, \beta_p)^T$ is a vector of the regression coefficient;
- $\beta_0 \in \mathbb{R}$ is an intercept;
- ε_i are i.i.d. random variables with mean zero and variance σ_ε^2, independent of $\mathbf{X}_{i\cdot}$.

If $p = 1$, then we will write $\mathbf{X}_{i\cdot} = X_i$.

Let $L : \mathbb{R}^{p+1} \to \mathbb{R}$ be a loss function. The goal is to solve the optimization problem (3.1). We will focus on $m = 1$, hence we have one constraint $g(\beta) \leq 0$.

Remark 3.2.1. Formally, the function L in (3.1) is defined on $\mathbb{R}^p$. Here, we include the intercept, hence we have a different dimension. However, there is no constraint on the intercept, hence this change of dimension does not cause any problems when applying the Lagrange method.

As discussed in the previous section, the optimization problem can be written equivalently in the Lagrange multiplier form

$$\text{argmin}_{\beta_0\in\mathbb{R},\beta\in\mathbb{R}^p} H(\beta_0,\beta) := \text{argmin}_{\beta_0\in\mathbb{R},\beta\in\mathbb{R}^p} \left\{L(\beta_0,\beta)+P(\beta)\right\} , \tag{3.4}$$

where $P:\mathbb{R}^p\to\mathbb{R}$ is a penalty function. The relation between the constraint g and the penalty is $P(\beta)=\lambda g(\beta)$ with $\lambda>0$.

Now, the problem in (3.4) can be solved using the method described in Section 3.1 (calculating the gradients). However, this method is applicable only when both the loss and penalty functions are differentiable. When either the loss or penalty is non-differentiable, one needs to use the subgradients. Hence, the standard Lagrange multiplier approach becomes complicated. In contrast, coordinate descent is more versatile. It can be applied in cases where either the loss or penalty function is non-differentiable, or even when both are non-differentiable. This technique entails finding the minimum of a function of one variable which is related to the target function H and applying it recursively.

In what follows, we show the link between the Lagrange multiplier approach and the coordinate descent for classical linear regression with ridge and LASSO constraints. This will be followed by the coordinate descent method only for quantile regression. Indeed, the coordinate descent method seems to be easier than the Lagrange method in the case when either loss or penalty is not smooth.

3.2.1 Regression with ridge

Let us start with $p=1$. Assume we have the data (X_i,Y_i), $i=1,\dots,n$, and the linear model $Y_i=\beta_0+\beta X_i+\varepsilon_i$, where ε_i are i.i.d. random variables with mean zero and variance σ_ε^2, independent of X_i. Consider the following minimization problem with the following constraint:

$$\text{argmin}_{\beta_0,\beta\in\mathbb{R}}\frac{1}{2n}\sum_{i=1}^{n}(Y_i-\beta_0-X_i\beta)^2 \text{ subject to } \beta^2\le t,$$

where $t>0$. Let now $\lambda>0$. We can rewrite the problem using the Lagrangian notation:

$$\text{argmin}_{\beta_0,\beta}\frac{1}{2n}\sum_{i=1}^{n}(Y_i-\beta_0-X_i\beta)^2+\frac{\lambda}{2}\beta^2 . \tag{3.5}$$

We note that we write $\lambda/2$ instead of λ. This is done for the computational convenience. Here, the loss function is quadratic,

$$L(\beta_0, \beta) = (1/2n) \sum_{i=1}^{n} (Y_i - \beta_0 - X_i\beta)^2$$

and the penalty function is $P(\beta) = \frac{\lambda}{2}\beta^2$. Originally, the constraint depends on t. This constraint is replaced by the λ term in the penalty.

In the lemma below we directly apply the Lagrange multiplier approach by taking the derivative of the target function.

Lemma 3.2.2 (Ridge regression)**.** The solution to the minimization problem (3.5) is:

$$\widehat{\beta} = \frac{\frac{1}{n}\sum_{i=1}^{n}(X_iY_i - \beta_0 X_i)}{\frac{1}{n}\sum_{i=1}^{n} X_i^2 + \lambda} \ . \tag{3.6}$$

Proof. We calculate the derivative of the function $H = L + P$ defined in (3.4) and set it to zero:

$$\begin{aligned}
\frac{d}{d\beta}H &= \frac{-1}{n}\sum_{i=1}^{n} X_i(Y_i - \beta_0 - X_i\beta) + \lambda\beta \\
&= \frac{-1}{n}\sum_{i=1}^{n}(X_iY_i - \beta_0 X_i - X_i^2\beta) + \lambda\beta \\
0 &= \frac{-1}{n}\sum_{i=1}^{n}(X_iY_i - \beta_0 X_i) + \frac{1}{n}\sum_{i=1}^{n} X_i^2\beta + \lambda\beta \\
\frac{1}{n}\sum_{i=1}^{n}(X_iY_i - \beta_0 X_i) &= \frac{1}{n}\sum_{i=1}^{n} X_i^2\beta + \lambda\beta \ .
\end{aligned}$$

□

For a general p, we solve the problem

$$\text{argmin}_{\beta_0,\beta}\frac{1}{2n}\sum_{i=1}^{n}(Y_i - \beta_0 - \mathbf{X}_i^T\beta)^2 + \frac{\lambda}{2}\sum_{j=1}^{p}\beta_j^2 \ . \tag{3.7}$$

Using again the direct calculation we obtain

$$\widehat{\beta} = (\mathbf{X}\mathbf{X}^T + \lambda I)^{-1}\mathbf{X}biY \ , \tag{3.8}$$

where Y is a $n \times 1$-dimensional vector.

Therefore, for ridge regression we have an explicit formula. In what follows, we will also show how to estimate the coefficients using the coordinate descent method. This is not needed here, however, it will be useful when dealing with LASSO.

Coordinate descent for ridge. Let $\lambda > 0$ and $t, u \in \mathbb{R}$. Consider the function $f : \mathbb{R} \to \mathbb{R}$ defined as

$$f_\lambda(\theta; t, u) = \frac{1}{2}(t - u\theta)^2 + \frac{1}{2}\lambda\theta^2 \,. \tag{3.9}$$

We find the minimum of the function f:

$$\begin{aligned} \frac{d}{d\theta} f_\lambda(\theta; t, u) &= -u(t - u\theta) + \lambda\theta \\ 0 &= -u(t - u\theta) + \lambda\theta \\ \widehat{\theta} &= G_\lambda(t, u) := \frac{ut}{u^2 + \lambda} \,. \end{aligned}$$

We generalize it to

$$f_\lambda(\theta; t_i, u_i, i = 1, \ldots, n) = \frac{1}{2n}\sum_{i=1}^{n}(t_i - u_i\theta)^2 + \frac{1}{2}\lambda\theta^2 \tag{3.10}$$

to obtain

$$\widehat{\theta} = \frac{\frac{1}{n}\sum_{i=1}^{n} u_i t_i}{\frac{1}{n}\sum_{i=1}^{n} u_i^2 + \lambda} \,.$$

To proceed, we introduce the notation

$$H(\beta_j \mid \beta_1, \ldots, \beta_{j-1}, \beta_{j+1}, \ldots, \beta_p)$$

to indicate that we treat the function $H : \mathbb{R}^p \to \mathbb{R}$ as a function of one variable, while the other variables are fixed.

This leads to the following implementation of the coordinate descent algorithm for linear regression with ridge penalty.

The coordinate descent for ridge for the general p:

- Standardize data so that $\frac{1}{n}\sum_{i=1}^{n} X_{ij}^2 = 1$ for $j = 1, \ldots, p$.
- Choose $\lambda > 0$ and a small $\delta > 0$.
- Set up initial values $\beta_1^{(0)}, \ldots, \beta_p^{(0)}$;
- Update β_j one at a time for $j = 1, \ldots, p$:

$$\widehat{\beta}_j^{(h+1)} := \mathrm{argmin}_{\beta_j} H(\beta_j \mid \widehat{\beta}_1^{(h)}, \ldots, \widehat{\beta}_{j-1}^{(h)}, \widehat{\beta}_{j+1}^{(h)}, \ldots, \widehat{\beta}_p^{(h)}) \,, \quad h = 0, 1, \ldots \,.$$

Hence,

$$\widehat{\beta}_j^{(h+1)} = \frac{\frac{1}{n}\mathbf{X}_{\cdot j}^T(\mathbf{Y} - \mathbf{X}^T\widehat{\beta}^{(h)}) + \widehat{\beta}_j^{(h)}}{1 + \lambda} \,.$$

- Stop when $\left\|\widehat{\beta}^{(h+1)} - \widehat{\beta}^{(h)}\right\|_2 < \delta$.

Below, we present a detailed calculation and implementation for $p = 2$ and $\beta_0 = 0$. Consider the function defined as

$$H(\beta_1, \beta_2) = \frac{1}{2n}\sum_{i=1}^{n}(Y_i - X_{i1}\beta_1 - X_{i2}\beta_2)^2 + \frac{1}{2}(\beta_1^2 + \beta_2^2).$$

We first set $\beta_1^{(0)}, \beta_2^{(0)}$ to arbitrary initial values. Now we will solve for the first iteration of $\beta_1 = \theta$:

$$\begin{aligned}\beta_1^{(1)} &= \text{argmin}_{\beta_1} H(\beta_1 \mid \beta_2 = \beta_2^{(0)}) \\ &= \text{argmin}_{\theta} \frac{1}{2n}\sum_{i=1}^{n}(\underbrace{Y_i - X_{i2}\beta_2^{(0)}}_{t_i} - \underbrace{X_{i1}}_{u_i}\theta)^2 + \frac{1}{2}\theta^2,\end{aligned}$$

hence

$$\begin{aligned}\widehat{\theta} &= \frac{\frac{1}{n}\sum_{i=1}^{n} X_{i1}(Y_i - X_{i2}\beta_2^{(0)})}{\underbrace{\frac{1}{n}\sum_{i=1}^{n} X_{i1}^2}_{1} + \lambda} \\ &= \frac{\frac{1}{n}\sum_{i=1}^{n} X_{i1}(Y_i - X_{i1}\beta_1^{(0)} - X_{i2}\beta_2^{(0)}) + \frac{1}{n}\beta_1^{(0)}\sum_{i=1}^{n} X_{i1}^2}{1+\lambda} \\ &= \frac{\frac{1}{n}\sum_{i=1}^{n} X_{i1}(Y_i - X_i^T\beta^{(0)}) + \beta_1^{(0)}}{1+\lambda}.\end{aligned}$$

As such we get:

$$\widehat{\beta}_1^{(1)} = \frac{\frac{1}{n}\mathbf{X}_{\cdot 1}^T(\mathbf{Y} - \mathbf{X}^T\beta^{(0)}) + \beta_1^{(0)}}{1+\lambda}.$$

Similarly, we get,

$$\widehat{\beta}_2^{(1)} = \frac{\frac{1}{n}\mathbf{X}_{\cdot 2}^T(\mathbf{Y} - \mathbf{X}^T\beta^{(0)}) + \beta_2^{(0)}}{1+\lambda}.$$

We note that when applying coordinate descent in the ridge regression problem, we assumed that the data is standardized so that $\frac{1}{n}\sum_{i=1}^{n} X_{ij}^2 = 1$. This is not needed in this particular case (well, we do not need to do coordinate descent in the first place), but will be necessary when dealing with LASSO regression below.

Example 3.2.3. In this example, we simulated a linear regression with two predictors and calculated the OLS estimator $\widehat{\beta}$ using the equation (3.6) and the algorithm for coordinate descent for ridge presented previously. With a sample size of $n = 100$ and no intercept, we repeated this $N = 1000$ times. The results are displayed in Figure 3.2. We see virtually no difference between the coordinate descent algorithm and the equation (3.6)

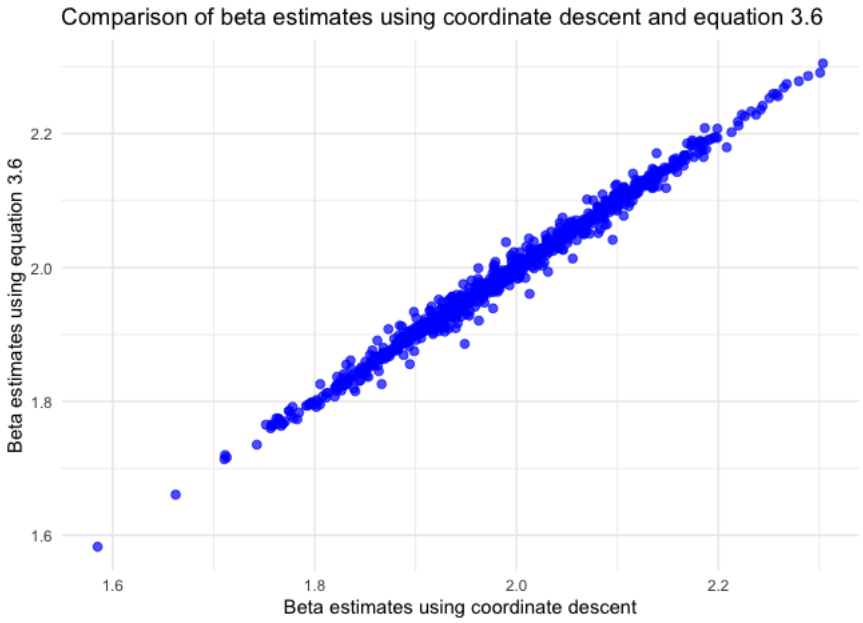

Figure 3.2: Beta estimates comparison for β_1.

3.2.2 Regression with LASSO

Let us start with $p = 1$. Assume we have the data (X_i, Y_i), $i = 1, \ldots, n$ and the linear model $Y_i = \beta_0 + \beta X_i + \varepsilon_i$, where ε_i are i.i.d. random variables with mean zero and variance σ_ε^2, independent of X_i.

We have the following constrained optimization problem:

$$\text{argmin}_{\beta_0,\beta} \frac{1}{2n} \sum_{i=1}^{n} (Y_i - \beta_0 - X_i\beta)^2 \text{ subject to } |\beta| \leq t \,, \tag{3.11}$$

where $t > 0$. Let $\lambda > 0$. We re-write the above problem using the Lagrangian form. We solve the following minimization problem:

$$\text{argmin}_{\beta_0,\beta} \frac{1}{2n} \sum_{i=1}^{n} (Y_i - \beta_0 - X_i\beta)^2 + \lambda|\beta| \,. \tag{3.12}$$

Here, the loss function is quadratic, $L(\beta_0, \beta) = (1/2n)\sum_{i=1}^n (Y_i - \beta_0 - X_i\beta)^2$ and the penalty function is $P(\beta) = \lambda|\beta|$.

We solve the problem (3.12) by using the Lagrangian method with subgradients, followed by the coordinate descent method.

Subgradient method for LASSO. We follow the description from Section 3.1.1. Assume that $\beta_0 = 0$. Assume also that $\frac{1}{n}\sum_{i=1}^n X_i^2 = 1$.

Below, $s = \text{sign}(\beta)$ if $\beta \neq 0$ or s is any value from $(-1, 1)$ if $\beta = 0$. We need to solve

$$
\begin{aligned}
0 &= -\frac{1}{n}\sum_{i=i}^{n}(Y_i - X_i\beta)X_i + \lambda s \\
\lambda s &= \frac{1}{n}\sum_{i=i}^{n}(Y_i - X_i\beta)X_i \\
\lambda s &= \frac{1}{n}\sum_{i=i}^{n} Y_i X_i - \frac{1}{n}\sum_{i=i}^{n} X_i^2\beta \\
\lambda s &= \frac{1}{n}\sum_{i=i}^{n} Y_i X_i - \beta, \\
\beta &= \frac{1}{n}\sum_{i=1}^{n} X_i Y_i - \lambda s\ .
\end{aligned}
$$

If $\beta > 0$, then $s = 1$ and

$$
\beta = \frac{1}{n}\sum_{i=1}^{n} X_i Y_i - \lambda\ .
$$

If $\beta < 0$, then $s = -1$ and

$$
\beta = \frac{1}{n}\sum_{i=1}^{n} X_i Y_i + \lambda\ .
$$

If $\beta = 0$,

$$
\begin{aligned}
0 &= \frac{1}{n}\sum_{i=1}^{n} X_i Y_i - \lambda s \\
\lambda s &= \frac{1}{n}\sum_{i=1}^{n} X_i Y_i \\
s &= \frac{(\frac{1}{n}\sum_{i=1}^{n} X_i Y_i)}{\lambda}\ .
\end{aligned}
$$

Hence, if $s \in (-1, 1)$, the solution to the optimization problem is 0. Combining the last three displays we obtain

$$\widehat{\beta} = \begin{cases} \frac{1}{n}\sum_{i=1}^n X_i Y_i + \lambda & \text{if } \frac{1}{n}\sum_{i=1}^n X_i Y_i < -\lambda \\ 0 & \text{if } -\lambda \le \frac{1}{n}\sum_{i=1}^n X_i Y_i \le \lambda \\ \frac{1}{n}\sum_{i=1}^n X_i Y_i - \lambda & \text{if } \frac{1}{n}\sum_{i=1}^n X_i Y_i > \lambda \end{cases} .$$

The argument above cannot be extended to general p, hence we need to apply a recursive technique such as the aforementioned coordinate descent.

Coordinate descent for LASSO. To solve the minimization problem, we can use the coordinate descent. Let $\lambda > 0$ and $t, u \in \mathbb{R}$. Similarly to (3.9), consider the function $f : \mathbb{R} \to \mathbb{R}$ defined as

$$f_\lambda(\theta; t, u) = \frac{1}{2}(t - u\theta)^2 + \lambda|\theta| . \tag{3.13}$$

(a) Let us start with the case when $\theta > 0$, thus we have:

$$f_\lambda(\theta; t, u) = \frac{1}{2}(t - u\theta)^2 + \lambda\theta . \tag{3.14}$$

We find the minimum of the function f at $\theta > 0$:

$$\begin{aligned} \frac{d}{d\theta} f_\lambda(\theta; t, u) &= -u(t - u\theta) + \lambda \\ &= -ut + u^2\theta + \lambda . \end{aligned}$$

Setting it to 0 we get

$$\widehat{\theta} = \frac{ut - \lambda}{u^2}.$$

(b) Now, let us look at the case when $\theta < 0$:

$$f_\lambda(\theta; t, u) = \frac{1}{2}(t - u\theta)^2 - \lambda\theta , \tag{3.15}$$

We find the minimum of the function f at $\theta < 0$:

$$\begin{aligned} \frac{d}{d\theta} f_\lambda(\theta; t, u) &= -u(t - u\theta) - \lambda \\ &= -ut + u\theta - \lambda . \end{aligned}$$

Setting it to 0 we get

$$\widehat{\theta} = \frac{ut + \lambda}{u^2}.$$

(c) If $\theta = 0$ we need to solve $0 = -ut + \lambda s$ with respect to $s \in (-1, 1)$. The solution is $s = ut/\lambda$. Hence, we obtain the following solution:

$$\widehat{\theta} = \begin{cases} \frac{ut+\lambda}{u^2} & \text{if } ut < -\lambda \\ 0 & \text{if } -\lambda \leq ut \leq \lambda \\ \frac{ut-\lambda}{u^2} & \text{if } ut > \lambda \end{cases} .$$

For future use, define the thresholding function

$$S_\lambda(t) = \begin{cases} t + \lambda & \text{if } t < -\lambda \\ 0 & \text{if } -\lambda \leq t \leq \lambda \\ t - \lambda & \text{if } t > \lambda \end{cases} . \tag{3.16}$$

Now, we repeat the same procedure for the function of n variables:

$$f_\lambda(\theta; t_i, u_i, i = 1, \ldots, n) = \frac{1}{2n} \sum_{i=1}^{n} (t_i - u_i\theta)^2 + \lambda|\theta| \tag{3.17}$$

to obtain

$$\widehat{\theta} = \begin{cases} \frac{\frac{1}{n}\sum_{i=1}^n u_i t_i + \lambda}{\frac{1}{n}\sum_{i=1}^n u_i^2} & \text{if } \frac{1}{n}\sum_{i=1}^n u_i t_i < -\lambda \\ 0 & \text{if } -\lambda \leq \frac{1}{n}\sum_{i=1}^n u_i t_i \leq \lambda \\ \frac{\frac{1}{n}\sum_{i=1}^n u_i t_i - \lambda}{\frac{1}{n}\sum_{i=1}^n u_i^2} & \text{if } \frac{1}{n}\sum_{i=1}^n u_i t_i > \lambda \end{cases} .$$

Assuming that $\frac{1}{n}\sum_{i=1}^n u_i^2 = 1$ we can express the above estimator as

$$\widehat{\theta} = S_\lambda \left(\frac{1}{n} \sum_{i=1}^{n} u_i t_i \right) .$$

Recall that we use the notation

$$H(\beta_j \mid \beta_1, \ldots, \beta_{j-1}, \beta_{j+1}, \ldots, \beta_p)$$

to indicate that we treat the function $H : \mathbb{R}^p \to \mathbb{R}$ as a function of one variable, while the other variables are fixed.

This leads to the following implementation of the coordinate decent algorithm for linear regression with the LASSO penalty.

The coordinate descent for LASSO for the general p:

- Standardize data so that $\frac{1}{n}\sum_{i=1}^n X_{ij}^2 = 1$ for $j = 1, \ldots, p$.

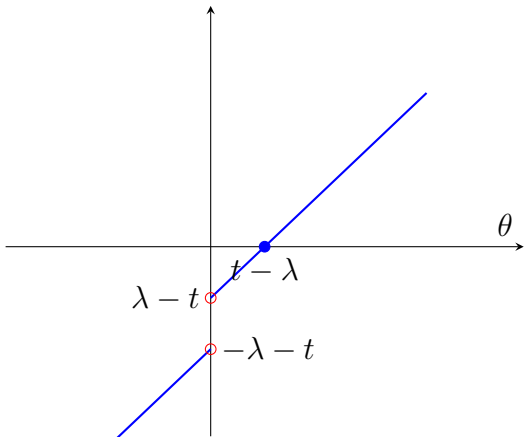

Figure 3.3: Coordinate descent method for LASSO: $t > \lambda$

- Choose $\lambda > 0$ and a small $\delta > 0$.
- Set up initial values $\beta_1^{(0)}, \ldots, \beta_p^{(0)}$;
- Update β_j one at a time for $j = 1, \ldots, p$:

$$\widehat{\beta}^{(h+1)} := \text{argmin}_{\beta_j} H(\beta_j \mid \widehat{\beta}_1^{(h)}, \ldots, \widehat{\beta}_{j-1}^{(h)}, \widehat{\beta}_{j+1}^{(h)}, \ldots, \widehat{\beta}_p^{(h)}) \ , \quad h = 0, 1, \ldots \ .$$

  Hence,

$$\widehat{\beta}_j^{(h+1)} = S_\lambda \left(\frac{1}{n} \mathbf{X}_{\cdot j}^T (\mathbf{Y} - \mathbf{X}^T \widehat{\beta}^{(h)}) + \widehat{\beta}_j^{(h)} \right) \ .$$

- Stop when $\left\| \widehat{\beta}^{(h+1)} - \widehat{\beta}^{(h)} \right\|_2 < \delta$.

In Figures 3.3-3.5, we provide the graphical representations of the coordinate descent method for the classical regression with LASSO.

3.3 Quantile regression

The classical quantile regression is usually solved by linear programming. The main reason for this approach is the non-differentiability of the loss function. Hence, we will replace the unconstrained optimization problem with a non-differentiable loss function with a constrained optimization problem.

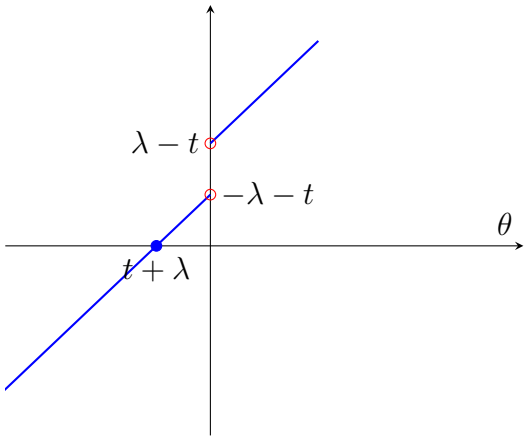

Figure 3.4: Coordinate descent method for LASSO: $t < -\lambda$

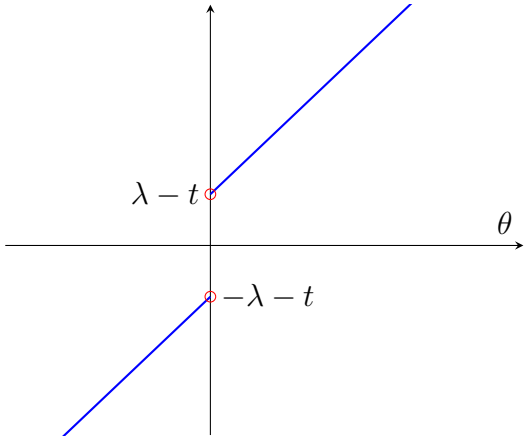

Figure 3.5: Coordinate descent method for LASSO: $-\lambda \leq t \leq \lambda$

3.3.1 Linear programming for quantile regression

In what follows, we will focus on $p = 1$ dimensional problem without intercept. The extension to higher dimensions is straightforward.

In order to solve quantile regression using the linear programming method, we must first transform the optimization problem into a constrained problem (linear program). In other words, we are taking the quantile regression equation written as:

$$\text{argmin}_\beta \sum_{i=1}^{n} \rho_\tau(Y_i - X_i\beta)$$

and transforming it into a linear program, which is written as:

$$\min_z c^T z \text{ subject to } Az = b,\ z \geq 0\ , \tag{3.18}$$

where the c, z, and b are vectors of the appropriate dimensions, and A is a matrix of the appropriate dimension. All these quantities have to be specified for a particular problem at hand. In what follows, we will specify them in case of quantile regression.

The goal of the linear program is to find a vector z that minimizes the quantile regression equation rewritten as $c^T z$ subject to certain constraints. To solve (3.18) and ensure that we do not break the non-negativity constraints, we will decompose $\varepsilon_i = Y_i - X_i\beta$ into positive and negative parts using slack variables. Thus, we have the following:

$$\begin{aligned} u_i &= \max(0, \varepsilon_i) = |\varepsilon_i| \mathbb{1}(\varepsilon_i \geq 0), \\ v_i &= \max(0, -\varepsilon_i) = |\varepsilon_i| \mathbb{1}(\varepsilon_i < 0), \\ \varepsilon_i &= u_i - v_i\ . \end{aligned}$$

Therefore, we can write

$$\sum_{i=1}^{n} \rho_\tau(Y_i - X_i\beta) = \sum_{i=1}^{n} \rho_\tau(\varepsilon_i) = \sum_{i=1}^{n} \tau u_i + (1-\tau)v_i = \tau \mathbf{1}_n^T \mathbf{U} + (1-\tau)\mathbf{1}_n^T \mathbf{V},$$

where $\mathbf{U} = (u_1, \cdots, u_n)^T$, $\mathbf{V} = (v_1, \cdots, v_n)^T$ and $\mathbf{1}_n = (1, \cdots, 1)^T$ is a vector of ones of dimension n. In addition, we will decompose β into its respective counterparts, so $\beta = \beta^+ - \beta^-$.

Using these variables, we are now able to determine A, b, z, and c. First, we know that

$$\mathbf{Y} = \mathbf{X}^T(\beta^+ - \beta^-) + \mathbf{I}_n\mathbf{U} - \mathbf{I}_n\mathbf{V},$$

where $\mathbf{I}_n$ is the identity matrix of size n. We observe that $b = \mathbf{Y}$ and we know that $b = Az$.

Thus, we have

$$b = \mathbf{X}^T(\beta^+ - \beta^-) + \mathbf{I}_n\mathbf{U} - \mathbf{I}_n\mathbf{V} = \begin{bmatrix}\mathbf{X} & -\mathbf{X} & \mathbf{I}_n & -\mathbf{I}_n\end{bmatrix} \begin{bmatrix}\beta^+ \\ \beta^- \\ \mathbf{U} \\ \mathbf{V}\end{bmatrix}.$$

So,

$$A = \begin{bmatrix}\mathbf{X} & -\mathbf{X} & \mathbf{I}_n & -\mathbf{I}_n\end{bmatrix},$$

is an $n \times (2 + 2n)$-matrix and

$$z = \begin{bmatrix}\beta^+ \\ \beta^- \\ \mathbf{U} \\ \mathbf{V}\end{bmatrix}$$

is a vector with dimension $2n + 2$. Knowing all the constraints, we are now able to solve the optimization problem, where the vector c is set as

$$\begin{bmatrix}\mathbf{0} \\ \tau\mathbf{1}_n \\ (1-\tau)\mathbf{1}_n\end{bmatrix},$$

and $\mathbf{0}$ is a 0 vector of size 2×1.

At this stage, we are ready to solve the linear optimization problem. It is not done analytically, rather numerically. As an illustration, we use the `R` package `lpSolve` that applies linear programming in the quantile regression context. In this situation, the user needs to provide the vectors and matrices A, c, z, b as defined above. In comparison, we will use the package `quantreg` which provides estimates for the quantile regression parameters. Here, the user does not need to provide the vectors and matrices A, c, z, b, they must simply load the data.

Example 3.3.1. In this example, we consider a quantile regression applied to a linear model with two predictors and no intercept. The sample size is $n = 100$ and we simulated the data from a normal distribution. We calculated $\widehat{\beta} = (\widehat{\beta}_1, \widehat{\beta}_2)$ using the `R` package `lpSolve`, setting the matrices and vectors A, c, z, b to those identified in Section 3.3.1 and setting $p = 2$ and $\tau = 0.5$. We also calculated the $\widehat{\beta}$ using the `R` package `quantreg`. We repeated it $N = 1000$ times. The results are displayed in Figures 3.6 and 3.7. We see virtually no difference - `quantreg` is using the linear programming.

3.3.2 Quantile regression and ridge

Let us start with $p = 1$. Assume we have the data (X_i, Y_i), $i = 1, \ldots, n$ and the linear model $Y_i = \beta_0 + \beta X_i + \varepsilon_i$, where ε_i are i.i.d. random variables with mean zero and

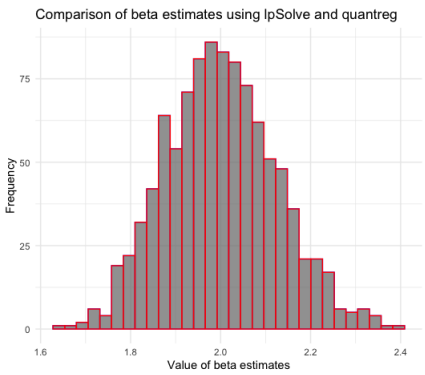

Figure 3.6: Estimates obtained from linear programming vs. obtained from quantreg - comparison using histograms.

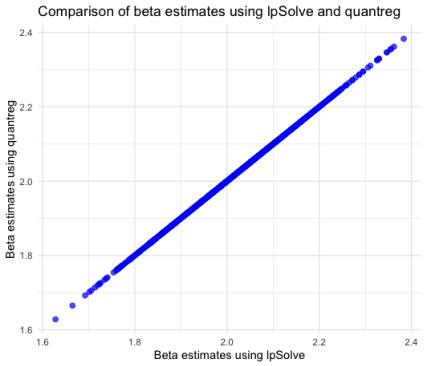

Figure 3.7: Estimates obtained from linear programming vs. obtained from quantreg - comparison using QQ-plot.

variance σ_ε^2, independent of X_i. Let $\lambda > 0$. The equation for quantile regression with ridge and an intercept is given by:

$$\frac{1}{2n}\sum_{i=1}^{n} L(Y_i - \beta_0 - X_i\beta) + \frac{1}{2}\lambda\beta^2 \ . \tag{3.19}$$

We solve the following minimization problem:

$$\text{argmin}_{\beta_0,\beta}\frac{1}{2n}\sum_{i=1}^{n} L(Y_i - \beta_0 - X_i\beta) + \frac{1}{2}\lambda\beta^2 \ . \tag{3.20}$$

Here, the loss function is piece-wise,

$$L(\beta_0, \beta) = \sum_{i=1}^{n} \tau(Y_i - \beta_0 - X_i\beta)\mathbb{1}\{Y_i \geq \beta\} - \sum_{i=1}^{n}(1-\tau)(Y_i - \beta_0 - X_i\beta)\mathbb{1}\{Y_i < \beta\}$$

and the penalty function is $P(\beta) = \frac{1}{2}\lambda\beta^2$.

In order to find the solution to the minimization problem, the first option is to proceed as in the case of linear regression with the ridge penalty. In that case, we calculated the derivative of the target function (the loss + the penalty). This technique will not work since the loss is not differentiable. Hence, similarly to LASSO linear regression (that involves a non-differentiable penalty), we could try the coordinate descent method in order to deal with the non-differentiability of the loss function. As we will see below, this is not going to work and other methods have to be implemented.

Coordinate descent for quantile regression with ridge.

To solve the minimization problem, we use coordinate descent. Let $\lambda > 0$, $\tau \in [0, 1]$ and $t \in \mathbb{R}$. Consider the function $f : \mathbb{R} \to \mathbb{R}$ defined as

$$f_{\lambda,\tau}(\theta; t) = \tau(t - \theta)\mathbb{1}(t \geq \theta) - (1 - \tau)(t - \theta)\mathbb{1}(t < \theta) + \frac{1}{2}\lambda\theta^2 \ . \tag{3.21}$$

(a) Let us start with the case when $t \geq \theta$, thus we have:

$$f_{\lambda,\tau}(\theta; t) = \tau(t - \theta) + \frac{1}{2}\lambda\theta^2 \ . \tag{3.22}$$

We find the minimum of the function f at $t \geq \theta$:

$$\frac{d}{d\theta} f_{\lambda,\tau}(\theta; t) = -\tau + \lambda\theta \ .$$

Setting it to 0, we get $\tau = \lambda\theta$ which gives: $\widehat{\theta} = \frac{\tau}{\lambda}$.

(b) Now let us look at the case when $t < \theta$, thus we have:

$$f_{\lambda,\tau}(\theta; t) = -(1 - \tau)(t - \theta) + \frac{1}{2}\lambda\theta^2 \ . \tag{3.23}$$

We find the minimum of the function f at $t < \theta$:

$$\frac{d}{d\theta} f_{\lambda,\tau}(\theta; t) = (1 - \tau) + \lambda\theta \ .$$

Setting it to 0, we get $-(1 - \tau) = \lambda\theta$ which gives: $\widehat{\theta} = \frac{-(1-\tau)}{\lambda}$.

As we can see the solution of the optimization problem depends on the knowledge of whether that solution is bigger or smaller than t. As a consequence, in order to implement the coordinate descent method, we need to know the original value of the parameter that, in fact, we want to estimate. As a result of the issues with the coordinate descent method, we must use another method to solve this optimization problem. Linear programming, as introduced above, is a known method used to solve quantile regression and can be extended to solve ridge and LASSO optimization problems.

Linear programming for quantile regression with ridge.

Similar to quantile regression and linear programming, we consider $p = 1$. In the case of quantile regression with ridge, we have a quadratic term for the penalty and as such we can use a modification of linear programming called quadratic programming. Thus, we want to rewrite our optimization problem as:

$$\text{argmin}_\beta \sum_{i=1}^{n} \rho_\tau(Y_i - X_i\beta) + \lambda\beta^2 = \text{argmin}_{e_i} \sum_{i=1}^{n} \rho_\tau(\varepsilon_i) + \lambda(\beta^+ - \beta^-)^2$$

and transform it into a quadratic linear program which is written like:

$$\min_z \left\{ \frac{1}{2} z^T G z + c^T z \right\} \text{ subject to } Az = b, z \geq 0 \, . \tag{3.24}$$

As in the quantile regression case, c, z, and b are vectors of the appropriate dimensions and A and G are matrices of appropriate dimension. Moreover, G has to be positive semi-definite. All of these vectors and matrices must be specified for a particular problem, and we will now specify them for the case of quantile regression with ridge. As there are no constraints on the ridge penalty, A, z, b, c remain the same as in the linear programming case for quantile regression. We must simply find the matrix G to solve the quadratic program. We want to find a matrix G such that

$$\frac{1}{2} z^T G z = \lambda\beta^2 \, .$$

As such,

$$G = \begin{bmatrix} 2\lambda\mathbf{I}_1 & 0 & 0 & 0 \\ 0 & 2\lambda\mathbf{I}_1 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix},$$

a $(2p + 2n) \times (2p + 2n)$ matrix with $p = 1$, and $\mathbf{I}_1$ is the identity matrix of size $p = 1$. (Of course, $\mathbf{I}_1$ is just a scalar, one, but we keep the current notation to indicate how the matrix G looks like when p is arbitrary).

As before, the quadratic optimization problem is solved with the help of some packages, such as `rqPen` in `R`, since an implementation of the procedure described above is rather complicated.

3.3.3 Quantile regression and LASSO

Let us start with $p = 1$. Assume we have the data (X_i, Y_i), $i = 1, \ldots, n$ and the linear model $Y_i = \beta_0 + \beta X_i + \varepsilon_i$, where ε_i are i.i.d. random variables with mean zero and

variance σ_ε^2, independent of X_i. Let $\lambda > 0$. The equation for quantile regression with LASSO and an intercept is given by:

$$\frac{1}{n}\sum_{i=1}^{n} L(Y_i - \beta_0 - X_i\beta) + \lambda|\beta| \ . \tag{3.25}$$

We solve the following minimization problem:

$$\text{argmin}_{\beta_0,\beta}\frac{1}{n}\sum_{i=1}^{n} L(Y_i - \beta_0 - X_i\beta) + \lambda|\beta| \ . \tag{3.26}$$

Here, the loss function is piece-wise,

$$L(\beta_0, \beta) = \sum_{i=1}^{n} \tau(Y_i - \beta_0 - X_i\beta)\mathbb{1}(Y_i \geq \beta) - \sum_{i=1}^{n}(1-\tau)(Y_i - \beta_0 - X_i\beta)\mathbb{1}(Y_i < \beta)$$

and the penalty function is $P(\beta) = \lambda|\beta|$.

As we have already learned in the context of quantile regression with the ridge penalty, the coordinate descent method is not going to work. Therefore, we will solve the optimization problem using a linear program.

Linear programming for quantile regression with LASSO.

Similar to quantile regression and linear programming, we will consider $p = 1$. In this case, we want to rewrite our optimization problem given by:

$$\text{argmin}_\beta \sum_{i=1}^{n} \rho_\tau(Y_i - X_i\beta) + \lambda|\beta| = \text{argmin}_{\varepsilon_i} \sum_{i=1}^{n} \rho_\tau(\varepsilon_i) + \lambda\gamma,$$

with some γ such that $|\beta| \leq \gamma$ and transform it into a linear program, which is written as:

$$\text{min}_z c^T z \text{ subject to } Az = b, z \geq 0, \ -\gamma \leq \beta \leq \gamma \ . \tag{3.27}$$

It should be noted here that, compared to the simple case of quantile regression, quantile regression with LASSO has an additional parameter and constraint. Thus, we will have slightly different A, z, b, c. First, we know that

$$\begin{aligned}\mathbf{Y} &= \mathbf{X}^T(\beta^+ - \beta^-) + \mathbf{I}_n\mathbf{U} - \mathbf{I}_n\mathbf{V} \\ &= \begin{bmatrix}\mathbf{X} & -\mathbf{X} & \mathbf{I}_n & -\mathbf{I}_n & \mathbf{0}\end{bmatrix} \begin{bmatrix}\beta^+ \\ \beta^- \\ \mathbf{U} \\ \mathbf{V} \\ \gamma\end{bmatrix} \ ,\end{aligned}$$

where $\mathbf{0}$ is a vector of size $p = 1$. So,

$$A = \begin{bmatrix} \mathbf{X} & -\mathbf{X} & \mathbf{I}_n & -\mathbf{I}_n & \mathbf{0} \end{bmatrix},$$

is an is the $n \times (3 + 2n)$ matrix and

$$z = \begin{bmatrix} \beta^+ \\ \beta^- \\ \mathbf{U} \\ \mathbf{V} \\ \gamma \end{bmatrix},$$

is a vector with dimension $3+2n$. Our last constraint is the inequality: $-\gamma \leq \beta \leq \gamma$. To solve this, we can rewrite the inequality into 2 inequalities, $\beta - \gamma \leq 0$ and $-\gamma - \beta \leq 0$. As such, we can write the constraint as the product of two matrices:

$$\begin{bmatrix} \mathbf{I}_1 & -\mathbf{I}_1 \\ -\mathbf{I}_1 & \mathbf{I}_1 \end{bmatrix} \begin{bmatrix} \beta \\ \gamma \end{bmatrix} \leq \begin{bmatrix} 0 \\ 0 \end{bmatrix}.$$

Now that we have all the constraints, we are now able to solve the optimization problem, where the vector c is set as

$$\begin{bmatrix} \mathbf{0} \\ \tau \mathbf{1}_n \\ (1-\tau)\mathbf{1}_n \\ \lambda \mathbf{1_1} \end{bmatrix},$$

a $(3 + 2n)$-dimensional vector.

Again, the optimization problem is solved with the help of some packages, such as `rqPen` since an implementation of the procedure described above is rather complicated.

Smoothed quantile regression with ridge.

As the main issue with quantile regression is due to the non-differentiability of the loss function, we can approximate it with a smooth loss function. This method is an alternative approach to the linear and quadratic programming showed above. We consider this approach as the implementation of it can be easier numerically compared to linear and quadratic programming. This smooth loss function is differentiable and is written as the following. Let $k > 0$ and define:

$$L_k(u;\tau) = \begin{cases} -(1-\tau)|u| - \frac{k(1-\tau)^2}{2} & \text{if } u < -(1-\tau)k, \\ \frac{u^2}{2k} & \text{if } -(1-\tau)k \leq u \leq \tau k, \\ \tau|u| - \frac{k(\tau)^2}{2} & \text{if } \tau k > u. \end{cases}$$

Its derivative with respect to u is:

$$L_k'(u;\tau) = \begin{cases} \tau - 1 & \text{if } u < -(1-\tau)k, \\ \frac{u}{k} & \text{if } -(1-\tau)k \leq u \leq \tau k, \\ \tau & \text{if } \tau k > u. \end{cases}$$

It should be noted that as k increases, the loss function becomes quadratic. Hence, the estimator $\widehat{\beta}$ outputted by this method resembles the OLS estimator.

Thus, we can now write the optimization problem. Let $p = 1$. Assume we have the data (X_i, Y_i), $i = 1, \ldots, n$ and the linear model $Y_i = \beta X_i + \varepsilon_i$, where ε_i are i.i.d. random variables with mean zero and variance σ_ε^2, independent of X_i. Let $\lambda > 0$ and $k > 0$. We solve the following minimization problem:

$$\text{argmin}_\beta F_k(\beta;\tau) := \text{argmin}_\beta \frac{1}{n} \left\{ \sum_{i=1}^{n} L_k(Y_i - X_i\beta;\tau) + \lambda\beta^2 \right\}. \tag{3.28}$$

We can solve this optimization problem numerically.

Example 3.3.2. In this example, we consider a quantile regression with one predictor and the ridge penalty. Assume the true $\beta = 2$. For a set of candidates $\beta^{(1)}, \ldots, \beta^{(q)}$ for β, we will evaluate F_k' at these different values of β. The value of β that gives $F_k'(\beta;\tau)$ closest to zero is our optimal solution. In our simulation, the sample size is $n = 100$, $\tau = 0.5$, we set $\lambda = 0$, $k = 1000$, hence we should obtain an estimator close to the OLS one. We simulated the predictors and the error, independently, from a normal distribution with a mean of 0 and variance of 1. We see that in Figure 3.8 the value of $\widehat{\beta}$ is around 2. We repeated this experiment with different values of k. The second graph illustrates the case of $k = 5$. The optimization problem is not very sensitive to the choice of k.

If $\lambda \neq 0$ the estimates seem to be very sensitive when k is large, however, they are very stable when k is small.

However, this numerical procedure (as any numerical procedure of this type) is not efficient in high dimensions. In the paper, [8], the authors did some expansions of the target function. However, it is not obvious what is exactly the benefit of using the smoothed regression. The optimization problem is as complex as linear programming for quantile regression. Furthermore, the parameter k has to be chosen. Thus, we do not pursue this direction.

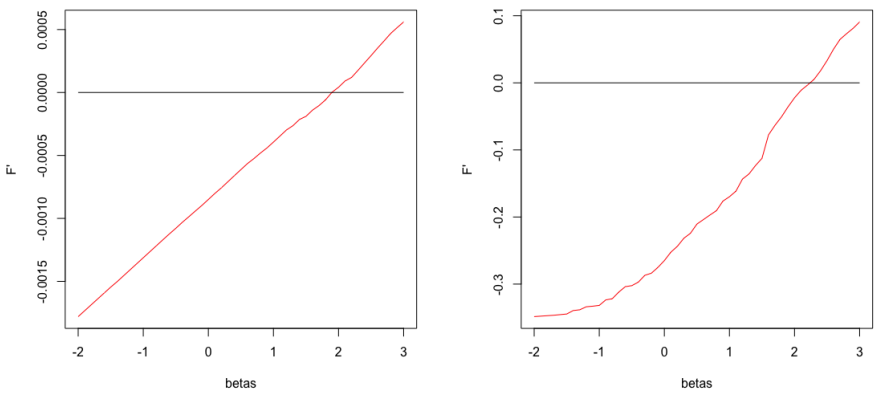

Figure 3.8: Smoothed quantile regression for k=1000 in the first plot and k=5 in the second.

Chapter 4

Extreme Quantile Regression

In this chapter, we integrate quantile regression and optimization techniques into an extreme value setting. Before we can apply ridge and LASSO penalization to extreme quantile regression, we must understand some fundamentals of extreme value theory and methods for estimating extreme quantiles. We begin this chapter with an introduction to regular variation, a classical and crucial concept for extreme value theory. This is based on [6] and is presented in Section 4.1. Following this introduction, we present a method to estimate extreme unconditional quantiles. This method, inspired by [10], utilizes the Hill estimator of the tail index. So, we will review the inference for this estimator, which was also based upon [6] and presented in Section 4.2.

In Section 4.3, we consider the first estimation method for extreme conditional quantiles. We assume that the conditional distribution is heavy-tailed, that is

$$F_{Y|X}(y \mid x) = \mathbb{P}(Y \leq y \mid X = x) = 1 - y^{-1/\xi(x)},$$

where $\xi(x)$ depends on the predictors x and the coefficients $\beta_0, \ldots, \beta_p$. This leads to

$$Q_{Y|X}(\tau \mid x) = (1-\tau)^{-\xi(x)}.$$

The parametric form of the conditional quantile allows us to calculate the log-likelihood, with a penalty. This likelihood is approximated by a quadratic function. As such, the optimal coefficients can be obtained by maximizing a quadratic function, with some challenges coming in case of LASSO penalty. This is combined with a coordinate descent. Although this approach has been considered in the literature, [10], the entire Section 4.3 presents the original contributions of the author of this thesis. Some approaches in the literature, [3], suggest estimating the slope and intercepts of a model separately. This does not seem to be feasible.

In Section 4.4, we consider the estimation of extreme conditional quantiles using extreme quantile regression. We assume that

$$Q_{Y|\mathbf{X}}(\tau \mid \mathbf{x}) \sim \beta_0(\tau) + \beta^T(\tau)\mathbf{x}.$$

Note that this is a completely different model from what is considered above. This approach, using extreme quantile regression to estimate the extreme conditional quantiles, originates from [4]. In particular, we consider the penalized regression under the common slope assumption. This approach stems from [10], but all the examples are original contributions of the author.

4.1 Regular Variation

Extreme value theory for distributions with unbounded support is composed of two main groups of distributions called Fréchet and Gumbel. These two groups can also be categorized as heavy-tailed and light-tailed distributions, respectively. In this thesis, we will focus on heavy-tailed distributions. To be more specific, we will only be working with regularly varying heavy tails.

We will now introduce a few basic notions for regularly varying random variables and functions.

Definition 4.1.1 (Regularly varying random variable)**.** *A positive random variable is called regularly varying with index $\alpha > 0$, also written as $X \in RV_{-\alpha}$, if:*

$$\lim_{x\to\infty} \frac{\mathbb{P}(X > tx)}{\mathbb{P}(X > x)} = t^{-\alpha} \tag{4.1}$$

for all $t > 0$.

The main distribution attributed to regular variation is the Pareto distribution.

Example 4.1.2 (Pareto random variable)**.** Take X a random variable that is Pareto with index $\alpha > 0$. Then, $\mathbb{P}(X > x) = x^{-\alpha}$, $x > 1$. Thus, we get:

$$\frac{(tx)^{-\alpha}}{x^{-\alpha}} = t^{-\alpha},$$

whenever $x > 1$ and $tx > 1$. So, X is regularly varying.

Example 4.1.3 (Pareto quantile function)**.** Take X, a Pareto random variable with index $\alpha > 0$, with a distribution function $F_X(x) = 1 - x^{-\alpha}$, $x > 1$. So, the quantile function is:

$$\begin{aligned} y &= 1 - x^{-\alpha} \\ x &= (1-y)^{-1/\alpha} \\ Q(u) &= (1-u)^{-1/\alpha}, \end{aligned}$$

Theorem 4.1.4 (Karamata theorem)**.** *Let f be a locally bounded measurable function on $[1,\infty)$ regularly varying at ∞ with index α. For $\beta \in \mathbb{R}$,*

$$\lim_{x\to\infty} \frac{\int_1^x t^{\beta-1} f(t)dt}{x^\beta f(x)} = \frac{1}{\alpha+\beta},\ \beta+\alpha>0,$$
$$\lim_{x\to\infty} \frac{\int_1^x t^{\beta-1} f(t)dt}{x^\beta f(x)} = -\frac{1}{\alpha+\beta},\ \beta+\alpha<0.$$

If l is a locally bounded measurable function on $[1,\infty)$ and slowly varying at ∞, then the function L defined by:

$$L(x) = \int_1^x \frac{l(t)}{t} dt \tag{4.2}$$

is slowly varying at ∞ and $\lim_{x\to\infty} \frac{l(x)}{L(x)} = 0$.

The Karamata theorem, in simpler terms, gives us the following results:

$$\int_x^\infty L(u)u^{-\alpha}du \sim \frac{1}{\alpha-1}x^{-\alpha+1}L(x), \text{with } \alpha>1.$$

This theorem shows us the tail behaviour of a slowly varying function and a regular varying function.

Lemma 4.1.5 (Breiman's lemma)**.** *Let X and Y be independent and non-negative random variables, such that X is regularly varying with tail index α, $\alpha>0$ and there exists $\varepsilon>0$ such that $\mathbb{E}[Y^{\alpha+\varepsilon}]<\infty$. Then XY is regularly varying with index α and*

$$\lim_{x\to\infty} \frac{\mathbb{P}(XY>x)}{\mathbb{P}(X>x)} = \mathbb{E}[Y^\alpha]. \tag{4.3}$$

Proof. In what follows we do the proof by assuming that X is Pareto. Then

$$\begin{aligned}
\mathbb{P}(XY>x) &= \int_0^\infty \mathbb{P}(Xy>x) f_Y(y)\,dy \\
&= \int_0^\infty \mathbb{P}(X>x/y) f_Y(y)\,dy \\
&= \int_0^\infty \left(\frac{x}{y}\right)^{-\alpha} f_Y(y)\,dy \\
&= x^{-\alpha}\int_0^\infty y^\alpha f_Y(y)\,dy \\
&= x^{-\alpha}\mathbb{E}[Y^\alpha] \\
&= \mathbb{P}(X>x)\mathbb{E}[Y^\alpha].
\end{aligned}$$

If X is regularly varying, then the result follows from a modified asymptotic argument in the spirit of the Karamata Theorem. □

Example 4.1.6. Let X and Y be independent random variables with the same Pareto distribution with tail index $\alpha > 0$. Let F be the distribution function of XY. Then,

$$\begin{aligned}
\mathbb{P}(XY > x) &= \int_1^\infty \mathbb{P}(Xy > x) f_Y(y)\, dy \\
&= \int_1^\infty \mathbb{P}(X > x/y) \alpha y^{-\alpha-1}\, dy \\
&= \int_1^x (x/y)^{-\alpha} \alpha y^{-\alpha-1}\, dy + \int_x^\infty \alpha y^{-\alpha-1}\, dy \\
&= \alpha x^{-\alpha} \int_1^x (y)^{\alpha} y^{-\alpha-1}\, dy + \int_x^\infty \alpha y^{-\alpha-1}\, dy \\
&= \alpha x^{-\alpha} \int_1^x y^{-1}\, dy + \alpha \int_x^\infty y^{-\alpha-1}\, dy \\
&= \alpha x^{-\alpha} (\log(x) - \log(1)) + \alpha((-1/\alpha)(\infty)^{-\alpha} - (-1/\alpha)(x)^{-\alpha})) \\
&= \alpha x^{-\alpha} \log(x) + x^{-\alpha}.
\end{aligned}$$

Lemma 4.1.7. *Let Y be regularly varying at ∞ with index α. Let $b > 0$. Then,*

$$\mathbb{P}(Y > x + b) \sim \mathbb{P}(Y > x)$$

as $x \to \infty$.

4.2 Estimation of extreme unconditional quantiles

As we saw in Chapter 2, the sample quantiles are the order statistics of the sample $Y_1, \ldots, Y_n$ being used. Let us take the quantile function $Q(u) = (1-u)^{-1/\alpha}$ that corresponds to the Pareto distribution with the parameter α. Setting $u = \frac{1}{n+1}$, we get that $\widehat{Q}(\frac{1}{n+1}) = Y_{(1)}$. Similarly, $\widehat{Q}(\frac{n}{n+1}) = Y_{(n)}$.

We know that

$$\begin{aligned}
\mathbb{P}(Y > Q(u)) &= Q(u)^{-\alpha} \\
&= ((1-u)^{-1/\alpha})^{-\alpha} \\
&= 1 - u\ .
\end{aligned}$$

So, let us take $u = 1 - p_n$, where $p_n \to 0$ so $u \to 1$ as $n \to \infty$. As such we get:

$$\mathbb{P}(Y > Q(1-p_n)) = \mathbb{P}\left(Y > Q\left(\frac{n}{n+1}\right)\right) = \frac{1}{n+1}\ .$$

Now, we can estimate $Q(\frac{n}{n+1})$ by $\widehat{Q}(\frac{n}{n+1}) = Y_{(n)}$. If $p_n = \frac{1}{n+1}$ we can estimate the quantile function by the largest order statistic, $\widehat{Q}(1-p_n) = Y_{(n)}$. If p_n is much smaller than $\frac{1}{n+1}$,

then we still estimate $Q(1-p_n)$ by $Y_{(n)}$. This estimation method has a few issues such as:

$$Q\left(1-\frac{1}{n+1}\right) = (n+1)^{1/\alpha},$$

$$Q(1-p_n) = p_n^{-1/\alpha}, \text{with } p_n << \frac{1}{n+1}.$$

As we can see, the second equation explodes faster, but we estimate both theoretical quantiles by the same value. Furthermore, there is no CLT for the largest order statistic $Y_{(n)}$. Thus, we use a different method to estimate the quantile function. Write

$$\begin{aligned}\frac{Q(1-p_n)}{Q(1-\frac{k}{n+1})} &= \frac{p_n^{-1/\alpha}}{(\frac{k}{n+1})^{-1/\alpha}} \\ &= \left(\frac{k}{(n+1)p_n}\right)^{1/\alpha}.\end{aligned}$$

Hence,

$$Q(1-p_n) = \left(\frac{k}{(n+1)p_n}\right)^{1/\alpha}\left(Q\left(1-\frac{k}{n+1}\right)\right).$$

Thus, we can estimate $Q(1-p_n)$ by:

$$\widehat{Q}(1-p_n) = \left(\frac{k}{(n+1)p_n}\right)^{1/\widehat{\alpha}}\left(\widehat{Q}\left(1-\frac{k}{n+1}\right)\right) = \left(\frac{k}{(n+1)p_n}\right)^{1/\widehat{\alpha}}(Y_{(n-k)}). \tag{4.4}$$

We must choose a k such $k = k_n \to \infty$ and $k_n/n \to 0$. To estimate $1/\widehat{\alpha} = \widehat{\xi}$, we will use the Hill estimator, which will be discussed next.

4.2.1 Estimation of the tail index

In order to estimate the quantiles using the formula provided in equation (4.4), we need to estimate the tail index of Pareto-like distributions. We will use the classical Hill estimator based on the sample $Y_1, \ldots, Y_n$.

Definition 4.2.1. *Let $\xi = 1/\alpha$. Let k be an integer such that $k = k_n \to \infty$ and $k/n \to 0$. Then, the Hill estimator is:*

$$\widehat{\xi}_k = \frac{1}{k}\sum_{j=1}^{n}\log_+(Y_j/Y_{(n-k)}) \tag{4.5}$$

To be able to do inference using the Hill estimator, we must break it down into steps. From this inference, we are also able to do inference on $1/\xi = \alpha$.

Step 1. The first step is looking at the tail empirical distribution based on the sample $Y_1, \ldots, Y_n$.

Definition 4.2.2 (Tail empirical distribution)**.** *Let $u_n \to \infty$ as $n \to \infty$ be such that $n\mathbb{P}(Y > u_n) \to \infty$. Let $s > 0$. Then*

$$\widehat{T}_n(s) = \frac{1}{n\mathbb{P}(Y > u_n)} \sum_{j=1}^{n} \mathbb{1}(Y_j > u_n s)$$

is called the tail empirical distribution.

We note that scaling by u_n means that we consider only the extremal values. Hence, formally, we should call $\widehat{T}_n$ the *extremal tail empirical distribution.*

In the next lemma, we state some properties of the tail empirical distribution.

Lemma 4.2.3. *Let $u_n \to \infty$ as $n \to \infty$ be such that $n\mathbb{P}(Y > u_n) \to \infty$. Let $s > 0$. Then*

$$\lim_{n\to\infty} \mathbb{E}[\widehat{T}_n(s)] = s^{-\alpha}$$

and

$$\lim_{n\to\infty} \left\{ n\mathbb{P}(Y > u_n) \mathrm{Var}(\widehat{T}_n(s)) \right\} = s^{-\alpha}.$$

Proof. We have

$$\begin{aligned}
\mathbb{E}[\widehat{T}_n(s)] &= \frac{1}{n\mathbb{P}(Y > u_n)} n\mathbb{P}(Y > u_n x) \xrightarrow[n\to\infty]{} s^{-\alpha} \\
\mathrm{Var}(\widehat{T}_n(s)) &= \frac{1}{n^2\mathbb{P}^2(Y > u_n)} n\mathbb{P}(Y > u_n s) \\
&= \frac{1}{n\mathbb{P}(Y > u_n)} \frac{\mathbb{P}(Y > u_n s)}{\mathbb{P}(Y > u_n)} \\
&\sim \frac{1}{n\mathbb{P}(Y > u_n)} s^{-\alpha}.
\end{aligned}$$

□

For a fixed $s > 0$, the above lemma suggests that the following CLT holds:

$$\sqrt{n\mathbb{P}(Y > u_n)} \{\widehat{T}_n(s) - s^{-\alpha}\} \xrightarrow{d} \mathcal{N}(0, s^{-\alpha}). \tag{4.6}$$

The result above can be proved using the characteristic function approach similar to Theorem 2.2.3. In order to have a limit for $\widehat{T}_n(s)$ considered as a function of s, the *functional central limit theorem* has to be considered, which we will not discuss here. Furthermore, we note that the centering by $s^{-\alpha}$ instead of $\mathbb{E}[\widehat{T}_n(s)]$ requires an additional control of bias. We will not discuss it here.

Step 2. The second step is looking at the log averages for $\xi = 1/\alpha$. Assume $Y \sim$ Pareto(α). Then, we know that $\mathbb{E}[\log(Y)] = \xi = 1/\alpha$. Hence,

$$\mathbb{E}\left[\frac{1}{n}\sum_{j=1}^{n}\log(Y_j)\right] = \xi = 1/\alpha.$$

However, since we don't always have that $Y \sim$ Pareto(α), we need to work with regularly varying tails.

Lemma 4.2.4 (Log averages)**.** *Let $u_n \to \infty$ as $n \to \infty$ be such that $n\mathbb{P}(Y > u_n) \to \infty$. Define*

$$\widehat{W}_n = \frac{1}{n\mathbb{P}(Y > u_n)}\sum_{j=1}^{n}\log_+(Y_j/u_n).$$

Then,

$$\mathbb{E}[\widehat{W}_n] \xrightarrow[n\to\infty]{} \xi = 1/\alpha$$

and

$$\lim_{n\to\infty}\left\{n\mathbb{P}(Y > u_n)\mathrm{Var}(\widehat{W}_n)\right\} = 2/\alpha^2.$$

Proof. Let F be the distribution of Y. Then

$$\begin{aligned}
\mathbb{E}[\widehat{W}_n] &= \frac{1}{\mathbb{P}(Y > u_n)}\mathbb{E}[\log_+(Y/u_n)] \\
&= \frac{1}{\mathbb{P}(Y > u_n)}\int_{u_n}^{\infty}\log(y/u_n)F(dy) \\
&= \frac{1}{\mathbb{P}(Y > u_n)}\int_{1}^{\infty}\log(s)F(u_n ds), s = y/u_n \\
&= \int_{1}^{\infty}\log(s)\frac{F(u_n ds)}{\mathbb{P}(Y > u_n)} \\
&\xrightarrow[n\to\infty]{} \int_{1}^{\infty}\log(s)\alpha s^{-\alpha-1}ds = 1/\alpha.
\end{aligned}$$

Moreover,

$$\begin{aligned}
\mathrm{Var}\left(\widehat{W}_n\right) &= \frac{1}{n^2\mathbb{P}^2(Y>u_n)} n\mathrm{Var}(\log_+(Y/u_n)) \\
&= \frac{1}{n\mathbb{P}(Y>u_n)} \frac{\mathrm{Var}(\log_+(Y/u_n))}{\mathbb{P}(Y>u_n)} \\
&= \frac{1}{n\mathbb{P}(Y>u_n)} \left(\frac{\mathbb{E}[\log_+^2(Y/u_n)]}{\mathbb{P}(Y>u_n)} - \left(\frac{\mathbb{E}[\log_+(Y/u_n)]}{\mathbb{P}(Y>u_n)} \right)^2 \mathbb{P}(Y>u_n) \right) \\
&= \frac{1}{n\mathbb{P}(Y>u_n)} \left(\frac{\mathbb{E}[\log_+^2(Y/u_n)]}{\mathbb{P}(Y>u_n)} - \frac{1}{\alpha^2}(1+o(1))o(1) \right) \\
&= \frac{1}{n\mathbb{P}(Y>u_n)} \int_1^\infty \log^2(s) \frac{F(u_n ds)}{\mathbb{P}(Y>u_n)} + o(1).
\end{aligned}$$

Hence,

$$\begin{aligned}
\lim_{n\to\infty} \left\{ n\mathbb{P}(Y>u_n)\mathrm{Var}\left(\widehat{W}_n\right) \right\} &= \lim_{n\to\infty} \int_1^\infty \log^2(s) \frac{F(u_n ds)}{\mathbb{P}(Y>u_n)} \\
&= \int_1^\infty \log^2(s) \alpha s^{-\alpha-1} ds \\
&= 2/\alpha^2.
\end{aligned}$$

□

Having computed the limiting mean and variance, we can expect the following central limit theorem that can be proven using a similar approach as for (4.6)

Corollary 4.2.5 (CLT for log averages)**.** *Let $u_n \to \infty$ as $n \to \infty$ be such that $n\mathbb{P}(Y > u_n) \to \infty$. Then*

$$\sqrt{n\mathbb{P}(Y>u_n)} \left(\frac{1}{n\mathbb{P}(Y>u_n)} \sum_{j=1}^n \log_+(Y_j/u_n) - 1/\alpha \right) \xrightarrow{d} \mathcal{N}(0, 2/\alpha^2).$$

Step 3. However, we want a CLT for the data-based Hill estimator,

$$\widehat{\xi}_k = \frac{1}{k} \sum_{j=1}^n \log_+(Y_j/Y_{(n-k)}).$$

So we must choose an integer k such that $k = k_n \to \infty, k/n \to 0$. Thus, we choose $k = n\mathbb{P}(Y > u_n)$, so that $u_n = Q(1 - k/n)$. We want to emphasize that this choice is theoretical only. In practice, we can choose k_n arbitrary, no need to link it to a specific u_n (formally, for each choice of k_n there is the corresponding u_n).

So rewriting,

$$\frac{1}{n\mathbb{P}(Y > u_n)} \sum_{j=1}^{n} \log_+(Y_j/u_n) = \frac{1}{k} \sum_{j=1}^{n} \log_+ \left(\frac{Y_j}{Q(1-k/n)} \right).$$

Since we know that

$$\frac{Y_{(n-k)}}{Q(1-k/n)} \xrightarrow{P} 1,$$

we have (as $n \to \infty$),

$$\frac{1}{n\mathbb{P}(Y > u_n)} \sum_{j=1}^{n} \log_+(Y_j/u_n) \sim \frac{1}{k} \sum_{j=1}^{n} \log_+(Y_j/Y_{(n-k)}).$$

However, this last approximation changes the asymptotics. In fact, we obtain as $k = k_n \to \infty$ (under the appropriate conditions to handle bias)

$$\sqrt{k} \left\{ \widehat{\xi}_k - 1/\alpha \right\} \xrightarrow{d} \mathcal{N}(0, 1/\alpha^2).$$

This proof relies on the functional convergence in (4.6) and is beyond the scope of this thesis. However, note that the limiting variance changes: becomes $1/\alpha^2$ instead of $2/\alpha^2$.

Step 4. Now we can perform the final step. Knowing the CLT for $\widehat{\xi}_k = 1/\widehat{\alpha}_k$, we can find the CLT for $\widehat{\alpha}_k$ by using the delta method.

Corollary 4.2.6 (CLT for $\widehat{\alpha}_k$)**.** *We have as $k = k_n \to \infty$,*

$$\sqrt{k}(\widehat{\alpha}_k - \alpha) \xrightarrow{d} \mathcal{N}(0, \alpha^2).$$

Proof. Let $\xi = 1/\alpha$ and $g(\xi) = 1/\xi$. Then, $g'(\xi) = -1/\xi^2$. As such,

$$\sqrt{k}(g(\widehat{\xi}_k) - g(\xi)) \xrightarrow{d} \mathcal{N}(0, (-1/\xi^2)^2 \xi^2),$$
$$\sqrt{k}(g(\widehat{\xi}_k) - g(\xi)) \xrightarrow{d} \mathcal{N}(0, 1/\xi^2),$$
$$\sqrt{k}(\widehat{\alpha}_k - \alpha) \xrightarrow{d} \mathcal{N}(0, \alpha^2).$$

□

4.3 Estimation of extreme conditional quantiles using penalized likelihood

We will present the first estimation method for extremal conditional quantiles, based on penalized likelihood. In order to solve the corresponding optimization problem, we will develop a coordinate descent algorithm.

Coordinate descent methodology, $p = 1$.

First, we will start by looking at $p = 1$ and extend it to higher dimensions. As such, we start by taking data (X_j, Y_j), where Y is conditionally Pareto given $X = x$. To be more precise, we assume that

$$F_{Y|X}(y \mid x) = \mathbb{P}(Y \leq y \mid X = x) = 1 - y^{-1/\xi(x)}. \tag{4.7}$$

The function $\xi(x)$ plays a role of the conditional tail index. We will impose a particular form, suitable for the likelihood method. Assume that

$$\xi(x) = e^{\beta_0 + \beta_1 x}, \tag{4.8}$$

where $\beta_0, \beta_1 \in \mathbb{R}$. Our goal is to estimate the coefficients β_0, β_1. To do so, we will start by finding the log-likelihood of the function based on the data (X_j, Y_j). The conditional density is

$$f_Y(y|x) = \frac{1}{\xi(x)} y^{-1/\xi(x)-1} = \frac{1}{e^{\beta_0+\beta_1 x}} y^{-\frac{1}{e^{\beta_0+\beta_1 x}}-1}.$$

Thus, the log-likelihood is

$$\ell(\beta_0, \beta_1) = \sum_{j=1}^{n} \left\{ -(\beta_0 + \beta_1 X_j) - (e^{-(\beta_0+\beta_1 X_j)} + 1) \right\} \log(Y_j).$$

As we can see, we cannot find a solution for β_0, β_1 analytically. We will employ an iterative method, by fixing one parameter, optimizing with respect to the other, and vice versa. For this, we need to derive a quadratic approximation for each of the variables β_0, β_1 separately, while the other variable is being fixed. As such, we will use Taylor's expansion to find a quadratic approximation. For β_0 and β_1 we will have, respectively:

$$Q(\beta_0, \gamma_0 \mid \beta_1) = \ell(\gamma_0, \beta_1) + (\beta_0 - \gamma_0) \frac{d\ell(\beta_0, \beta_1)}{d\beta_0} \mid_{\beta_0=\gamma_0} + \frac{1}{2}(\beta_0 - \gamma_0)^2 \frac{d^2\ell(\beta_0, \beta_1)}{d\beta_0^2} \mid_{\beta_0=\gamma_0}$$

and

$$Q(\beta_1, \gamma_1 \mid \beta_0) = \ell(\beta_0, \gamma_1) + (\beta_1 - \gamma_1) \frac{d\ell(\beta_0, \beta_1)}{d\beta_1} \mid_{\beta_1=\gamma_1} + \frac{1}{2}(\beta_1 - \gamma_1)^2 \frac{d^2\ell(\beta_0, \beta_1)}{d\beta_1^2} \mid_{\beta_1=\gamma_1} .$$

That is, $Q(\beta_0, \gamma_0 \mid \beta_1)$ is the expansion of the log-likelihood function for the variable β_0 around the point γ_0, while β_1 is fixed. Similarly, for $Q(\beta_1, \gamma_1 \mid \beta_0)$. The main point is that we are doing Taylor approximation with respect to both variables simultaneously.

In order to do so, we must find the respective first and second-order derivatives. First, we will find the first-order derivatives:

$$\frac{d\ell(\beta_0, \beta_1)}{d\beta_0} = \sum_{j=1}^{n} \{-1 + e^{-(\beta_0+\beta_1 X_j)} \log(Y_j)\}.$$

Setting $\beta_0 = \gamma_0$ gives

$$\frac{d\ell(\beta_0, \beta_1)}{d\beta_0} \mid_{\beta_0=\gamma_0} = \sum_{j=1}^{n} \{-1 + e^{-\gamma_0} e^{-\beta_1 X_j} \log(Y_j)\}.$$

Likewise,

$$\frac{d\ell(\beta_0, \beta_1)}{d\beta_1} \mid_{\beta_1=\gamma_1} = \sum_{j=1}^{n} \{-X_j + X_j e^{-\beta_0} e^{-\gamma_1 X_j} \log(Y_j)\}.$$

Now, we will find the second-order derivatives:

$$\frac{d^2\ell(\beta_0, \beta_1)}{d\beta_0^2} \mid_{\beta_0=\gamma_0} = -e^{-\gamma_0} \sum_{j=1}^{n} e^{-\beta_1 X_j} \log(Y_j)$$
$$\frac{d^2 l(\beta_0, \beta_1)}{d\beta_1^2} \mid_{\beta_1=\gamma_1} = -e^{-\beta_0} \sum_{j=1}^{n} X_j^2 e^{-\gamma_1 X_j} \log(Y_j).$$

Now, we can write out each approximation, by ignoring the terms that do not affect the optimization. We start with β_0. We replace $Q(\beta_0, \gamma_0 \mid \beta_1)$ with

$$\widetilde{Q}(\beta_0, \gamma_0 \mid \beta_1) = \beta_0 \frac{d\ell(\beta_0, \beta_1)}{d\beta_0}|_{\beta_0=\gamma_0} + \frac{1}{2}(\beta_0 - \gamma_0)^2 \frac{d^2\ell(\beta_0, \beta_1)}{d\beta_0^2}|_{\beta_0=\gamma_0} \tag{4.9}$$

For β_1, we have:

$$\widetilde{Q}(\beta_1, \gamma_1 \mid \beta_0) = \beta_1 \frac{d\ell(\beta_0, \beta_1)}{d\beta_1}|_{\beta_1=\gamma_1} + \frac{1}{2}(\beta_1 - \gamma_1)^2 \frac{d^2\ell(\beta_0, \beta_1)}{d\beta_1^2}|_{\beta_1=\gamma_1} \tag{4.10}$$

As such, to find the estimates for β_0 and β_1, we will be solving iteratively the following univariate minimization problems:

$$\text{argmin}_{\beta_0} \frac{-\widetilde{Q}(\beta_0, \gamma_0 \mid \beta_1)}{n} \tag{4.11}$$

and

$$\text{argmin}_{\beta_1} \frac{-\widetilde{Q}(\beta_1, \gamma_1 \mid \beta_0)}{n} + P_\lambda(\beta_1). \tag{4.12}$$

Above, $P_\lambda(\beta_1)$ is a penalty for the variable β_1, where $\mathrm{P}_\lambda(\beta) = \lambda|\beta|$ or $\frac{\lambda}{2}\beta^2$.

Coordinate descent, general p.

The formulas for general p are obtained in a similar manner, with some cumbersome notations. The log-likelihood is

$$\ell(\beta_0, \ldots, \beta_p) = \sum_{j=1}^{n} \left\{ -(\beta_0 + \beta_1 X_{j1} + \cdots + \beta_p X_{jp}) - (e^{-(\beta_0 + \beta_1 X_j + \cdots + \beta_p X_{jp})} + 1) \right\} \log(Y_j).$$

Its coordinatewise quadratic approximations become (we omit β_j variables in the ℓ notation)

$$Q(\beta_m, \gamma_m \mid \beta_j, j = 0, \ldots, p, j \neq m) = \ell(\beta_0, \ldots, \beta_{m-1}, \gamma_m, \beta_{m+1}, \ldots, \beta_p)$$
$$+ (\beta_m - \gamma_m) \frac{d\ell}{d\beta_m} \mid_{\beta_m = \gamma_m} + \frac{1}{2} (\beta_m - \gamma_m)^2 \frac{d^2\ell}{d\beta_m^2} \mid_{\beta_m = \gamma_m} .$$

As before, for optimization purposes, we replace Q with

$$\widetilde{Q}(\beta_m, \gamma_m \mid \beta_j, j = 0, \ldots, p, j \neq m) = \beta_m \frac{d\ell}{d\beta_m} \mid_{\beta_m = \gamma_m} + \frac{1}{2} (\beta_m - \gamma_m)^2 \frac{d^2\ell}{d\beta_m^2} \mid_{\beta_m = \gamma_m} .$$

Thus, we consider the following penalized optimization problem

$$\text{argmin}_{\beta_m} \frac{-\widetilde{Q}(\beta_m, \gamma_m \mid \beta_j, j = 0, \ldots, p, j \neq m)}{n} + P_\lambda(\beta_m) 1\{m \neq 0\}, \tag{4.13}$$

where P_λ is a penalty. We only need to consider the penalty for β_m, since the penalty function is separable in β_j.

We note also that as the penalty does not influence β_0, its estimate theoretically should not change for any penalty. However, the way our coordinate descent method is implemented, we will see the difference between the estimates of β_0 in the case of ridge and LASSO penalties. We do not know how to fix it.

4.3.1 Penalized likelihood with ridge penalty

Coordinate descent for ridge, $p = 1$.

First, we will explicitly write out what $-\frac{\widetilde{Q}}{n}$ is for both β_0 and β_1. We have

$$\frac{-\widetilde{Q}(\beta_0, \gamma_0 \mid \beta_1)}{n} = \beta_0 - \beta_0 e^{-\gamma_0} \frac{1}{n} \sum_{j=1}^{n} (e^{-\beta_1 X_j} \log(Y_j)) + \frac{1}{2} (\beta_0 - \gamma_0)^2 e^{-\gamma_0} \frac{1}{n} \sum_{j=1}^{n} (e^{-\beta_1 X_j} \log(Y_j))$$
$$= \beta_0 - \beta_0 e^{-\gamma_0} Z_n(\beta_1) + \frac{1}{2} (\beta_0 - \gamma_0)^2 e^{-\gamma_0} Z_n(\beta_1)$$

where

$$Z_n(\beta_1) = \frac{1}{n}\sum_{j=1}^{n}(e^{-\beta_1 X_j}\log(Y_j)). \tag{4.14}$$

Next,

$$\begin{aligned}
&\frac{-\widetilde{Q}(\beta_1,\gamma_1 \mid \beta_0)}{n} \\
&= \beta_1\bar{X} - \beta_1 e^{-\beta_0}\frac{1}{n}\sum_{j=1}^{n}(X_j e^{-\gamma_1 X_j}\log(Y_j)) + \frac{1}{2}(\beta_1-\gamma_1)^2 e^{-\beta_0}\frac{1}{n}\sum_{j=1}^{n}(X_j^2 e^{-\gamma_1 X_j}\log(Y_j)) \\
&= \beta_1\bar{X} - \beta_1 e^{-\beta_0}V_n^{(1)}(\gamma_1) + \frac{1}{2}(\beta_1-\gamma_1)^2 e^{-\beta_0}V_n^{(2)}(\gamma_1),
\end{aligned}$$

where for $q = 1, 2$,

$$V_n^{(q)}(\gamma_1) = \frac{1}{n}\sum_{j=1}^{n}(X_j^q e^{-\gamma_1 X_j}\log(Y_j)). \tag{4.15}$$

Now, we will evaluate (4.11). Taking the derivative with respect to β_0 and setting it to 0, we will find the estimate $\widehat{\beta_0}$:

$$\frac{d}{d\beta_0}\frac{-\tilde{Q}(\beta_0,\gamma_0 \mid \beta_1)}{n} = 1 - e^{-\gamma_0}Z_n(\beta_1) + (\beta_0-\gamma_0)e^{-\gamma_0}Z_n(\beta_1),$$

hence

$$\widehat{\beta_0} = 1 + \gamma_0 - \frac{1}{e^{-\gamma_0}Z_n(\beta_1)}. \tag{4.16}$$

Now, we will evaluate (4.13). Taking the derivative with respect to β_1 and setting it to 0, we will find the estimate $\widehat{\beta_1}$:

$$\frac{d}{d\beta_1}\frac{-\widetilde{Q}(\beta_1,\gamma_1 \mid \beta_0)}{n} = \bar{X} - e^{-\beta_0}V_n^{(1)}(\gamma_1) + (\beta_1-\gamma_1)e^{-\beta_0}V_n^{(2)}(\gamma_1) + \lambda\beta_1,$$

hence

$$\widehat{\beta_1} = \frac{-\bar{X} + e^{-\beta_0}V_n^{(1)}(\gamma_1) + \gamma_1 e^{-\beta_0}V_n^{(2)}(\gamma_1)}{e^{-\beta_0}V_n^{(2)}(\gamma_1) + \lambda}. \tag{4.17}$$

Implementation of the coordinate descent for ridge, general p.

Let $\bar{X}_i = \sum_{j=1}^{p} X_{ji}$, $i = 1, \ldots, p$,

$$Z_n(\beta_1^{(j)}, \ldots, \beta_p^{(j)}) = \frac{1}{n} \sum_{j=1}^{n} (e^{-(\beta_1 X_{j1} + \ldots + \beta_p X_{jp})} \log(Y_j))$$

and

$$V_n^{(q)}(\beta_1^{(j)}, \ldots, \beta_p^{(j)}) = \frac{1}{n} \sum_{j=1}^{n} (X_j^q e^{-(\beta_1 X_{j1} + \ldots + \beta_p X_{jp})} \log(Y_j)).$$

This leads to the following implementation of the coordinate descent for extremal quantile regression with the ridge penalty for the general p.

The coordinate descent for extremal quantile regression with ridge penalty for the general p:

- Choose $\lambda > 0$ and a small $\delta > 0$.
- Set up initial value $\beta_0^{(0)}, \beta_1^{(0)}, \ldots, \beta_p^{(0)}$;
- Update β_0 and β_i simultaneously for $i = 1, \ldots, p$:

$$\widehat{\beta}_0^{(h+1)} = 1 + \beta_0^{(h)} - \frac{1}{e^{-\beta_0^{(h)}} Z_n(\beta_1^{(h)}, \ldots, \beta_p^{(h)})}, \quad h = 0, 1, \ldots .$$

$$\widehat{\beta}_i^{(h+1)} = \frac{-\bar{X}_i + e^{-\beta_0^{(h)}} V_n^{(1)}(\beta_1^{(h)}, \ldots, \beta_p^{(h)}) + \beta_i^{(h)} e^{-\beta_0^{(h)}} V_n^{(2)}(\beta_1^{(h)}, \ldots, \beta_p^{(h)})}{e^{-\beta_0^{(h)}} V_n^{(2)}(\beta_1^{(h)}, \ldots, \beta_p^{(h)}) + \lambda}, \quad h = 0, 1, \ldots .$$

- Stop when $\left\| \widehat{\beta}^{(h+1)} - \widehat{\beta}^{(h)} \right\|_2 < \delta$, where $\widehat{\beta}^{(h)} = (\widehat{\beta}_1^{(h)}, \ldots, \widehat{\beta}_p^{(h)})$.

4.3.2 Penalized likelihood with LASSO penalty

Coordinate descent for LASSO, $p = 1$.

Recall that we considered the following problem

$$\text{argmin}_{\beta_m} \frac{-\widetilde{Q}(\beta_m, \gamma_m \mid \beta_0, \beta_j, j = 1, \ldots, p, j \neq m)}{n} + P_\lambda(\beta_m), \tag{4.18}$$

where P_λ is the LASSO penalty: $P_\lambda(\beta_m) = \lambda |\beta_m|$.

We will first solve the minimization problem for $p = 1$ and extend it to general dimension p. To simplify the mathematical equations for the LASSO penalty, we will start as we did in (3.13). This approach was not needed in the ridge case.

Let $\lambda > 0$ and $t, u, v \in \mathbb{R}$. Consider the function $f : \mathbb{R} \to \mathbb{R}$ defined as:

$$f_\lambda(\theta; t, u, v) = \theta t + \frac{1}{2}(\theta - u)^2 v + \lambda|\theta|. \tag{4.19}$$

The goal is to find the minimum for θ.

(a) Let us start with the case when $\theta > 0$. Taking the derivative with respect to θ and setting it to 0, we will find the estimate $\widehat{\theta}$:

$$\frac{d}{d\theta} f_\lambda(\theta; t, u, v) = t + (\theta - u)v + \lambda,$$

hence

$$\widehat{\theta} = \frac{uv - t - \lambda}{v}.$$

(b) Now, let us look at the case when $\theta < 0$. Taking the derivative with respect to θ and setting it to 0, we will find the estimate $\widehat{\theta}$:

$$\frac{d}{d\theta} f_\lambda(\theta; t, u, v) = t + (\theta - u)v - \lambda,$$

hence

$$\widehat{\theta} = \frac{uv - t + \lambda}{v}.$$

(c) If $\theta = 0$, we need to use the subgradient method. The optimization condition becomes

$$0 \in \frac{d}{d\theta}\left\{\theta t + \frac{1}{2}(\theta - u)^2 v\right\} |_{\theta=0} + \lambda \partial(|\theta|),$$

where $\partial(|\theta|)$ is the subdifferential of $|\theta|$. We recall that the subdifferential at 0 is the interval $(-1, 1)$ which can be parameterized by s, $s \in (-1, 1)$. We re-write the latter expression as

$$t + (\theta - u)v \; |_{\theta=0} + \lambda s = 0,$$

where $s = s(\theta)$ and $s = \text{sign}(\theta)$ if $\theta \neq 0$, while $s \in (-1, 1)$ is arbitrary if $\theta = 0$. Hence, we are solving

$$0 = t - uv + \lambda s$$

yielding

$$s = \frac{uv - t}{\lambda}.$$

Thus, we have the following thresholding estimator:

$$\widehat{\theta} = \begin{cases} \frac{uv-t-\lambda}{v} & \text{if } \lambda < uv - t \\ 0 & \text{if } -\lambda \leq uv - t \leq \lambda \\ \frac{uv-t+\lambda}{v} & \text{if } \lambda < -(uv - t) \end{cases} .$$

We can rewrite this using the soft-thresholding function (3.16), and so we have:

$$\frac{S_\lambda(uv - t)}{v}.$$

Having derived this "simple" equation, we will now identify what t, u, v, θ represent in the case of penalized likelihood:

$$\begin{aligned} \theta &= \beta_1 \ , \\ t &= \bar{X} - e^{-\beta_0} V_n^{(1)}(\gamma_1) \ , \\ u &= \gamma_1 \ , \\ v &= e^{-\beta_0} V_n^{(2)}(\gamma_1). \end{aligned}$$

Thus, we have:

$$\widehat{\beta}_1 = \frac{S_\lambda(-\bar{X} + e^{-\beta_0} V_n^{(1)}(\gamma_1) + \gamma_1 e^{-\beta_0} V_n^{(2)}(\gamma_1))}{(e^{-\beta_0} V_n^{(2)}(\gamma_1))}.$$

This leads to the following implementation of the coordinate descent for LASSO for the general p.

The coordinate descent for LASSO for the general p:

- Choose $\lambda > 0$ and a small $\delta > 0$.
- Set up initial value $\beta_0^{(0)}, \beta_1^{(0)}, \ldots, \beta_p^{(0)}$;
- Update β_0 and β_i simultaneously for $i = 1, \ldots, p$ and $h = 0, 1, \ldots$:

$$\widehat{\beta}_0^{(h+1)} = 1 + \beta_0^{(h)} - \frac{1}{e^{-\beta_0^{(h)}} Z_n(\beta_1^{(h)}, \ldots, \beta_p^{(h)})} \ ,$$

$$\widehat{\beta}_i^{(h+1)} = \frac{S_\lambda(-\bar{X}_i + e^{-\beta_0^{(h)}} V_n^{(1)}(\beta_1^{(h)}, \ldots, \beta_p^{(h)}) + \beta_i^{(h)} e^{-\beta_0^{(h)}} V_n^{(2)}(\beta_1^{(h)}, \ldots, \beta_p^{(h)}))}{(e^{-\beta_0^{(h)}} V_n^{(2)} \beta_1^{(h)}, \ldots, \beta_p^{(j)})} \ .$$

- Stop when $\left\| \widehat{\beta}^{(h+1)} - \widehat{\beta}^{(h)} \right\|_2 < \delta$.

4.3.3 Numerical considerations

The algorithms above, as written, could be very sensitive to the choice of the starting value of $\beta_0^{(0)}$, especially in the ridge case. We explain below why this is the case and how to mitigate it. For the sake of simplicity, let $p = 1$.

Recall the recursion defining equations (4.16)-(4.17). Recall also the notation from (4.15) and (4.14). We can apply the Law of Large Numbers (LLN) to the empirical means in (4.15) and (4.14). Assume that X has a density $f_X(x)$. Then

$$\begin{aligned} Z_n(\gamma_1) &\approx \mathbb{E}[e^{-\gamma_1 X}\log(Y)] \\ &= \int \mathbb{E}[e^{-\gamma_1 X}\log(Y) \mid X = x] f_X(x)dx \\ &= \int e^{-\gamma_1 x}\mathbb{E}[\log(Y) \mid X = x] f_X(x)dx \\ &= \int e^{-\gamma_1 x}\xi(x) f_X(x)dx \\ &= \int e^{-\gamma_1 x}e^{\beta_0+\beta_1 x} f_X(x)dx \\ &= e^{\beta_0}\int e^{-(\gamma_1-\beta_1)x} f_X(x)dx. \end{aligned}$$

The approximation above holds in probability (or even almost surely). We provide some explanations for the above calculation. Note that if a random variable Z has a Pareto distribution with the parameter $1/\xi$, then $\mathbb{E}[\log(Z)] = \xi$. Now, see (4.7), Y is Pareto conditionally on $X = x$, yielding $\mathbb{E}[\log(Y) \mid X = x] = \xi(x)$. Furthermore, $\xi(x)$ is given in (4.8).

Now, recall that the iteration for β_0 is

$$\widehat{\beta_0} = 1 + \gamma_0 - \frac{e^{\gamma_0}}{Z_n(\gamma_1)}.$$

Assume now that we start our algorithm at $\gamma_1 = \beta_1$ and an arbitrary γ_0 as the initial approximation for β_0. Then, thanks to the LLN described above,

$$\widehat{\beta_0} \approx 1 + \gamma_0 - \frac{e^{\gamma_0}}{e^{\beta_0}}.$$

This being said we can see from this equation that even if we guess perfectly β_1, unless we choose an initial approximation γ_0 close to the true value β_0, then we cannot control the ratio $\frac{e^{\gamma_0}}{e^{\beta_0}}$. In fact, we observe in our numerical studies that the problem stems from the ratio $e^{\gamma_0}/Z_n(\gamma_1)$ being too small. Then, at many initial iterations i, we will be getting the estimates of β_0 behaving like $1 + \gamma_0 i$. As a consequence, the estimation of β_0 may be very inaccurate, which in turn affects the estimation of β_1. Hence, we propose a **clipping** procedure, to control the ratio from below.

Example 4.3.1 (Extremal quantile regression with ridge, no clipping)**.** In this example, we simulated data $X \sim \text{Exp}(1)$ with a sample size of $n = 1000$ and dimension $p = 1$. We chose the true values of β_0 and β_1 as, respectively, -0.13 and -2.5 (these values come from a particular matching of a conditional model to an unconditional one).

We then simulated Y from a Pareto distribution with the parameter $e^{-\beta_0-\beta_1 X}$. From here, we applied the coordinate descent algorithm with $\lambda = 0.01$, $\delta = 1 \times 10^{-6}$, and a maximum of $n_{\max} = 1000$ iterations.

We performed two experiments with these values and different starting values, their plots are displayed in Figures 4.1 and 4.2. Figure 4.1 displays the convergence of β_0 and β_1 for starting values $\beta_0^{(0)} = 0$ and $\beta_1^{(0)} = 0$, whereas Figure 4.2 displays the convergence of β_0 and β_1 for starting values $\beta_0^{(0)} = \beta_0 - 0.5$ and $\beta_1^{(0)} = \beta_1 - 0.5$. Both experiments converge rapidly to values close to the true values. However, starting values have an influence on the quality of the algorithm (how close we get to the true values and the speed of convergence). We also observe big jumps in our iterates, thanks to the aforementioned issue with the ratio of the exponents.

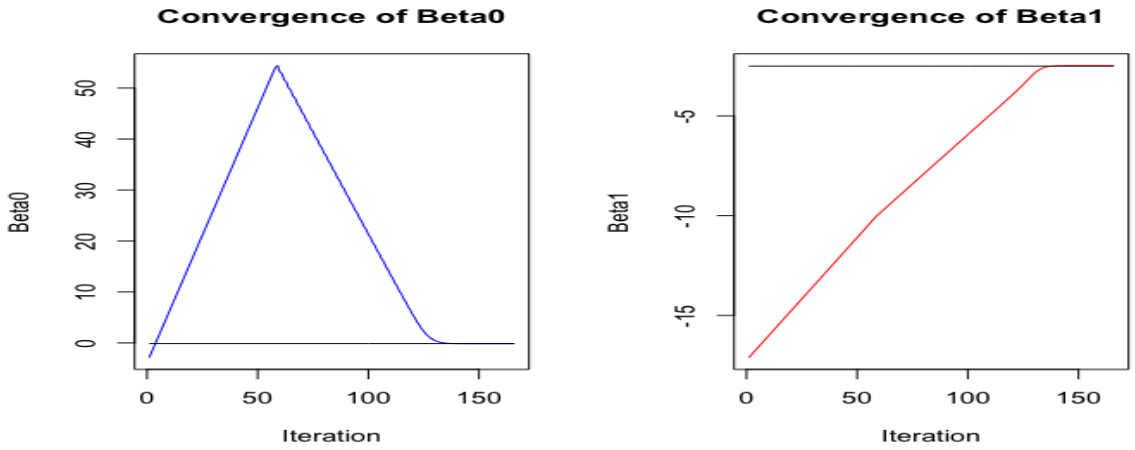

Figure 4.1: Extremal quantile regression with ridge, no clipping. Convergence of β_0 and β_1 estimates. Starting values are $\beta_0^{(0)} = 0$ and $\beta_1^{(0)} = 0$.

Example 4.3.2 (Extremal quantile regression with ridge, with clipping)**.** We continue with Example 4.3.1. This time we apply clipping from below, to prevent very small changes in the iterates for β. We clip the change at the level 0.9. The graph 4.3 below should be compared to Figure 4.1. The effect of clipping is obvious as the values of the β estimates are significantly smaller, they do not peak as high.

Example 4.3.3 (Extremal quantile regression with LASSO, with clipping)**.** Using the same initialization as in Example 4.3.1, we performed three experiments with these values

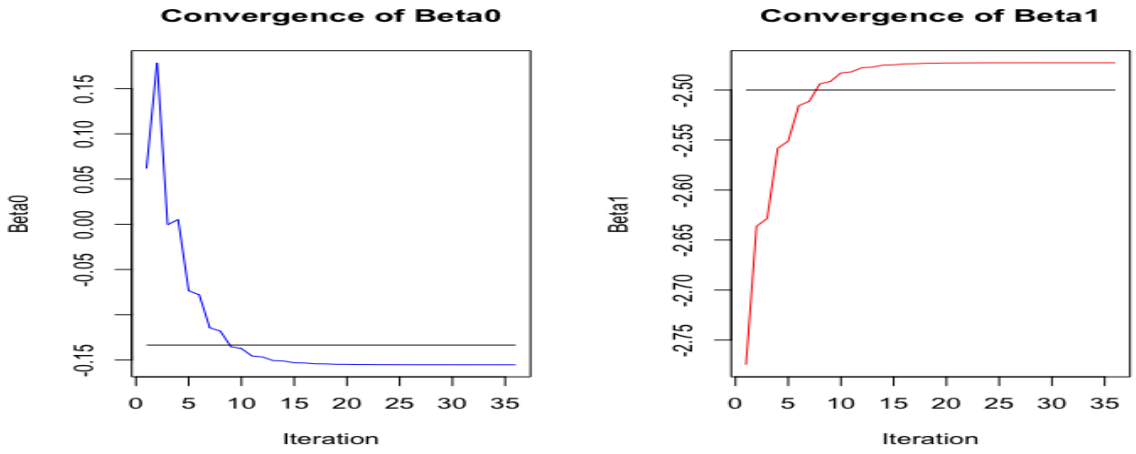

Figure 4.2: Extremal quantile regression with ridge, no clipping. Convergence of β_0 and β_1 estimates. Starting values are $\beta_0^{(0)} = \beta_0 - 0.5$ and $\beta_1^{(0)} = \beta_1 - 0.5$.

and different clipping values. Recall that $\lambda = 0.01$. For these experiments, the starting values are $\beta_0^{(0)} = 0$ and $\beta_1^{(0)} = 0$ for all three clipping values (0.1, 0.5, 0.9). We display the plots for the experiments in Figures 4.4, 4.5, and 4.6. All three experiments converge to values close to the true values. However, a higher clipping value slightly improves the quality of the algorithm. We also observe that the big jumps in our iterates decrease with a higher clipping. This helps with the convergence speed of the algorithm. We also note that there is little influence of thresholding. It is what is to be expected.

Example 4.3.4 (LASSO paths for extremal quantile regression, $p = 1$)**.** Recall that the LASSO path depicts estimates of different parameters in relation to the truncation value of λ. For this example, we will investigate the effects of different starting values and different clipping values on the LASSO path, where λ ranged from 0.01 to 2 with a step of 0.01. We simulate the data as in Example 4.3.1. We set the starting values to three different values; $0, -1, 0.5$ for both $\beta_0^{(0)}$ and $\beta_1^{(0)}$, and produced their respective LASSO paths to see if the starting values have an effect on the convergence of the estimates. As it is displayed in Figure 4.7, we see that the starting values do not have an effect on the convergence. Furthermore, we used three different clipping values; $0.1, 0.5, 0.9$, and produced their respective LASSO paths. As it can be seen in Figure 4.8, the clipping values do not have an effect on the convergence.

Example 4.3.5 (LASSO paths for extremal quantile regression, $p = 10$)**.** For this example, we started by simulation a predictor matrix of size 100×10, so we have $n = 100$ and $p = 10$. Each column is generated independently from $X \sim \text{Exp}(1)$. We set $\beta_0 = -0.12$ and

$$\beta_j = (-1.5, -1, -1.2, -0.9, -0.6, -1.9, -0.03, -0.0005, -0.9, -0.11)$$

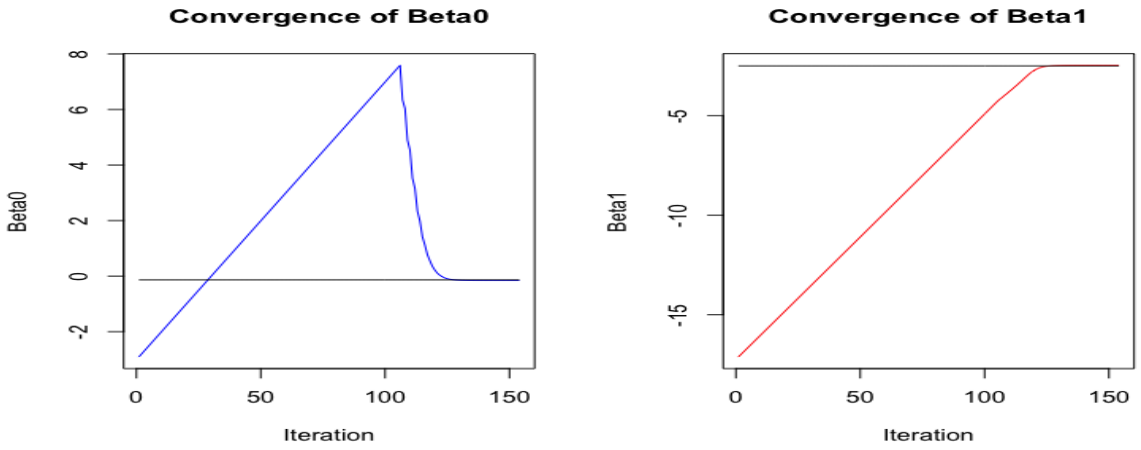

Figure 4.3: Extremal quantile regression with ridge, with clipping. Convergence of β_0 and β_1 estimates. Starting values are $\beta_0^{(0)} = 0$ and $\beta_1^{(0)} = 0$. Clipping value of 0.9.

for $j = 1, \ldots, 10$. We then simulated Y from a Pareto distribution with the parameter

$$e^{-(\beta_0+\beta_1 X_{j1}+\ldots+\beta_{10} X_{j10})}.$$

From here, we applied the coordinate descent algorithm with λ ranging from 0.01 to 1 with a step of 0.01, $\delta = 1 \times 10^{-6}$, and a maximum of $n_{\max} = 1000$ iterations. We note that we used a clipping value of 0.7, but as seen in Example 4.3.4, the clipping value does not affect the LASSO paths. The LASSO path for this experiment is displayed in Figure 4.9. We see here that as λ increases, the parameters go to 0. Furthermore, in this example, we see that $\beta_7, \beta_8, \beta_{10}$ go to 0 with a small value of λ which makes sense as they are the values that are the closest to 0 and each predictor has the same order of magnitude.

Summary of methodology. The examples performed above have shown us that the coordinate descent method with the ridge penalty is sensitive to the starting values. This problem can be mitigated with the clipping parameter. Additionally, for the sake of ease, we propose always having a starting value of 0 and adjusting the clipping value to improve the estimation rather than choosing a random starting value. On the other hand, the coordinate descent method with the LASSO penalty is not sensitive to the starting values. However, we saw that as the penalization parameter λ increases, the estimation of β_0 decreases in accuracy. As such, we propose this method to be used as a tool for model selection; finding the predictors that have an effect on the outcome, and then with this information performing the analysis with only those predictors. This way, we can ensure that the estimated values for all the β are accurate.

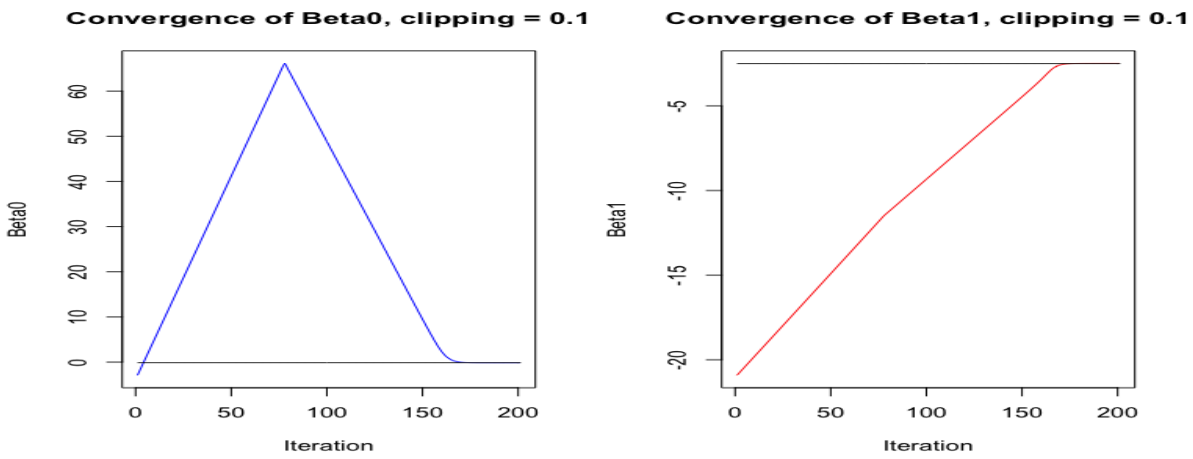

Figure 4.4: Extremal quantile regression with LASSO, with clipping of 0.1. Convergence of β_0 and β_1 estimates. The starting values are $\beta_0^{(0)} = 0$ and $\beta_1^{(0)} = 0$

4.4 Estimation of extreme conditional quantiles using extremal quantile regression

Extremal quantile regression is derived from regular quantile regression (see Chapter 2), but focuses solely on the extreme quantile values rather than on all quantiles. This means that we are looking at the situation when τ approaches 1 or 0. We will be working with $\tau \to 1$, which corresponds to the right tail.

Notation and assumption. Let's begin by introducing notation and assumptions. Let $(X_1, \ldots, X_p)$ be a random vector (predictor) and Y be a random variable (dependent variable). Let $\beta_0 = \beta_0(\tau)$ and $\beta(\tau) = (\beta_1(\tau), \ldots, \beta_p(\tau))^T \in \mathbb{R}^p$. The parameters β may depend on the quantile τ. Recall Definition 2.2.2 of the conditional quantile. We assume that as $\tau \to 1$,

$$Q_{Y|X}(\tau \mid x) \sim \beta_0(\tau) + x^T \beta(\tau), \tag{4.20}$$

where $x = (x_1, \ldots, x_p)^T$. There are several scenarios that lead us to equation (4.20). These scenarios were discussed already, after Definition 2.2.2. In this section, we focus on a particular situation when we model extreme tails.

We will go over three examples, a linear model, a non-linear model, and a multiplicative model. Also, it is important to note that the conditional Pareto model considered in (4.7) does not have the linear representation as in (4.20). Indeed, starting from (4.7), the conditional quantile is

$$Q_{Y|X}(\tau \mid x) = (1 - \tau)^{-\xi(x)}.$$

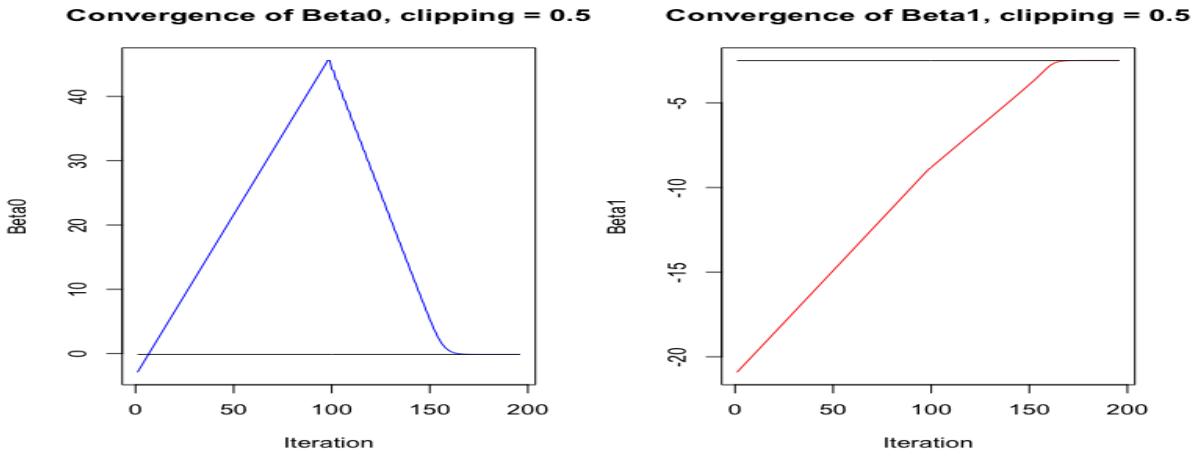

Figure 4.5: Extremal quantile regression with LASSO, with clipping of 0.5. Convergence of β_0 and β_1 estimates. The starting values are $\beta_0^{(0)} = 0$ and $\beta_1^{(0)} = 0$

As such, we will not be able to do a numerical comparison between the estimation methods as they use two different models.

Example 4.4.1 (Linear model)**.** Take the linear model $Y = \widetilde{\beta}_0 + \widetilde{\beta}_1 X + \varepsilon$, where ε has Pareto like tail, that is

$$Q_\varepsilon(\tau) \sim (1-\tau)^{-1/\alpha}, \quad \tau \to 1$$

and X has lighter tails than ε. Then

$$\mathbb{P}(Y \leq y \mid X = x) = \mathbb{P}(\varepsilon \leq y - \widetilde{\beta}_0 - \widetilde{\beta}_1 x) = F_\varepsilon(y - \widetilde{\beta}_0 - \widetilde{\beta}_1 x),$$

hence the conditional quantiles become

$$Q_{Y|X}(\tau \mid x) = Q_\varepsilon(\tau) + \widetilde{\beta}_0 + \widetilde{\beta}_1 x. \tag{4.21}$$

Hence, the extreme conditional quantile has the form (4.20) with

$$\beta_0(\tau) = \widetilde{\beta}_0 + Q_\varepsilon(\tau), \quad \beta_1(\tau) = \widetilde{\beta}_1. \tag{4.22}$$

Note: While regression methods estimate $\widetilde{\beta}_0$, the parameter of the original regression model, the quantile regression methods estimate β_0, the parameter of the quantile regression model.

Example 4.4.2. Take the regression model $Y = \widetilde{\beta}_0 + \widetilde{\beta}_1 g(X) + \varepsilon$, where ε has Pareto like tail, that is

$$Q_\varepsilon(\tau) \sim (1-\tau)^{-1/\alpha}, \quad \tau \to 1$$

and $g(X)$ has lighter tails than ε. The conditional quantiles become

$$Q_{Y|X}(\tau \mid x) = Q_\varepsilon(\tau) + \widetilde{\beta}_0 + \widetilde{\beta}_1 g(x).$$

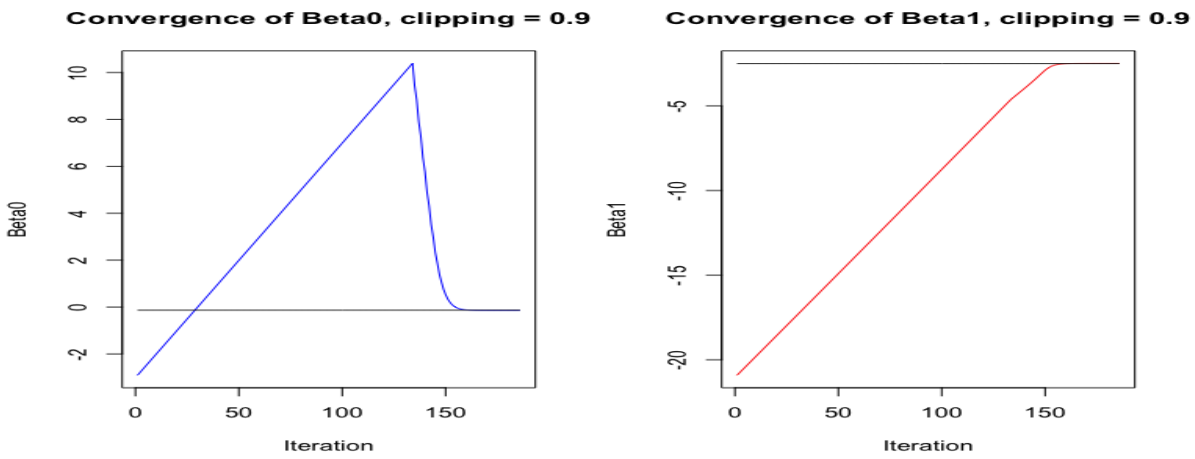

Figure 4.6: Extremal quantile regression with LASSO, with clipping of 0.9. Convergence of β_0 and β_1 estimates. The starting values are $\beta_0^{(0)} = 0$ and $\beta_1^{(0)} = 0$

Hence, the extreme conditional quantile has the form (4.20) with

$$\beta_0(\tau) = \widetilde{\beta}_0 + Q_\varepsilon(\tau), \quad \beta_1(\tau) = \widetilde{\beta}_1$$

with x being replaced with $g(x)$.

Example 4.4.3 (Multiplicative model)**.** Take the multiplicative model

$$Y = \widetilde{\beta}_0 + \widetilde{\beta}_1 X(\varepsilon + 1)$$

where ε has Pareto like tail, that is

$$Q_\varepsilon(\tau) \sim (1-\tau)^{-1/\alpha}, \quad \tau \to 1.$$

Then

$$\mathbb{P}(Y \leq y \mid X = x) = F_\varepsilon\left(\frac{y - \widetilde{\beta}_0}{\widetilde{\beta}_1 x} - 1\right).$$

Therefore,

$$Q_{Y|X}(\tau \mid x) = \widetilde{\beta}_0 + \widetilde{\beta}_1(1 + Q_\varepsilon(\tau))x.$$

Thus, this equates to (4.20) with $\beta_0(\tau) = \widetilde{\beta}_0$ and $\beta_1(\tau) = \widetilde{\beta}_1(1 + Q_\varepsilon(\tau))$.

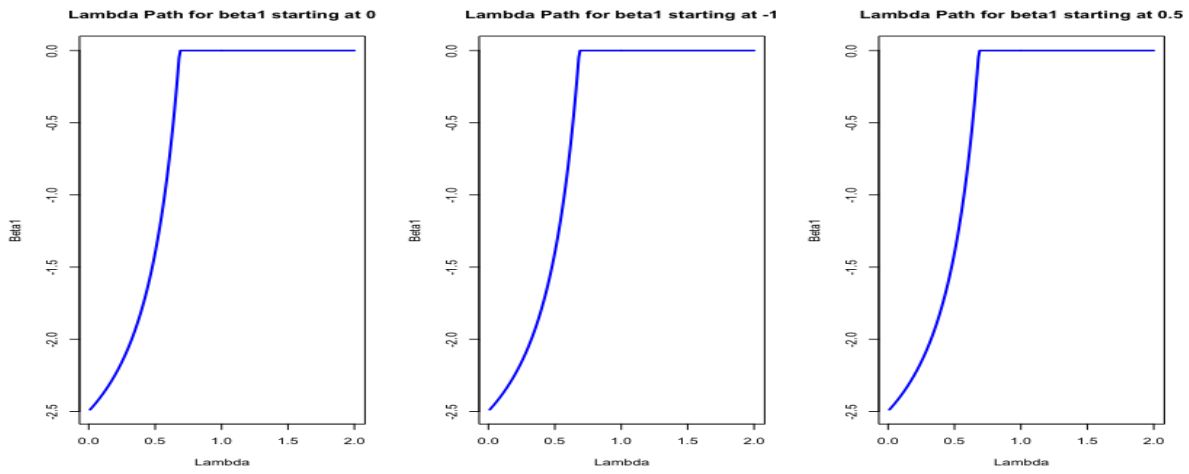

Figure 4.7: Estimation of β_1: LASSO paths for different starting values.

Estimation of extreme conditional quantiles under the common slope assumption. Under the common slope assumption, we suppose that

$$\beta(\tau) = \beta \ , \quad \beta_0(\tau) \sim (1-\tau)^{-1/\alpha}.$$

We note that the linear model leads to such an assumption, since $\beta_0(\tau) = \widetilde{\beta}_0 + Q_\varepsilon(\tau)$ and the latter part dominates when $\tau \to 1$. However, the multiplicative model does not fulfill the common slope assumption.

The idea of the common slope method is to separate the estimation of the slope from the estimation of the intercept:

- Apply the standard quantile regression to $Y - \mathbf{X}^T\beta$ (without including the intercept) to obtain the estimates $\widehat{\beta}$ of β;
- Calculate the residuals $\widehat{\varepsilon}_i := Y_i - \mathbf{X}_i^T\widehat{\beta}$ and model heavy tails of the residuals using the unconditional methods described in Section 4.2.
- Following (4.20), the final estimate of the conditional quantiles is

$$Q_{Y|X}(\tau \mid x) \sim (1-\tau)^{-1/\widehat{\alpha}} + x^T\widehat{\beta}(\tau), \tag{4.23}$$

  where $\widehat{\alpha}$ is an estimator of the tail index α.

We note that this approach allows to apply penalization directly in the standard quantile regression step.

This methodology is implemented in the numerical analysis below.

Example 4.4.4 (True conditional quantiles)**.** In this example, we simulated data $X \sim$ Exp(1) with a sample size of $n = 1000$ and dimension $p = 1$. We then set $\beta_0 = 0$ and

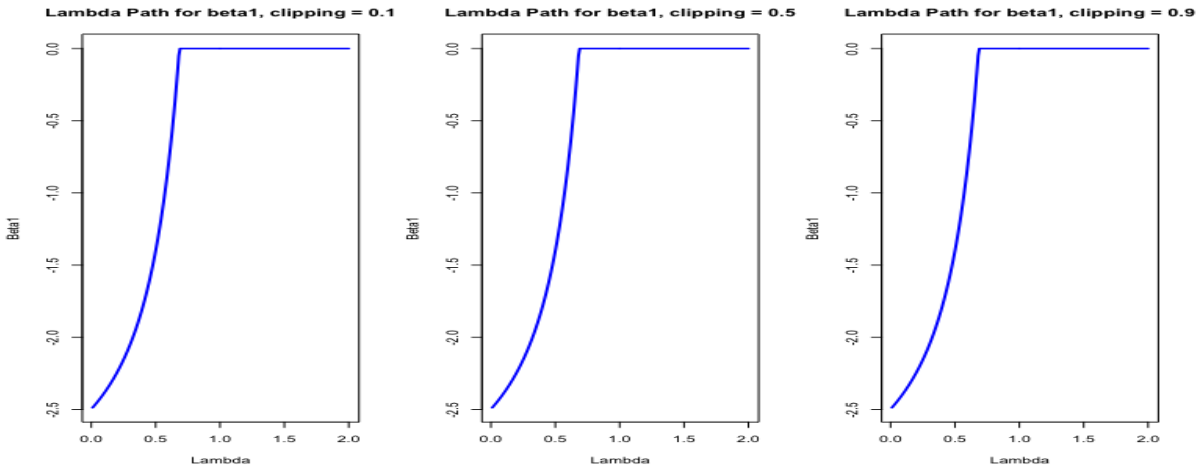

Figure 4.8: Estimation of β_1: LASSO paths for different clipping values.

$\beta_1 = 1$. We simulated the error (ε) from a Pareto distribution with parameter $\alpha = 4$ and so we had the outcome variable $Y = 1 \times X + \varepsilon$. Using the equation derived in (4.21), we found the true conditional quantiles for Y. We calculated the conditional quantiles for values of τ ranging from (0.1 to 0.9). The results for a few values of $X = x$ are displayed in Figure 4.10. We see that the quantile curves have the same shape, but the curves are shifted according to different values of x, which is to be expected. Indeed, the shape of the conditional quantiles does not depend on x.

Example 4.4.5 (Estimation of extreme conditional quantiles using the common slope method)**.** In this example, we have the same setup as in Example 4.4.4, with the exception of τ. In this experiment, we use a $\tau = 0.5, 0.7, 0.9$. Using the `quantreg` package in `R`, we find the β_1 estimates for each τ value. Then using these estimates, we calculate the residuals and calculate the estimated α values. From here, we are able to calculate the estimated extreme conditional quantiles. We display the results in Figure 4.11. Finally, we used the package `EXRQ` to validate our results, as it is the package for the estimation of extreme conditional quantiles using the common slope method. Plotting both sets of estimated extreme conditional quantiles, Figure 4.12, we see that they are practically identical in value.

One aspect to point out is that even if we try to force `quantreg` to set the intercept as zero, it will return its estimate, different for distinct values of τ. The reason for this is very simple. The returned estimate is for $\beta_0(\tau)$, not for $\widetilde{\beta}_0$; cf. (4.22).

In summary, comparing Figures 4.10 and 4.11, we see that our common slope approach correctly estimates the quantile curves. Furthermore, by looking at Figure 4.12, we see that our approach produces almost identical estimates as the package. The advantage of using our method is that we can easily adapt to penalization, which is not possible with

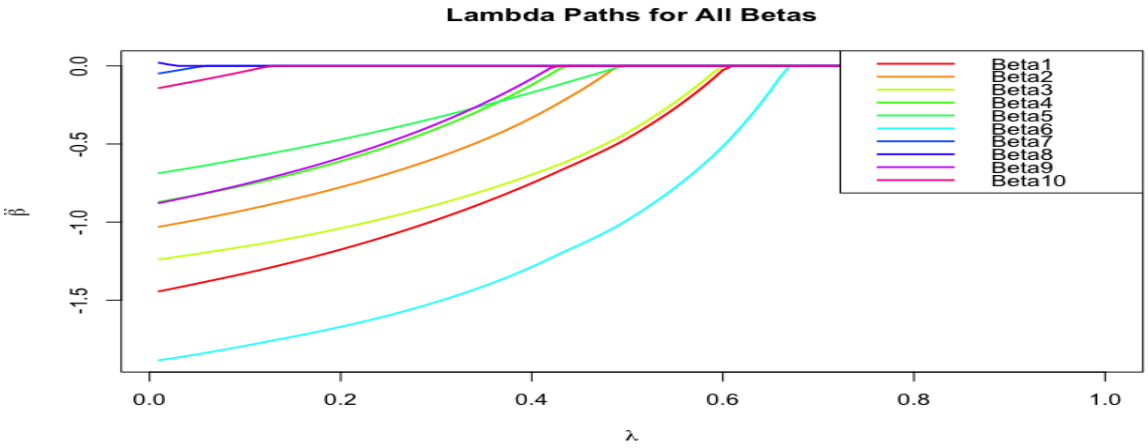

Figure 4.9: LASSO paths for $p = 10$.

the package. In the next example, we will use our method for penalization.

Example 4.4.6 (Estimation of extreme conditional quantiles using the common slope method with LASSO penalization)**.** In this example, we simulated a predictor matrix of size 1000×2, so we have $n = 1000$ and $p = 2$. Each column is generated independently from $X \sim \text{Exp}(1)$. We then set $\beta_0 = 0$, $\beta_1 = 1$, and $\beta_2 = 0.5$. We simulated the error (ε) from a Pareto distribution with parameter $\alpha = 4$ and so we had the outcome variable $Y_j = 1 \times X_{j1} + 0.5 \times X_{j2} + \varepsilon$. Using τ ranging from 0.1 to 0.9 and the `rqPen` package in `R`, we apply a LASSO penalty and find the β_1 and β_2 estimates for each τ value and use cross-validation to find the optimal λ value. Then using these estimates, we calculate the residuals and calculate the estimated α values. From here, we are able to calculate the estimated extreme conditional quantiles. We plot the estimated extreme conditional quantiles for some values of X, which is displayed in Figure 4.13.

Remark 4.4.7. We would like to point out another potential problem with the implementation of the common slope method. Assume again that $Y_j = \widetilde{\beta}_1 \times X_{j1} + \widetilde{\beta}_2 \times X_{j2} + \varepsilon$, but this time ε_j are two-sided Pareto. Then, after estimating the regression coefficients, we calculate residuals. We expect half the residuals to be negative and for the purpose of calculating the tail index (via the Hill estimator) and so, we discard them. If we use LASSO, we introduce bias, and it is possible that all the residuals are negative. One could take the absolute values, but it may not be a good idea if we are interested in the right tail behaviour. In short, the usefulness of LASSO penalization in the common slope model is doubtful.

Summary of penalization methods The penalized likelihood method developed in Section 4.3 and the common slope method with penalization developed above, can both

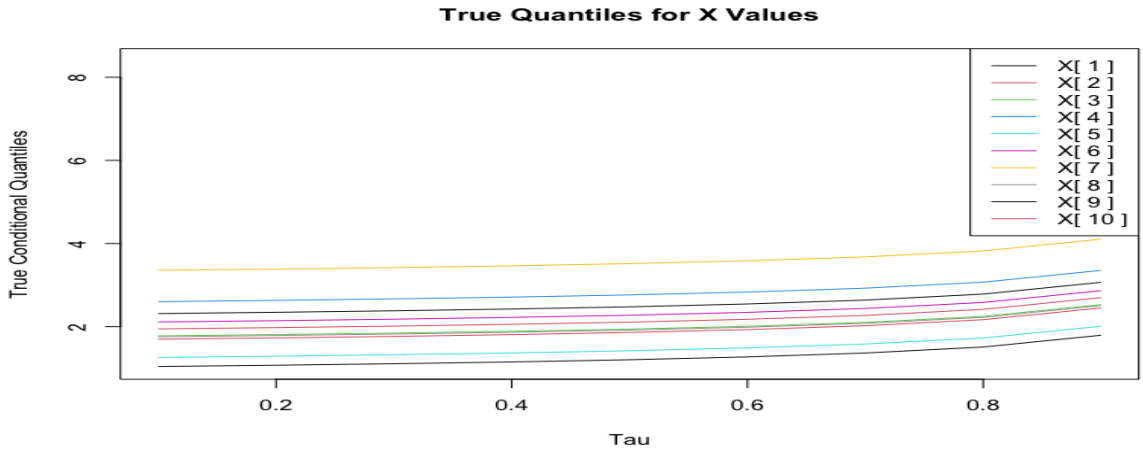

Figure 4.10: True extreme conditional quantiles for the linear model.

be used to find the estimated quantiles and/or the estimated coefficients for extreme quantile regression. However, these methods cannot be compared to one another. The penalized likelihood method assumes that the outcome variable, Y, is Pareto-like with a parameter that incorporates the coefficients (β_0, β_j) and the predictor matrix X. This makes the tail behaviour depend on the predictor X. On the other hand, the common slope method uses the assumption that the tail behaviour comes from the error (ε) and not from the model itself. As such, this method does not have its tail behaviour dependent on X. Furthermore, the coefficients (β_0, β_j) have different meanings for both methods. In other words, for the penalized likelihood method, these coefficients do not have any dependence on τ, but in the common slope method, they do depend on τ. As such, we cannot formulate an experiment that could compare both methods using the same data.

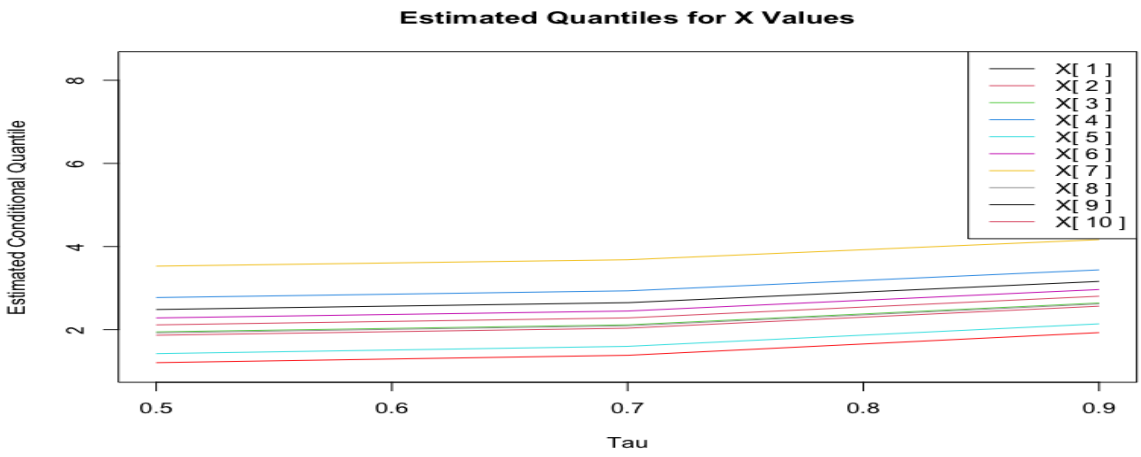

Figure 4.11: Estimated extreme conditional quantiles for respective τ.

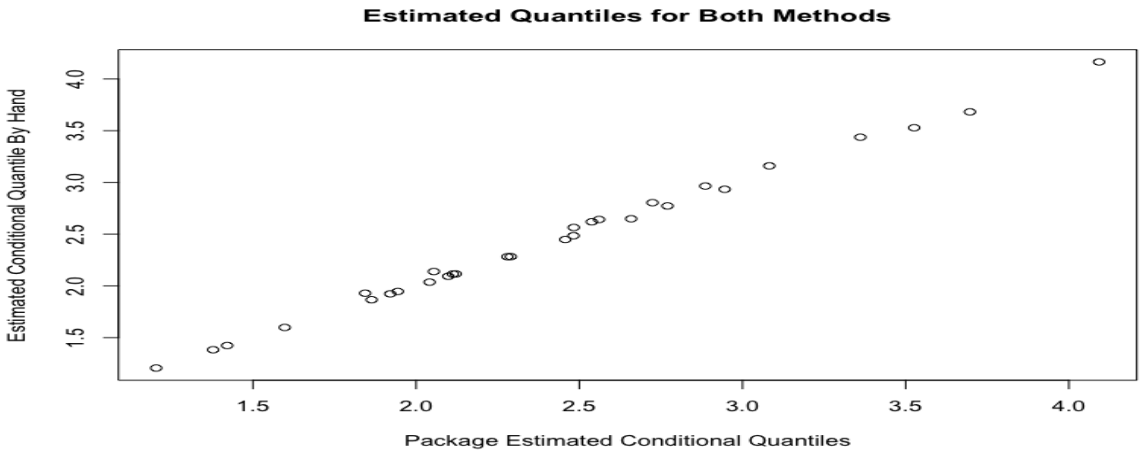

Figure 4.12: Estimated extreme conditional quantiles using the package and using our method.

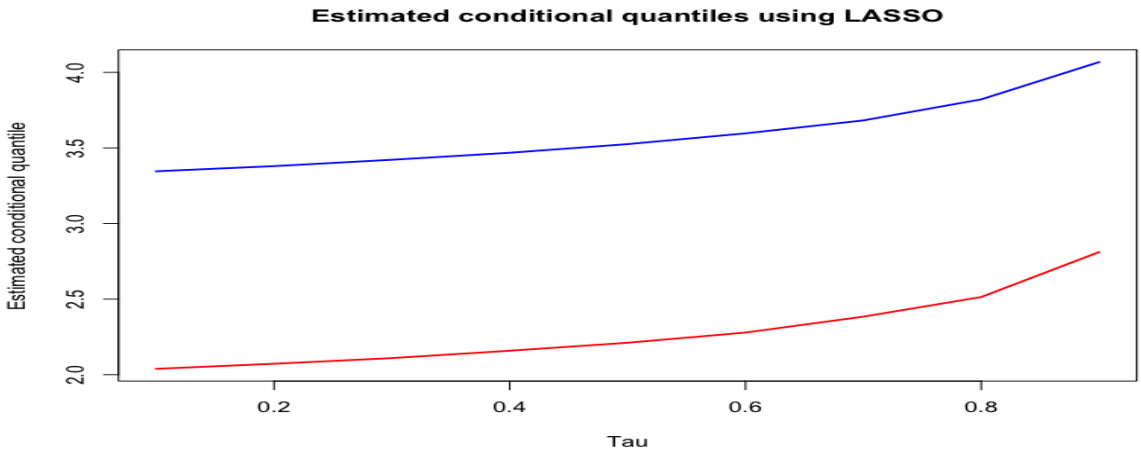

Figure 4.13: Estimated extreme conditional quantiles for respective τ.

Chapter 5

Real data examples

We performed six experiments using real data to be able to better demonstrate how some of the techniques shown in this thesis are applied in real-world settings. We used NFL data taken from [1]. We used the same dataset for all the experiments that we conducted and used statistical software `R` along with our own codes to do all the analyses. We performed both linear and quantile regression, repeated these using a LASSO penalty, and finished this series of experiments by plotting the quantile boxplots. These analyses were to see the relationship between the total points scored in a year (dependent variable) and the total yards, the total offensive plays, the total turnovers, the total number of first downs, and the total number of penalty yards in a year.

The dataset is composed of 10 variables with 160 observations. The variables in the dataset are: team, year ($2020 - 2024$), games played, total points scored in a year, the total yards in a year, the total offensive plays in a year, the total turnovers in a year, the total number of first downs in a year, and the total number of penalty yards in a year.

Prior to performing the analyses, we found a few basic statistics to get a better understanding of the data.

Statistic	**Minimum**	**Maximum**	**Mean**
Total Yards	4479	7224	5744
Total Points	236	564	383.5
Total Offensive Plays	940	1187	1060
Total Turnovers	8	37	21.71
Total First Downs	2	410	143.7
Total Penalty Yards	75	1134	682.2

Table 5.1: Summary Statistics for Team Performance

5.1 Experiment 1: Linear regression

As we know, linear regression will give us the mean behaviour of the relationship between these predictors. In other words, the results of this analysis will determine the coefficients of the predictors and their importance, if a specific team wants to score an average number of points. Using the function `lm`, we determined the coefficients and whether or not they are significant. The resulting equation is :

$$\begin{aligned}\text{total points} = &-83.18 + 0.11 \times \text{total yards} - 0.085 \times \text{total offensive plays} \\ &- 2.50 \times \text{total turnovers} + 0.041 \times \text{total first downs} \\ &- 0.013 \times \text{total penalty yards.}\end{aligned}$$

Using an $\alpha = 0.05$ and looking at the p-values, we determined that the following variables are significant: **total number of yards** (with the p-value of $< 2 \times 10^{-16}$), the **total number of turnovers** (with a p-value of 9.15×10^{-7}), and the **total number of first downs** (with a p-value of 0.0148). As such, if a specific team wants to have an average number of points scored in a year, they should focus on the total yards, the number of turnovers, and the first downs.

5.2 Experiment 2: Linear regression with LASSO penalty

Following the linear regression, we applied a LASSO penalty using the `glmnet` package to do variable selection. This being said, the LASSO penalty will select the predictors that are most important for a team to have an average number of points in a year. To find the coefficients and the optimal λ, we performed cross-validation using the function `cv.glmnet`. As a result, the optimal λ is 0.178 and resulting equation is:

$$\begin{aligned}&\text{total points} = \\ &-123.57 + 0.096 \times \text{total yards} - 2.06 \times \text{total turnovers} + 0.011 \times \text{total first downs.}\end{aligned}$$

As we can see, the predictors selected are the total yards, the total turnovers, and the total first downs.

5.3 Experiment 3: Quantile regression

Contrary to linear regression, quantile regression determines the relationship between the predictors at a given quantile. Thus, if a team wanted to be, for example, in the top 90th percentile for the total number of points, then we would need to set $\tau = 0.9$ and observe the coefficients and their statistical significance. We performed this experiment using different quantiles ranging from 0.1 to 0.9. Below are the results, where $Y =$ total

points, X_1 = total yards, X_2 = total offensive plays, X_3 = total turnovers, X_4 = total first downs, and X_5 = total penalty yards:

Table 5.2: Quantile Regression Equations for Different τ

τ	Regression Equation
0.1	$Y = -26.60 + 0.10X_1 - 0.14X_2 - 2.02X_3 + 0.04X_4 - 0.01X_5$
0.2	$Y = -87.93 + 0.10X_1 - 0.08X_2 - 2.64X_3 + 0.05X_4 - 0.02X_5$
0.3	$Y = -56.03 + 0.11X_1 - 0.14X_2 - 2.04X_3 + 0.01X_4 - 0.00X_5$
0.4	$Y = -75.56 + 0.11X_1 - 0.12X_2 - 2.03X_3 + 0.04X_4 - 0.00X_5$
0.5	$Y = -127.02 + 0.11X_1 - 0.06X_2 - 2.23X_3 + 0.04X_4 - 0.01X_5$
0.6	$Y = -111.76 + 0.11X_1 - 0.06X_2 - 2.42X_3 + 0.03X_4 - 0.02X_5$
0.7	$Y = -41.99 + 0.11X_1 - 0.13X_2 - 2.42X_3 + 0.05X_4 - 0.02X_5$
0.8	$Y = -72.45 + 0.11X_1 - 0.06X_2 - 2.81X_3 + 0.06X_4 - 0.03X_5$
0.9	$Y = -93.18 + 0.10X_1 + 0.02X_2 - 3.88X_3 + 0.06X_4 - 0.02X_5$

We make several observations:

- We note that the coefficients in the quantile regression do not depend too much on the level of τ as the values for the coefficients do not change much for the different levels of τ.

- As we indicated in the thesis, there is no reason that the coefficients obtained in the linear regression context will match those in the quantile regression.

To determine the significant predictors, we look at the confidence intervals obtained by doing the regression. If they contain 0 then, the predictor is not statistically significant. Focusing on $\tau = 0.9$, we see that the significant variables are: the total yards, the total turnovers, the total first downs, and the total penalty yards. The total offensive plays are not significant. We note also that for different τ's, different predictors are significant.

5.4 Experiment 4: Quantile regression with LASSO

Following the quantile regression, we applied a LASSO penalty using the package `rqPen` in `R` using a $\tau = 0.9$ to do variable selection. Thus, the results would give us the coefficients for the predictors selected by the LASSO method for a specific team to be in the 90th percentile of total points. To find the optimal lambda, we performed cross-validation using the function `rq.pen.cv` and found $\lambda = 0.0016$. The equation that went

along with this optimal λ was:

$$\begin{aligned}\text{total points} = &-119.66 + 0.102 \times \text{total yards} + 0.034 \times \text{total offensive plays} \\ &- 3.57 \times \text{total turnovers} + 0.063 \times \text{total first downs} - 0.02 \times \text{total penalty yards}.\end{aligned}$$

As we can see, the variables selected are the same as in the original model. Since there is no difference between the variables selected in the non-penalized and penalized approach, LASSO is not of particular use here.

5.5 Experiment 5: Quantile boxplots

The last experiment that we performed using this NFL data was creating boxplots that displayed the total points by total yards with three quantiles; $0.25, 0.5, 0.75$. This boxplot 5.1 shows that there is a linear relationship between the total points and yards. The red line is when $\tau = 0.25$, the blue is when $\tau = 0.5$, and the green is when $\tau = 0.75$.

5.6 Experiment 6: Quantile boxplots for a modified NFL dataset

We would like to better understand the role of potential outliers on the estimation procedure. For this, we selected the top 5 teams in terms of the total number of points, and for these teams, we artificially added 2000 yards. This has little effect on the linear regression model, it has some effect on the low quantiles, but (as expected) it has a huge effect on the upper quantiles. In other words, the outliers in the predictors will not affect the estimation of the linear model too much but will have a strong effect on the top quantiles. See Figure 5.2.

5.7 Summary

After conducting these experiments, we can see that this data follows a linear model and so linear regression worked well. Quantile regression produced the equations that were similar to linear regression, which could mean that the factors that influence the average performance also could be the ones that distinguish top teams. Furthermore, there were not many outliers affecting particular quantiles, which can be seen in the boxplots in Figure 5.1. Also, as we did not see a change in the different quantile levels, we can conclude that when trying to improve the total number of points scored, there is not one particular variable on its own that will improve the score but the combination of them will. Finally, for the case of quantile regression and LASSO, we can say that using the LASSO penalty was not particularly useful as it simply just added bias and produced approximately the same equation as the standard quantile regression.

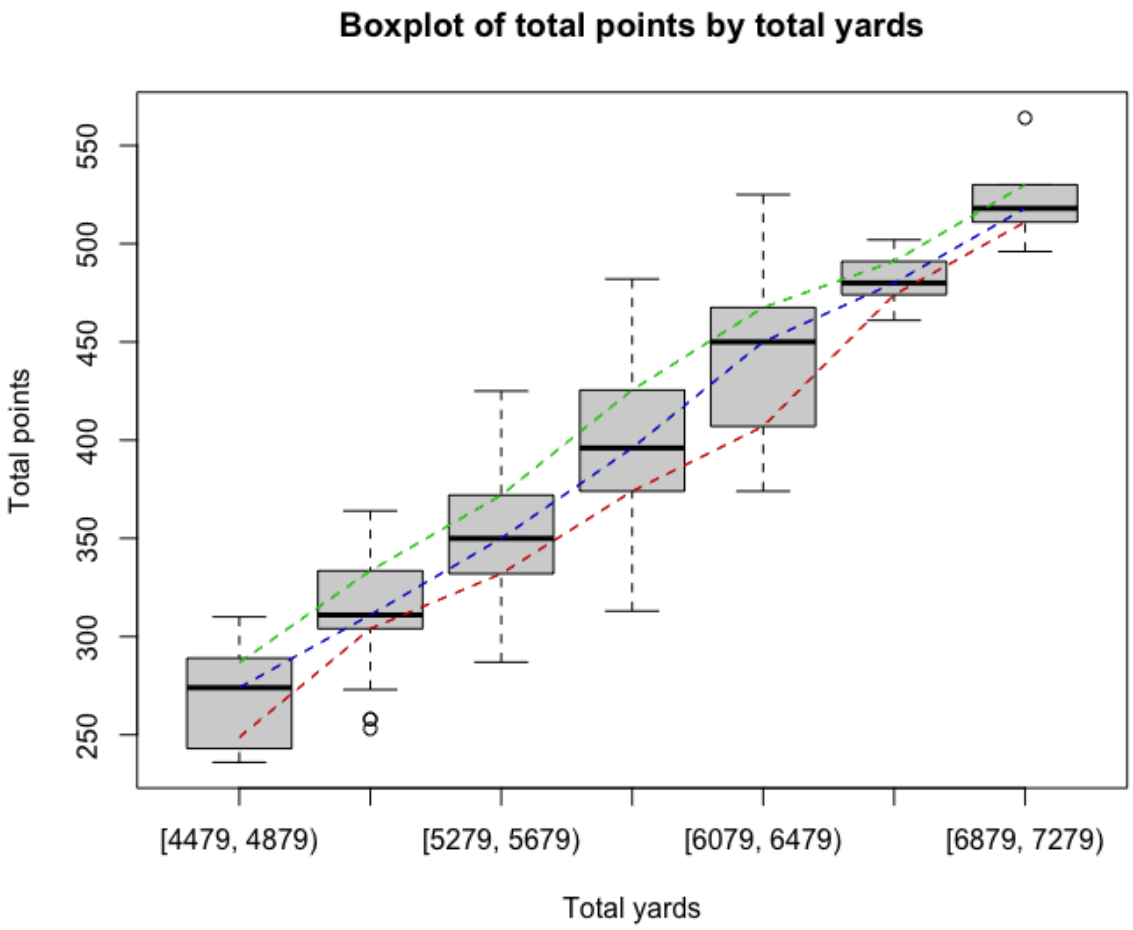

Figure 5.1: Boxplot comparison.The red line is when $\tau = 0.25$, the blue is when $\tau = 0.5$, and the green is when $\tau = 0.75$. The quantiles are estimated from the model.

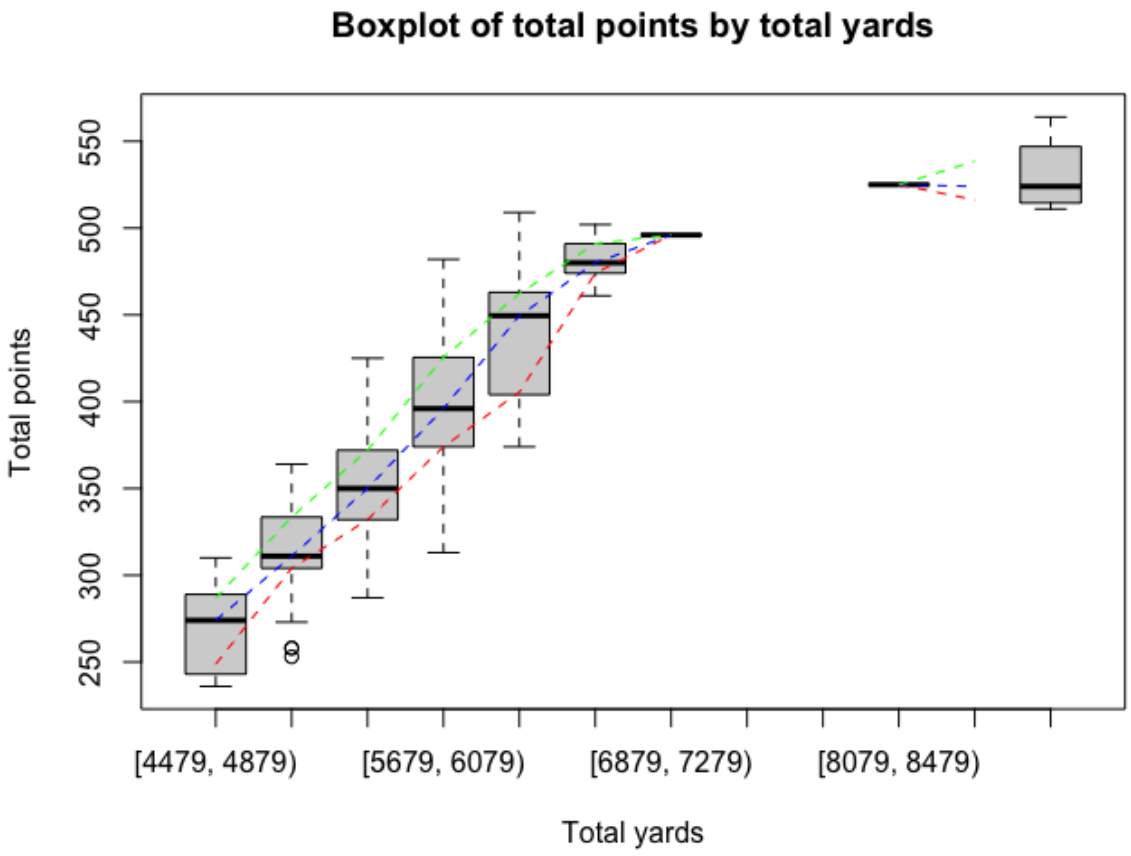

Figure 5.2: Boxplot comparison.

Chapter 6

Summary and future directions

6.1 Summary of results

This thesis develops and evaluates methodologies for extreme quantile regression in high-dimensional contexts. More specifically, it introduces two approaches: penalized likelihood approach and the penalized common slope method. We provide an extensive review and assessment of existing methods for solving quantile regression problems and explore how optimization techniques can be applied to high-dimensional data cases using quantiles. The methodologies assessed and developed in this thesis were tested both in simulation and on a real-world dataset from the NFL.

6.2 Contributions

Most of the theoretical and methodological foundations presented in the thesis stem from the literature cited below. However:

- All the theoretical examples were fully developed by the author;
- All the numerical studies were developed by the author;
- The specific implementation of the penalized quantile regression, based on linear programming, was developed by the author;
- The entire methodology of both the penalized likelihood approach and the penalized common slope method for extreme conditional quantiles were developed by the author, correcting some issues present in the literature.

6.3 Strengths and limitations

Strengths

A big strength of this thesis is the development of methodologies for penalized extreme quantile regression. These methods were supported by both theoretical justification and numerical evidence, providing options for users depending on their data.

6.3.1 Limitations

Despite these strengths, the thesis has limitations as well:

- The penalized likelihood and common slope approaches are based on different assumptions, making direct comparisons between them challenging.
- The implementation of the common slope model with penalties may introduce significant bias to the estimation of the tail behaviour.

6.3.2 Future directions

Several directions for future research emerges from this work. These are listed below:

- Applying these methodologies to higher dimensional real data.
- Extending our study to the case of $p = p_n$ where the dimension is dependent on n.
- Developing limiting theory to construct confidence intervals for both penalized likelihood method and common slope.

Chapter 7

R codes

7.1 Codes for Chapter 2

7.1.1 Example 2.1.4

```
#Simulating N=1000 times to see if they differ
N <- 1000 #Number of iterations
n <- 100 #Sample size
q_tilda <- c() #To store values from equation 2.5
q_hat <- c() #To store values from equation 2.4
H <- function(x, u, tau) { #loss function
  result <- numeric(length(u))
  for (i in seq_along(u)) {
    L <- (tau - 1)*sum(x[x < u[i]] - u[i]) + tau*sum((x[x >= u[i]] - u[i]))
    result[i]<-L
  }
  return(result)
}
for(i in 1:N) {
 Y <- rnorm(n) #Outcome variable
 #All possible values that minimize the loss
 u <- seq(min(Y), max(Y), length.out = n)
 tau <- 0.5 #Quantile of interest
 q_tilda[i] <- u[which.min(H(Y, u, tau))] #Quantile values from equation 2.5
 q_hat[i] <- quantile(Y,0.5) #Quantile values from equation 2.4
}

#Plot to see if they are similar
plot(q_tilda,q_hat)
```

7.2 Codes for Chapter 3

7.2.1 Example 3.2.3

```
#Coordinate descent algorithm
coordinate.ridge <- function(X, Y, lambda, delta, nmax) {
  n <- ncol(X) #Sample size
  p <- nrow(X)  # Number of predictors
  b <- rep(0, p)  # Initialize coefficients
  #Standardize the data
  X_std <- scale(t(X))  # Standardize columns of X (transposed for correct dimension)
  X_std <- t(X_std)     # Transpose back to maintain consistency
  X_sds <- apply(t(X), 2, sd) #standard deviation of original data
  iteration <- 0
  while (iteration < nmax) {
    b_old <- b #For convergence purposes
    for (j in 1:p) {
      gamma <- Y - b %*% X_std + b[j] * X_std[j, ]
      # Update the coefficient for feature j
      b[j] <- sum(gamma * X_std[j, ]) / (sum(X_std[j, ]^2) + lambda)
    }
    iteration <- iteration + 1
    # Check for convergence
    if (sqrt(sum((b - b_old)^2)) < delta) {
      break
    }
  }
  #Unstandardizing to get coefficients
  b <- b / X_sds
  return(b)
}
#Comparing the direct method and the coordinate descent method to
#calculate beta estimates
delta <- 1e-6
nmax <- 1000 #Maximum iterations
lambda <- 0.5
ridge_beta_cd <- matrix(0,N,2) #To store coordinate descent method values
ridge_beta <- matrix(0,N,2) #To store direct method values
N <- 1000
n <- 100
for (i in 1:N){
  X <- matrix(rnorm(200), 2, n)  # pxn matrix
  Z <- rnorm(n)  # Noise
  beta <- c(2, 1)  # True coefficients
  Y <- beta %*% X + Z #Outcome variable
```

```
  I.2 <- diag(1, nrow(X))  # Identity matrix of size p x p
  #Coordinate descent method
  ridge_beta_cd[i,] <- coordinate.ridge(X, Y, lambda, delta, nmax)
  ridge_beta[i,] <- (solve(X%*%t(X)+ lambda*I.2))%*%X%*%t(Y) #Direct method
}
# Compare results for both predictors
plot(ridge_beta_cd[,1],ridge_beta[,1])
plot(ridge_beta_cd[,2],ridge_beta[,2])
```

7.2.2 Example 3.3.1

```
#Comparing estimates for quantile regression between lpSolve package and quantreg
#Loading packages
library(lpSolve)
library(quantreg)

#Initializing
N <- 1000 #Number of iterations
n <-100 #Sample size
tau <- 0.5 #Quantile wanted
beta_lp <- matrix(0,N,2) #To store lp solve values
beta_qr <- matrix(0,N,2) #To store quantreg values
p <- 2 #Number of predictors

for (i in 1:N){
  X <- rnorm(n) #Predictors
  X.1 <- cbind(rep(1,n),X) #Predictors for lpSolve
  Z <- rnorm(n) #Error
  Y <- 2*X.1[,2]+Z #Outcome
  b <- Y #For lpSolve
  A <- cbind(X.1,-X.1,diag(n),-diag(n))
  c <- c(rep(0,2*p),tau*rep(1,n),(1-tau)*rep(1,n))
  linprog <- lp("min",c,A,"=",b)
  beta_lp[i,] <- linprog$sol[1:p]-linprog$sol[(1:p+p)] #lpSolve technique
  beta_qr[i,] <- rq(Y~X)$coeff #quantreg technique
}

par(mfrow=c(2,1))
hist(beta_lp[,2])
hist(beta_qr[,2])
qqplot(beta_lp[,2],beta_qr[,2])
```

7.2.3 Example 3.3.2

```
#Smoothed function for quantile regression
```

```
beta <- 2 #True beta value
X <- rnorm(100) #Predictors
Z <- rnorm(100) #Error
Y <- X*beta + Z #Outcome
tau <- 0.5 #Wanted quantile
k <- 1000
smoothed_quantreg <- function(X,Y,tau,k,b) {
  n <- length(X) #Sample size
  u <- Y-X*b #Residuals
  A <- X*Y
  B <- X^2
  part1 <- -(1/(n*k))*sum(A[u >= -(1-tau) & u <= tau*k])
  part2 <- (b/(n*k))*sum(B[u >= -(1-tau) & u <= tau*k])
  part3 <- -(tau/n)*sum(X[u > tau*k])
  part4 <- ((1-tau)/n)*sum(X[-(1-tau)*k > u])
  out <- part1 +part2 + part3 + part4
}
betas <- seq(0, 3, by=0.1) #Grid of possible beta values
L <- numeric(length(betas)) #To store the values
for (i in 1:length(betas)) {
  L[i] <- smoothed_quantreg(X,Y,tau,k,betas[i]) #Iterating through each beta value
}
plot(betas,L,type="l")
```

7.3 Codes for Chapter 4

7.3.1 Example 4.3.1

```
#Coordinate descent algorithm to find beta0 and beta1 recursively, for p=1
#Initializing data
n <- 1000 #Sample size
X <- rexp(n,1) #Predictors simulated from exponential(1)
alpha <- 4
true.beta1 <- -2.5 #True beta1 value
true.beta0 <- -log(alpha/(1-true.beta1)) #True beta0 value
Y <- NULL
for (i in 1:n){
  Y[i] <- (runif(1))^(-exp(true.beta0+true.beta1*X[i])) #Outcome variable
}

coordinate_descent <- function(X,Y,lambda,gamma0,gamma1,delta,nmax){
  #Storing all beta0 and beta1 values to be able to plot them
  beta0_all <- NULL
  beta1_all <- NULL
```

```
  n <- length(Y) #Sample size

  iteration <- 0
  while( iteration < nmax) {
    gamma0_old <- gamma0
    gamma1_old <- gamma1

    Z <- (1/n)*sum(exp(-gamma1*X)*log(Y))
    V_1 <- (1/n)*sum((X^1)*exp(-gamma1*X)*log(Y))
    V_2 <- (1/n)*sum((X^2)*exp(-gamma1*X)*log(Y))

    beta0 <- 1 + gamma0 - 1/(exp(-gamma0)*Z)
    beta1 <- (gamma1*exp(-gamma0)*V_2 - mean(X) + exp(-gamma0)*V_1)/
    (exp(-gamma0)*V_2 + lambda)

    gamma0 <- beta0
    gamma1 <- beta1

    beta0_all <- c(beta0_all, beta0)
    beta1_all <- c(beta1_all, beta1)

    # Check for convergence
    if (sqrt((gamma1- gamma1_old)^2+(gamma0 - gamma0_old)^2) < delta) {
      cat("Converged at iteration:", iteration + 1, "\n")
      break
    }
    iteration <- iteration + 1
 }
  #Plotting all the beta0 and beta1 values
  m <- length(beta0_all)
  o <- length(beta1_all)
  par(mfrow=c(1,2))
  plot(beta0_all, type="l", col="blue", main="Convergence of Beta0",
  xlab="Iteration", ylab="Beta0")
  lines(rep(true.beta0,m),type="l", col="black")
  plot(beta1_all, type="l", col="red", main="Convergence of Beta1",
  xlab="Iteration", ylab="Beta1")
  lines(rep(true.beta1,o),type ="l", col="black")
  return(list(beta0=gamma0, beta1 = gamma1))
}

#Initializing values for coordinate descent
gamma0 <- 0 #Starting value for beta0
gamma1 <- 0 #Starting value for beta1
```

```
lambda <- 0.01
delta <- 1e-6
nmax <- 1000 #Maximum iterations
coordinate_descent(X,Y,lambda,gamma0,gamma1,delta,nmax)
print(c(true.beta0,true.beta1)) #Printing to compare the estimates to the true value
```

7.3.2 Example 4.3.2

```
#Coordinate descent: find beta0 and beta1 recursively with clipping, for p=1
#Initializing data
n <- 1000 #Sample size
X <- rexp(n,1) #Predictors simulated from exponential(1)
alpha <- 4
true.beta1 <- -2.5 #True beta1 value
true.beta0 <- -log(alpha/(1-true.beta1)) #True beta0 value
Y <- NULL
for (i in 1:n){
  Y[i] <- (runif(1))^(-exp(true.beta0+true.beta1*X[i])) #Outcome variable
}

coordinate_descent <- function(X,Y,lambda,gamma0,gamma1,delta,nmax,clipping1){
  #Storing all beta0 and beta1 values to be able to plot them
  beta0_all <- NULL
  beta1_all <- NULL
  n <- length(Y) #Sample size

  iteration <- 0
  while( iteration < nmax) {
    gamma0_old <- gamma0
    gamma1_old <- gamma1

    Z <- (1/n)*sum(exp(-gamma1*X)*log(Y))
    V_1 <- (1/n)*sum((X^1)*exp(-gamma1*X)*log(Y))
    V_2 <- (1/n)*sum((X^2)*exp(-gamma1*X)*log(Y))

    change <- 1/(exp(-gamma0)*Z) #Ratio we want to clip
    change.1 <- max(change,clipping1) #Clipping the ratio
    beta0 <- 1 + gamma0 -change.1 #Calculating beta0 with the clipped ratio
    beta1 <- (gamma1*exp(-gamma0)*V_2 - mean(X) + exp(-gamma0)*V_1)/
    (exp(-gamma0)*V_2 + lambda)

    gamma0 <- beta0
    gamma1 <- beta1
    beta0_all<- c(beta0_all,beta0)
    beta1_all<- c(beta1_all,beta1)
```

```
    # Check for convergence
    if (sqrt((gamma1- gamma1_old)^2+(gamma0-gamma0_old)^2) < delta) {
      cat("Converged at iteration:", iteration + 1, "\n")
      break
    }
    iteration <- iteration + 1
  }
  #Plotting beta0 and beta1 values
  m=length(beta0_all);
  par(mfrow=c(1,2))
  plot(beta0_all, type="l", col="blue", main="Convergence of Beta0",
  xlab="Iteration", ylab="Beta0")
  lines(rep(true.beta0,m),type="l",col="black")
  plot(beta1_all, type="l", col="red", main="Convergence of Beta1",
  xlab="Iteration", ylab="Beta1")
  lines(rep(true.beta1,m),type="l",col="black")
  return(list(beta0=gamma0, beta1 = gamma1))
}

#Initializing values for coordinate descent
gamma0 <- 0 #Starting value for beta0
gamma1 <- 0 #Starting value for beta1
lambda <- 0.01
delta <- 1e-6
nmax <- 1000 #Number of iterations
clipping1 <- 0.9 #Clipping value
coordinate_descent(X,Y,lambda,gamma0,gamma1,delta,nmax,clipping1)
print(c(true.beta0,true.beta1))
```

7.3.3 Example 4.3.3

```
#Coordinate descent: find beta0 and beta1 recursively with clipping, for p=1
#Initializing data
n <- 1000 #Sample size
X <- rexp(n,1) #Predictors simulated from exponential(1)
alpha <- 4
true.beta1 <- -2.5 #True beta1 value
true.beta0 <- -log(alpha/(1-true.beta1)) #True beta0 value
Y <- NULL
for (i in 1:n){
  Y[i] <- (runif(1))^(-exp(true.beta0+true.beta1*X[i])) #Outcome variable
}

soft_thresholding <- function(t,lambda){ #Soft-thresholding function
```

```
  st <- sign(t)*pmax(0,abs(t)-lambda)
  return(st)
}

coordinate_descent <- function(X,Y,lambda,gamma0,gamma1,delta,nmax,clipping1){
  #Storing all beta0 and beta1 values to be able to plot them
  beta0_all <- NULL
  beta1_all <- NULL

  n <- length(Y) #Sample size
  iteration <- 0

  while( iteration < nmax) {
    gamma0_old <- gamma0
    gamma1_old <- gamma1

    Z <- (1/n)*sum(exp(-gamma1*X)*log(Y))
    V_1 <- (1/n)*sum((X^1)*exp(-gamma1*X)*log(Y))
    V_2 <- (1/n)*sum((X^2)*exp(-gamma1*X)*log(Y))

    change <- 1/(exp(-gamma0)*Z) #Ratio we want to clip
    change.1 <- max(change,clipping1) #Clipping the ratio
    beta0 <- 1 + gamma0 -change.1 #Calculating beta0 with the clipped ratio
    beta1 <- soft_thresholding((gamma1*exp(-gamma0)*V_2 - mean(X)
    + exp(-gamma0)*V_1),lambda)/(exp(-gamma0)*V_2)

    gamma0 <- beta0
    gamma1 <- beta1
    beta0_all<- c(beta0_all,beta0)
    beta1_all<- c(beta1_all,beta1)

    # Check for convergence
    if (sqrt((gamma1- gamma1_old)^2+(gamma0-gamma0_old)^2) < delta) {
      cat("Converged at iteration:", iteration + 1, "\n")
      break
    }
    iteration <- iteration + 1
  }
  #Plotting beta0 and beta1 values
  m=length(beta0_all);
  par(mfrow=c(1,2))
  plot(beta0_all, type="l", col="blue", main="Convergence of Beta0, clipping = 0.9",
  xlab="Iteration", ylab="Beta0")
  lines(rep(true.beta0,m),type="l",col="black")
```

```
  plot(beta1_all, type="l", col="red", main="Convergence of Beta1, clipping = 0.9",
  xlab="Iteration", ylab="Beta1")
  lines(rep(true.beta1,m),type="l",col="black")
  return(list(beta0=gamma0, beta1 = gamma1))
}

#Initializing values for coordinate descent and the different clipping
gamma0 <- 0 #Starting value for beta0
gamma1 <- 0 #Starting value for beta1
lambda <- 0.01
delta <- 1e-6
nmax <- 1000 #Number of iterations
clipping1 <- 0.9 #Clipping values
coordinate_descent(X,Y,lambda,gamma0,gamma1,delta,nmax,clipping1)
print(c(true.beta0,true.beta1))
```

7.3.4 Example 4.3.4

```
##Part 1: Different starting values
#Initializing the parameters for all experiments
par(mfrow=c(1,3)) #To be able to plot all 3 plots that we are comparing
clipping <- 0.5 #Clipping value
lambdas<- seq(0.01, 2, by =0.01) #Setting lambdas

gamma0 <- 0 #Starting value for beta0
gamma1 <- 0 #Starting value for beta1
a <- NULL
for (i in 1:length(lambdas)){
#Calculating the different values of beta1 for each different lambda
a[i]<-coordinate_descent(X,Y,lambdas[i],gamma0,gamma1,delta,nmax,clipping1)$beta1
}
plot(lambdas, a, type = "l", col = "blue", lwd = 2,
     xlab = "Lambda", ylab = "Beta1",
     main = "Lambda Path for beta1 starting at 0")

gamma0 <- -1 #Starting value for beta0
gamma1 <- -1 #Starting value for beta1
a <- NULL
for (i in 1:length(lambdas)){
#Calculating the different values of beta1 for each different lambda
a[i]<-coordinate_descent(X,Y,lambdas[i],gamma0,gamma1,delta,nmax,clipping1)$beta1
}
plot(lambdas, a, type = "l", col = "blue", lwd = 2,
     xlab = "Lambda", ylab = "Beta1",
     main = "Lambda Path for beta1 starting at -1")
```

```
gamma0 <- 0.5 #Starting value for beta0
gamma1 <- 0.5 #Starting value for beta1
a <- NULL
for (i in 1:length(lambdas)){
#Calculating the different values of beta1 for each different lambda
a[i]<-coordinate_descent(X,Y,lambdas[i],gamma0,gamma1,delta,nmax,clipping1)$beta1
}
plot(lambdas, a, type = "l", col = "blue", lwd = 2,
     xlab = "Lambda", ylab = "Beta1",
     main = "Lambda Path for beta1 starting at 0.5")

##Part 2: Different clipping values
#Initializing the parameters for all experiments
par(mfrow=c(1,3)) #To be able to plot all 3 plots that we are comparing
gamma0 <- 0 #Starting value for beta0
gamma1 <- 0 #Starting value for beta1
lambdas<- seq(0.01, 2, by =0.01) #Setting lambdas
clipping1 <- 0.1 #Clipping value
a <- NULL
for (i in 1:length(lambdas)){
#Calculating the different values of beta1 for each different lambda
a[i]<-coordinate_descent(X,Y,lambdas[i],gamma0,gamma1,delta,nmax,clipping1)$beta1
}
plot(lambdas, a, type = "l", col = "blue", lwd = 2,
     xlab = "Lambda", ylab = "Beta1",
     main = "Lambda Path for beta1, clipping = 0.1")

clipping1 <- 0.5 #Clipping value
a <- NULL
for (i in 1:length(lambdas)){
#Calculating the different values of beta1 for each different lambda
a[i]<-coordinate_descent(X,Y,lambdas[i],gamma0,gamma1,delta,nmax,clipping1)$beta1
}
plot(lambdas, a, type = "l", col = "blue", lwd = 2, xlab = "Lambda", ylab = "Beta1",
     main = "Lambda Path for beta1, clipping = 0.5")

clipping1 <- 0.9 #Clipping values
a <- NULL
for (i in 1:length(lambdas)){
#Calculating the different values of beta1 for each different lambda
a[i]<-coordinate_descent(X,Y,lambdas[i],gamma0,gamma1,delta, nmax, clipping1)$beta1
}
plot(lambdas, a, type = "l", col = "blue", lwd = 2,
```

```
     xlab = "Lambda", ylab = "Beta1",
     main = "Lambda Path for beta1, clipping = 0.9")
```

7.3.5 Example 4.3.5

```
#Coordinate descent: find beta0 and betas recursively for, with clipping with p=10
#Initializing data
n <- 1000 #Sample size
p <- 10 #Number of predictors
X <- matrix(rexp(p*n,1),n,p) #Predictors simulated from exponential(1)
true.beta1 <- -1.5 #True beta1 value
true.beta2 <- -1 #True beta2 value
true.beta3 <- -1.2 #True beta3 value
true.beta4 <- -0.9 #True beta4 value
true.beta5 <- -0.6 #True beta5 value
true.beta6 <- -1.9 #True beta6 value
true.beta7 <- -0.03 #True beta7 value
true.beta8 <- -0.0005 #True beta8 value
true.beta9 <- -0.9 #True beta9 value
true.beta10 <- -0.11 #True beta10 value
true.beta0 <- -0.12 #True beta0 value
Y <- NULL
for (i in 1:n){ #Outcome variable
  Y[i] <- (runif(1))^(-exp(true.beta0+true.beta1*X[i,1]+true.beta2*X[i,2]+
  true.beta3*X[i,3]+true.beta4*X[i,4]+true.beta5*X[i,5]+true.beta6*X[i,6]+
  true.beta7*X[i,7]+true.beta8*X[i,8]+true.beta9*X[i,9]+true.beta10*X[i,10]))
}
#plot.ts(Y)

soft_thresholding <- function(t,lambda){ #Soft-thresholding function
  st <- sign(t)*pmax(0,abs(t)-lambda)
  return(st)
}
coordinate_descent_p <- function(X, Y, lambda, gamma0, gammas,
delta, nmax, which.to.plot,clipping1){
  n <- length(Y) #Sample size
  p <- ncol(X) #Number of predictors
  #Standardizing the data
  X_std <- scale(X)
  X_means <- colMeans(X)
  X_sds <- apply(X, 2, sd)
  #Storing all betas that we wish to plot
  betas_to_plot <- matrix(NA,nmax,length(which.to.plot))

  iteration <- 0
```

```
while( iteration < nmax) {
  gammas_old <- gammas
  gamma0_old <- gamma0

  X_gammas <- X_std %*% gammas
  Z <- (1/n)*sum(exp(-X_gammas)*log(Y))

  change <- 1/(exp(-gamma0)*Z) #Ratio we want to clip
  change.1 <- max(change,clipping1) #Clipping the ratio
  beta0 <- 1 + gamma0 -change.1 #Calculating beta0 with the clipped ratio
  gamma0 <- beta0

  for (j in 1:p){
    V_1 <- (1/n)*sum(X_std[,j]*exp(-X_gammas)*log(Y))
    V_2 <- (1/n)*sum((X_std[,j]^2)*exp(-X_gammas)*log(Y))
    beta_j <- soft_thresholding(gammas[j]*exp(-gamma0)*V_2 - mean(X_std[,j]) +
     exp(-gamma0)*V_1, lambda)/(exp(-gamma0)*V_2)
    gammas[j] <- beta_j
  }
  #Storing the values for betas
  for (l in 1:length(which.to.plot)) {
    if (which.to.plot[l] == 0) {
    betas_to_plot[iteration + 1,l] <- gamma0
    }
    else {
      betas_to_plot[iteration +1,l] <- gammas[which.to.plot[l]]
    }
  }
  # Check for convergence
  if (sqrt(sum((gammas-gammas_old)^2)+(gamma0-gamma0_old)^2) < delta) {
    #cat("Converged at iteration:", iteration + 1, "\n")
    break
  }
  iteration <- iteration +1
}
gammas <- gammas / X_sds
gamma0 <- gamma0 - sum(gammas*X_means)

#If we want to plot each individual beta we can uncomment this
#par(mfrow=c(1, length(which.to.plot)))
#for (k in 1:length(which.to.plot)) {
# plot(1:nrow(betas_to_plot), betas_to_plot[, k], type="l",
#main=paste("Beta", which.to.plot[k]), xlab="Iteration", ylab="Value",
#       xlim=c(0,iteration+1))
```

```
  #}

  return(list(beta0=gamma0, betas=gammas))
}
#Initializing the parameters for the coordinate descent
gamma0 <- 0 #Starting value for beta0
gammas <- rep(0,p) #Starting values for betas
delta <- 1e-6
lambda <- 0.1
nmax <- 1000 #Number of iterations
clipping1 <- 0.1 #Clipping value
coordinate_descent_p(X, Y, lambda, gamma0, gammas, delta, nmax, c(0,1,2), clipping1)
print(c(true.beta0,true.beta1,true.beta2,true.beta3,true.beta4,true.beta5,
true.beta6,true.beta7,true.beta8,true.beta9,true.beta10))

#Plotting the lambda path
lambdas <- seq(0.01, 1, by = 0.01) #Setting lambdas
p <- 10  # Number of predictors
#Storing beta estimates for each lambda
beta_paths <- matrix(NA, nrow = length(lambdas), ncol = p)
for (i in seq_along(lambdas)) {#Looping to find the beta estimates for each lambda
  result <- coordinate_descent_p(X, Y, lambdas[i], gamma0, gammas, delta,
  nmax, 1:p, clipping1)
  beta_paths[i, ] <- result$betas  # Store all beta estimates at once
}
#Plot all the lambda path
matplot(lambdas, beta_paths, type = "l", lty = 1, col = rainbow(p), lwd = 2,
        xlab = expression(lambda), ylab = expression(hat(beta)),
        main = "Lambda Paths for All Betas")
legend("topright", legend = paste0("Beta", 1:p), col = rainbow(p), lty = 1, lwd = 2)
```

7.4 Codes for Chapter 5

7.4.1 Section 5.1

```
#Loading required packages
library(readxl)

#Basic statistics for data
summary(NFL_data)

#Performing a linear regression
linear <- lm(NFL_data$`Points scored` ~ NFL_data$`Total yards` +
NFL_data$`Total offensive plays` + NFL_data$`Total turnovers`+
```

```
NFL_data$`Total first downs`+ NFL_data$`Total penalty yards`)

#Displaying the estimates and their p-values
summary(linear)
```

7.4.2 Section 5.2

```
#Loading required packages
library(glmnet)

#Grouping the predictors into one matrix
predictors <- cbind(NFL_data$`Total yards`,NFL_data$`Total offensive plays`,
              NFL_data$`Total turnovers`, NFL_data$`Total first downs`,
              NFL_data$`Total penalty yards`)

#Performing the linear regression with LASSO
linear_lasso <- glmnet(predictors,NFL_data$`Points scored`)
coef(linear_lasso)

#Finding the optimal lambda with cross-validation
linear_lasso.cv <- cv.glmnet(predictors, NFL_data$`Points scored`)
linear_lasso.cv

#The resulting coefficients
coef(linear_lasso.cv)
```

7.4.3 Section 5.3

```
#Loading required packages
library(glmnet)

#Quantile regression
mod <- rq(NFL_data$`Points scored` ~ NFL_data$`Total yards`
+ NFL_data$`Total offensive plays` + NFL_data$`Total turnovers`
+ NFL_data$`Total first downs`+ NFL_data$`Total penalty yards`,
tau=c(0.1,0.2,0.3,0.4,0.5,0.6,0.7,0.8,0.9))

#Displaying the estimates and their confidence intervals
summary(mod)
```

7.4.4 Section 5.4

```
#Loading required packages
library(rqPen)
```

```
#Performing the quantile regression with LASSO
lasso_quantile <- rq.pen(as.matrix(predictors), as.matrix(NFL_data$‘Points scored‘),
tau = 0.9, penalty ="LASSO")
coef(lasso_quantile) #All the predictors for each lambda value

#Finding the optimal lambda with cross-validation
lasso_quantile_cv <- rq.pen.cv(as.matrix(predictors),
as.matrix(NFL_data$‘Points scored‘), tau = 0.9, penalty ="LASSO")

#The resulting coefficients
coef(lasso_quantile_cv)
```

7.4.5 Section 5.5

```
#Plotting the total points by total yards
breaks <- seq(4479, 7500, by = 400)
labels <- paste0("[",format(breaks[-length(breaks)], scientific = FALSE),", ",
                 format(breaks[-1], scientific = FALSE),")")
NFL_data$YardBins <- cut(NFL_data$‘Total yards‘,
breaks = breaks, right = FALSE, labels = labels)
boxplot(‘Points scored‘ ~ YardBins, data = NFL_data,
        main = "Boxplot of total points by total yards",
        xlab = "Total yards", ylab = "Total points")
#Finding the quantiles
quantiles <- c()
quantile_values <- c(0.25, 0.5, 0.75)
quantiles$bin1 <- quantile(NFL_data$‘Points scored‘[NFL_data$YardBins[]
== "[4479, 4879)"], probs=quantile_values)
quantiles$bin2 <- quantile(NFL_data$‘Points scored‘[NFL_data$YardBins[]
== "[4879, 5279)"], probs=quantile_values)
quantiles$bin3 <- quantile(NFL_data$‘Points scored‘[NFL_data$YardBins[]
== "[5279, 5679)"], probs=quantile_values)
quantiles$bin4 <- quantile(NFL_data$‘Points scored‘[NFL_data$YardBins[]
== "[5679, 6079)"], probs=quantile_values)
quantiles$bin5 <- quantile(NFL_data$‘Points scored‘[NFL_data$YardBins[]
== "[6079, 6479)"], probs=quantile_values)
quantiles$bin6 <- quantile(NFL_data$‘Points scored‘[NFL_data$YardBins[]
== "[6479, 6879)"], probs=quantile_values)
quantiles$bin7 <- quantile(NFL_data$‘Points scored‘[NFL_data$YardBins[]
== "[6879, 7279)"], probs=quantile_values)
quantiles <- as.data.frame(quantiles)
quantiles <- t(quantiles)
quantiles <- as.data.frame(quantiles)
#Plotting total points by yards with the quantiles
boxplot(‘Points scored‘ ~ YardBins, data = NFL_data,
```

```
        main = "Boxplot of total points by total yards",
        xlab = "Total yards", ylab = "Total points")
lines(1:7, quantiles$`25%`, col = "red", lty = "dashed")
lines(1:7, quantiles$`50%`, col = "blue", lty = "dashed")
lines(1:7, quantiles$`75%`, col = "green", lty = "dashed")
```

Bibliography

[1] Team offense. *ProFootball Reference*, 2000.

[2] Formulating quantile regression as linear programming problem. *Stack Exchange*, December 2018.

[3] Jan Beirlant and Yuri Goegebeur. Regression with response distributions of pareto-type. *Computational Statistics & Data Analysis*, 42(4):595–619, 2003.

[4] Victor Chernozhukov. Extremal quantile regression. *The Annals of Statistics*, 33(2):806–839, 2005.

[5] Roger Koenker. *Quantile Regression*. Cambridge University Press, 2005.

[6] Rafal Kulik and Philippe Soulier. Heavy-tailed time series. *Springer Series in Operations Research and Financial Engineering*, Jul 2020.

[7] Youjuan Li and Ji Zhu. L1-norm quantile regression. *Journal of Computational and Graphical Statistics*, 17(1):163–185, 2008.

[8] Abdallah Mkhadri, Mohamed Ouhourane, and Karim Oualkacha. A coordinate descent algorithm for computing penalized smooth quantile regression. *Statistics and Computing*, 27(4):865–883, 2017.

[9] Bo Peng and Lan Wang. An iterative coordinate descent algorithm for high-dimensional nonconvex penalized quantile regression. *Journal of Computational and Graphical Statistics*, 24(3):676–694, 2015.

[10] Huixia Judy Wang and Deyuan Li. Estimation of extreme conditional quantiles. in: Extreme value modeling and risk analysis: Methods and applications. chapter 15, pages 319–337. 1st edition, Jan 2016.

Index

Breiman's lemma, 39

Central limit theorem, 10
Classical regression with LASSO, 22
Classical regression with ridge, 18, 19
CLT log averages, 44
CLT sample quantiles, 12
CLT tail empirical distribution, 42
CLT tail index, 45
Conditional Pareto distribution, 46
Conditional quantile, 7
Conditional quantile function, 7
Constrained optimization problem, 14
Coordinate descent for extreme quantile regression with LASSO, 52
Coordinate descent for extreme quantile regression with ridge, 50
Coordinate descent for ridge linear regression, 21
Coordinate descent LASSO linear regression, 26

Delta method, 11
Direct method for ridge, 19

Extreme conditional quantile, 57
Extreme conditional quantile using common slope, 60

Hill estimator, 41

Karamata theorem, 39

Lagrange formulation, 14
Linear program LASSO, 33
Linear program quantile regression, 28
Linear program ridge, 32
Loss function, 5

Minimization problem, 18
Minimization problem extreme conditional quantile β_0, 47
Minimization problem extreme conditional quantile general p, 48

Optimal solution Lagrange, 14

Pareto quantile function, 38

Quadratic approximation β_0, 47
Quadratic approximation β_1, 47
Quantile function, 5
Quantile regression with LASSO, 33
Quantile regression with ridge, 30

Regression problem, 17
Regular varying random variable, 38

Sample quantile function, 6
Smoothed quantile regression, 35
Subgradient, 16
Subgradient LASSO linear regression, 24

Tail empirical distribution, 42

Unconditional quantile estimator, 41