Robustness against Conflicting Prior Information in Linear Regression and Application to Lasso

Guangyi Chen

St Cross College

University of Oxford

A dissertation submitted in partial fulfilment of the requirements for the degree of

Master of Science in Statistical Science

Trinity 2019

Abstract

Traditionally in linear regression, the conflict of information is mainly considered as coming from the outliers. In this dissertation, we explain and show that prior information is also a source to generate conflicts. The prior is set in a location-scale framework, where two cases of conflicting prior information regarding the location and scale parameters are discussed. Through drawing the inspiration from the work of robustness agsints outliers (Gagnon et al. (2018)), the same idea can be applied: replace the prior distribution by a super heavy-tailed distribution, which is the log-regularly varying distribution (LRVD) in this work. Based on it, two resolutions for the two cases are presented respectively. The conflicting prior information also exists in lasso regression and we propose the prior in lasso to be distributed as a LPTL distribution that belongs to the family of LRVD to construct a wholly robust lasso against conflicting prior information. By comparing with other widely-used regression models, we investigate the performance of this wholly robust model.

Keywords: *conflict of information, robustness, lasso, parameters, prior distribution, super heavy-tailed distribution*

Acknowledgements

I would like to express my sincere gratitude to Dr. Philippe Gagnon, my dissertation supervisor, for his time, patience and insightful guidance in this summer. He taught me how to study a problem from different angles and how to dig into something interesting, and dig further. Honestly, I indeed learn a lot from him and it's so lucky for me to be supervised by such a good mentor.

Contents

List of Figures

Chapter 1

Introduction

1.1 Conflict of Information

In the Bayesian context, as mentioned by Andrade and O'Hagan (2011), *conflict of information* is seen as one or more sources of information about the parameters which contradict the rest of the information. Traditionally in linear regression, the main source has been identified to be outliers, and several methods have been proposed to address the problem (as will be noticed in the next section). In this work, we explain and show that prior information is also a source to generate conflicts, and propose a solution regarding this important problem.

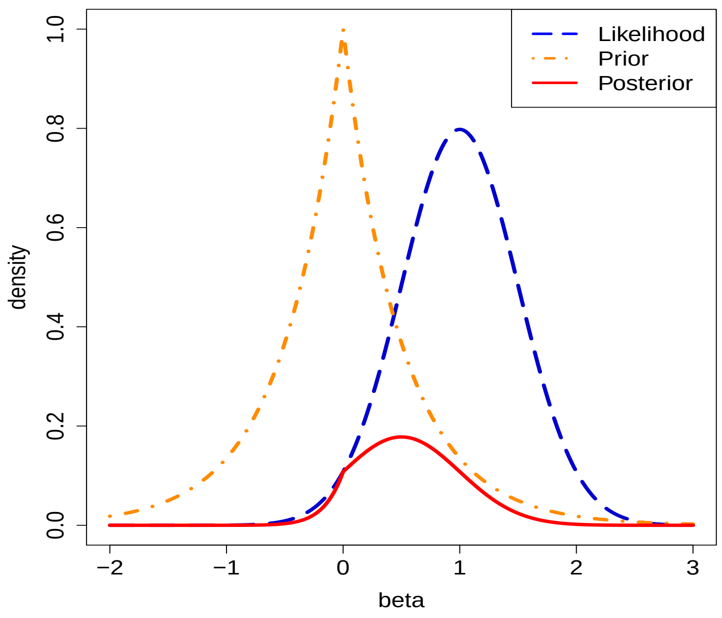

Figure 1.1: Plot of prior density: Laplace (0, 0.5), likelihood: $\mathcal{N}(\widehat{\beta}^{OLS}, \frac{1}{4})$, and the numerator of posterior density with respect to β.

To visualise the problem, we use an example with three functions : 1.the Laplace prior for the slope β centered at 0 and the scale is s; 2. the likelihood, which is a normal with mean

$\widehat{\beta}^{OLS}$ and variance $\frac{1}{n}$. Here, we fix $s = 0.5$ and $n = 4$ for visualization. Estimating the parameter is equivalent to maximizing the numerator of the posterior density (as the 3rd function) that is a product of the prior and the likelihood, i.e.

$$\pi(\beta) \propto \frac{1}{s} \exp\left(-\frac{|\beta|}{s}\right) \frac{\sqrt{n}}{\sqrt{2\pi}} \exp\left(\frac{(y-\beta)^2}{2/n}\right) \tag{1.1}$$

The plot of the three functions with respect to β are presented in Figure 1.1. Strictly, we should use the posterior density whose integration is 1 but doing so the red line in Figure 1.1 will be unclear so we use the numerator of it, where the marginal is a constant. The plot clearly shows that the prior information of the parameter has an impact on the likelihood that is supported by the data so that the posterior density of β is in between. Indeed, we can see that prior information is a source to generate conflicts.

1.2 Robustness against Outliers

A lot of work on the robustness against outliers in linear regression has been done in both frequentist and Bayesian statistics. In frequentist statistics, the concept of breakdown-point was firstly introduced by Hampel (1971) as a measure of the degree of the robustness in the presence of outliers. Huber et al. (1973) proposed the M-estimates that was obtained by minimizing $\sum_i^n \rho(r_i)$ rather than $\sum_i^n r_i^2$, where r_i is the residual between the observed value $\mathbf{y}$ and the estimated value $\mathbf{x}^T\widehat{\boldsymbol{\beta}}$. However, the breakdown-point of this estimate is 0.

High breakdown-point (0.5) was achieved by several other estimates later. Least median of squares (LMS, Rousseeuw (1984)) and Least trimmed squares (LTS, Rousseeuw (1985)) are the two of them, which minimizes the median and the trimmed mean of the squared residuals respectively. S-estimators (Rousseeuw and Yohai (1984)) are a generalization of LMS and LTS. It is defiend by a minimization of the residual scale and share good properties of M-estimates such as asymptotic and flexibility. MM-estimates was proposed by Yohai et al. (1987) with the purpose of having both high breakdown-point and high efficiency under normal errors. Gervini et al. (2002) achieved full efficiency under normal errors and maximum breakdown-point by put forwarding the robust and efficient weighted least-square estimator (REWLSE).

In Bayesian statistics, we cannot directly propose estimators. Instead, we have to work directly on modelling, which is what the other authors did. Conflict resolution was first described by De Finetti (1961), and then Box and Tiao (1968) gave a proposition that changed the distribution of the error term to a mixture of two normal, that is one component for the nonoutliers while the other with a larger variance for the outliers. West (1984) generalised this approach to make errors be distributed heavy-tailed as a scale mixture of normals and also utilized the Student distribution. Besides, he also gave the pricise definition of the conflict resolution agaisnt outliers, that is "an outlying observation is accommodated if the posterior distribution converges to that excluding the outlier as it tends to infinity". To deal with this issue, Peña et al. (2009) proposed something different (a heteroscedastic approach), but the result is also not so strong.

A new method presenting interesting results recently emerged. Drawing inspiration from Desgagné et al. (2015) with his context and work, Gagnon et al. (2018) proposed a model with super heavy-tailed distribution on the error term and gave the result that with such model, the posterior inference arising from the whole sample converges towards the posterior inference based on the nonoutliers only. This work also addressed the problem of partial robustness of the approach introduced by Andrade and O'Hagan (2011) who proposed a distribution on error term with heavier tails than the Student distribution, where partial robustness refers to a significant impact on the estimation of the parameters. The results of Gagnon et al. (2018) are referred to as wholly robustness, which is consistent with the definition of conflict resolution given by West (1984) above. Besides, the results hold for any types of covariates (and thus valid in ANOVA and ANCOVA), and also in model selection when the joint posterior of the models and their parameters are considered. The super heavy-tailed distribution of the error term that Gagnon et al. (2018) set is the log-Pareto-tailed normal (LPTN), which is introduced by Desgagné et al. (2015). The central part with mass ρ of the density of LPTN is standard normal so that it makes the error term still be normal in the nonoutlier case while the heavy tail essentially discards the information carried by the outliers. However, so far no solution was proposed for obtaining whole robustness with respect to conflicting prior information.

1.3 Robustness against Conflicting Prior Information

In Bayesian linear regression, we need to set the prior on the parameters $\boldsymbol{\beta}$ and σ. In previous work, Bayesian statisticians proposed to set the prior as a mixture of $h + 1$ conditional distributions in the variable-selection problem of regression. As introduced by Garthwaite et al. (1992), carrying out variable selection implies to give positive probability that some coefficients β are 0 in the elicitation process of prior. They let H_i be the hypothesis that some coefficients are 0 and the others are not, for each $i = 1, ..., h$. H_0 represents that all β are not 0. Thus there are $h+1$ hypothesis $H_0, ...H_h$ and the author assumes that exactly one of them has positive probability being true. The prior distribution can be written as:

$$f(\boldsymbol{\beta}, \sigma) = \sum_{i=0}^{h} f(\boldsymbol{\beta}_{(i)}, \sigma | H_i) P(H_i) \tag{1.2}$$

where $P(H_i)$ is the elicitation probability that H_i is true and $\boldsymbol{\beta}_{(i)}$ denotes the nonzero coefficients if H_i is true. Prior represented as this way was also introduced by Dickey (1975), Hill (1974), etc. Then, straightforwardly, $f(\boldsymbol{\beta}_{(i)}, \sigma|H_i)$ can be written as

$$f(\boldsymbol{\beta}_{(i)}, \sigma|H_i) = f(\boldsymbol{\beta}_{(i)}|H_i, \sigma) f(\sigma|H_i) \tag{1.3}$$

Continuing with above, Garthwaite et al. (1992) let each distribution in the RHS of (1.2) belongs to the natural conjugate family, that is

$$\boldsymbol{\beta}|H_0, \sigma \sim N(\boldsymbol{\mu}, \frac{\sigma^2 \boldsymbol{U}}{\omega}) \quad \sigma^2 \sim v\omega/\chi^2_v$$

where v, ω, μ and $\boldsymbol{U}$ are the hyper-parameters to be chosen in the elicitation just like the probability $P(H_i)$. Similarly, Raftery et al. (1997) used the standard gamma conjugate class of priors to study the Bayesian linear regression model averaging, i.e. set $\boldsymbol{\beta} \sim N(\boldsymbol{\mu}, \sigma^2 \boldsymbol{V})$ and $\frac{v\lambda}{\sigma^2} \sim \chi^2_v$, where v, λ, matrix $\boldsymbol{V}$ and vector $\boldsymbol{\mu}$ are the hyper-parameters to be determined. And they center the distribution on $\boldsymbol{\beta}$ (except β_0) on zero by setting all entries of $\boldsymbol{\mu}$ (except the first entry) to 0.

From above, we can see that people usually set a vector $\boldsymbol{\mu}$ to represent the center of the prior, which could be a problem when the prior has light tails such as normal described above. Because for some β the prior will be far from the true value supported by the likelihood, then the prior will act as the conflicting information in linear regression, which does the same impact as outliers. In particular, sometimes $\boldsymbol{\mu}$ is set to 0, which is the key idea of the regularization and is the Lasso framework that we will discuss later.

Therefore, when people set the prior as above, conflicting prior information indeed exist and is an issue. There is a few previous work regarding this problem. For instance, two approaches for Bayesian linear regression modeling based on conjugate priors (the standard conjugate prior (SCP) and "canonically constructed conjugate prior" (CCCP)) were considered by Walter and Augustin (2010) under prior-data conflict. In detail, SCP on the parameters $(\boldsymbol{\beta}, \sigma^2)$ is a normal – inverse gamma (NIG) distribution while CCCP model is a special case of SCP, which adds a fixed variance-covariance structure for the prior of $\boldsymbol{\beta}|\boldsymbol{\sigma^2}$. The main difference comparing to our model is that their prior models are unable to reflect the appropriateness of the prior assignments for each variable separately.

In this thesis, we continue this topic in linear regression. By drawing the inspiration from Gagnon et al. (2018) with his work of robustness against outliers, we construct a wholly robust model for prior. Essentially, a robust model gives resolution of conflicts by discarding the information coming from the minority, which is the prior in this case. Therefore, the same idea of dealing with the conflicting outliers information can be applied on prior, i.e. replacing the prior distribution by a super heavy-tailed distribution.

1.4 Overview of Objectives and Structure

In this section, we present our objectives and explain how the thesis is organised around these. Our objectives are the following:

- Study the robustness against conflicting prior information in linear regression in a general framework.
- Make the connection with the popular lasso regression and present its robust version.
- Empirically evaluate the performance of the wholly robust model on a real data set by comparing with other widely-used models.

In Chapter 2, we present the general framework of Bayesian linear regression. Next, we describe LRVD distributions and their property that serve as foundation for our approach. To study the conflicting prior information, we give two cases of it by setting the location and scale parameter of the coefficient in linear regression, respectively. Theoretical results (**Resolution 1**, **Resolution 2**) are also presented to strongly support our claim that our approach is relevant and valid.

In Chapter 3, we introduce lasso regression and its Bayesian interpretation. Next, Log-Pareto-tailed Laplace (LPTL), a super heavy-tailed distribution, is proposed as the distribution of the prior to replace the Laplace distribution in lasso. The central part with mass ρ of the density of LPTL is standard Laplace, where ρ is chosen by the user. Thus, LPTL prior makes the robust model remain the good property of variable selection from Lasso.

Based on the LPTL prior, the resolution of conflicts for lasso is presented, which is a special case of **Resolution 2**. To compare the lasso and LPTL model, we mathematically infer the relationship between $\widehat{\boldsymbol{\beta}}^{OLS}$, $\widehat{\boldsymbol{\beta}}^{LASSO}$ and $\widehat{\boldsymbol{\beta}}^{LPTL}$ in the situation where the covariates are orthogonal. Through implementing the models on a simulated data set, we clearly shows the difference between lasso and LPTL model.

Chapter 4 is our main application part, where we utilise the latest players' salary and performance data of 2018/19 NBA season. In this work, we compare our models with 4 widely-used approaches: best subset selection, lasso, ridge and elastic net on this dataset. Also, we apply LPTN to our approach as a tool to evaluate the presence of outliers in our dataset. To speed up the whole procedure, we implement parallel computing and adopt the `optimx` function with method BFGS in R in our algorithm, where BFGS is a kind of optimiser that performs iterative algorithm to solve nonlinear optimization problems. The results show that our approach is favorable comparing to other methods. A thorough analysis indicating why is carried out.

The proof of **Resolution 1** is presented in Chapter 6, where we list two assumptions that are hard to prove rigoriously. So we present the idea of the proof of the asummptions by drawing inspiration from the work of Gagnon et al. (2018). The proof of **Resolution 2** is similar to **Resolution 1** and we briefly discuss it in Section 2.4. Actually, proof is not in the objectives of this thesis, but I still try it and write down the ideas to show my work.

Chapter 2

Resolution of Conflicting Prior Information

2.1 Linear Regression Model

(i) Let $Y_1, Y_2, \ldots, Y_n$ be n random variables, which represents data points from the dependent variable; $\mathbf{x}_1^T := (1, x_{12}, \ldots, x_{1p}), \ldots, \mathbf{x}_n^T := (1, x_{n2}, \ldots, x_{np})$ be n vectors of data points from the covariates, where $n \geq p+1$, $p \in 2, 3, \ldots$, and $x_{ij} \in \mathbb{R}$ are assumed to be known constants. The linear regression model is as follows:

$$Y_i = \mathbf{x}_i^T \beta + \epsilon_i, \quad i = 1, \ldots, n \tag{2.1}$$

where the random variables $\epsilon_i, \ldots, \epsilon_n$ are errors and the p-dimensional random variable $\boldsymbol{\beta} := (\beta_1, \ldots, \beta_p)^T \in \mathbb{R}^p$ represents the coefficients in the linear regression.

(ii) We now give all the assumptions leading to the likelihood. As mentioned in (i), there are n + 1 random variables in total, which are conditionally independent given the scale parameter σ. Then we can set the conditional density of ϵ_i:

$$\epsilon_i | \boldsymbol{\beta}, \sigma \overset{indep.}{=\!=\!=} \epsilon_i | \sigma \overset{d}{\sim} (1/\sigma) f(\epsilon_i/\sigma), \quad i = 1, ..., n$$

As $\epsilon_i = y_i - \mathbf{x}_i^T \beta$, the likelihood is:

$$L(\boldsymbol{\beta}, \sigma; \mathbf{y}) = \prod_{i=1}^{n} (1/\sigma) f((y_i - \mathbf{x}_i^T \boldsymbol{\beta})/\sigma) \tag{2.2}$$

(iii) f is assumed to be a strictly positive continuous probability density function on $\mathbb{R}$, which is symmetric with respect to the origin. All parameters of it are known and there exists a threshold above which $g(z) = zf(z)$ is monotonic. For instance, f can be normal, Laplace, Student (fix degree of freedom) and LPTN/LPTL (Chapter 3).

(iv) The prior could be of any form, written as $\pi(\beta \mid \sigma)\pi(\sigma)$. We restrict our attention in this thesis to the cases where, a priori, $\beta_1, \ldots, \beta_p$ are conditionally independent given σ, with each their own location-scale parameters. Thus the conditional density for β_j is given by:

$$\beta_j | \sigma \overset{d}{\sim} (1/s_j)\, g((\beta_j - \mu_j)/s_j), \quad j = 1, ..., p$$

where $\mu_j \in \mathbb{R}$ and $s_j(\sigma)$ is a positive function of σ representing the location and scale parameter for β_j, respectively. We assume g is a PDF fulfilling the same conditions as f. Thus the form of our prior in the linear regression model is given by:

$$\pi(\boldsymbol{\beta}, \sigma) = \pi(\boldsymbol{\beta}|\sigma)\pi(\sigma) = \pi(\sigma)\prod_{j=1}^{p}\pi(\beta_j|\sigma) = \pi(\sigma)\prod_{j=1}^{p}(1/s_j)\, g((\beta_j - \mu_j)/s_j) \tag{2.3}$$

According to equation (2.2) and (2.3), we write the posterior density as:

$$\pi(\boldsymbol{\beta}, \sigma|\mathbf{y}) = \frac{L(\boldsymbol{\beta}, \sigma; \mathbf{y})\pi(\boldsymbol{\beta}, \sigma)}{m(\mathbf{y})} = [m(\mathbf{y})]^{-1}\pi(\sigma)\prod_{j=1}^{p}\frac{1}{s_j}\, g\Big(\frac{\beta_j - \mu_j}{s_j}\Big)\prod_{i=1}^{n}\frac{1}{\sigma}f\Big(\frac{y_i - \mathbf{x}_i^T\boldsymbol{\beta}}{\sigma}\Big) \tag{2.4}$$

where $m(\mathbf{y})$ is the marginal density of $\mathbf{y}$, i.e.

$$m(\mathbf{y}) = \int_{\mathbb{R}^p}\int_0^\infty L(\boldsymbol{\beta}, \sigma; \mathbf{y})\pi(\boldsymbol{\beta}, \sigma)d\sigma d\boldsymbol{\beta} \tag{2.5}$$

2.2 Log-Regularly Varying Distributions

As mentioned in the introduction, we know that the behavior of the tails of the distributions forming the linear regression is a key point for robustness. In this section, we present LRVD constructed from log-regularly-varying function (Desgagné et al. (2013) and Desgagné et al. (2015)) and their properties. The idea is from regularly varying functions, which was developed by Karamata (1930). This section is similar to Section 2.2 in Gagnon et al. (2018); we include that material to make the thesis self-contained.

Definition 2.1
We say that a measurable function g is log-regularly varying at ∞ with index $\theta \in \mathbb{R}$ written $g \in L_\theta(\infty)$, if

$$\lim_{z\to\infty}\frac{g(z^v)}{g(z)} = v^{-\theta}$$

uniformly in any set $v \in [\frac{1}{\eta}, \eta]$, for any $\eta \geq 1$. If $\theta = 0$, then g is said to be log-slowly varying at ∞

Desgagné et al. (2015) proves a equivalence of Definition 2.1: If there exists a constant $A > 1$ and a function $s \in L_0(\infty)$ such that for $z \geq A$, g can be written as

$$g(z) = (\log z)^{-\theta}s(z)$$

Two examples of log-regularly varying functions are: $g(z) = (\log z)^{-\theta}$ (set $s(z) = 1$) and $g(z) = (\log z)^{-\theta}\log(\log z)$. To show the latter is LRVF, we need to prove $s(z) = \log(\log z) \in L_0(\infty)$ by above. Let $v \in [\frac{1}{\eta}, \eta]$ for any $\eta \geq 1$ and by $L'H\hat{o}pital$'s Rule, we have

$$\lim_{z\to\infty}\frac{\log(\log z^v)}{\log(\log z)} \overset{L'H\hat{o}pital}{=\!=\!=} \lim_{z\to\infty}\frac{vz^v\log z}{z^v\log z^v} \overset{L'H\hat{o}pital}{=\!=\!=} \lim_{z\to\infty}\frac{vz^v}{vz^v} = 1 = v^{-0}$$

By Definition 2.1, $\log(\log z) \in L_0(\infty)$, thus $g(z) = (\log z)^{-\theta}\log(\log z)$ is a LRVF.

Definition 2.2 (LRVD)
A random variable Z and its distribution are said to be log-regularly varying with index $\theta \geq 1$ if their density f is such that $zf(z) \in L_\theta(\infty)$.

From Definition 2.2, we can see that any density f with tails behaving like $|z|^{-1}(\log|z|)^{-\theta}$ with $\theta > 1$ is a LRVD. The reason is that for such density, $|z|f(|z|) = \log(|z|) \in L_\theta(\infty)$. Thus by Definition 2.2, the distribution of Z is LRVD. Some examples of LRVD are the LPTN distribution presented in Desgagné et al. (2015) and LPTL introduced in Chapter 3. In addition, LRVD has a quite important property following from Definition 2.1, which is the asymptotic Location-scale invariance of their density (Proposition 2.1 below).

Proposition 2.1 (Location-scale invariance) *If $zf(z) \in L_\theta(\infty)$, then we have*

$$\frac{\frac{1}{\sigma}f(\frac{z-\mu}{\sigma})}{f(z)} \to 1 \quad as\ z \to \infty \tag{2.6}$$

uniformly on $(\mu, \sigma) \in [-\vartheta, \vartheta] \times [\frac{1}{\eta}, \eta]$, for any $\vartheta \geq 0$ and $\eta \geq 1$.

The proof is given by Desgagné et al. (2015).

Proposition 2.1 is essential for the resolution of conflicts in this work. Here, we briefly show the idea about how Proposition 2.1 leads to the resolution of conflicts in a general and interesting setting. The objective is to estimate the parameters using the maximum a posteriori (MAP) estimate and that $f := \mathcal{N}(0,1)$, g is a LRVD, then we maximise the numerator in the equation (2.4). We assume that all priors are well specified, in the sense that the data support the prior information and $s_j := \sigma/c_j$ with c_j is not too large, except for β_{j_0} for which there is a significant difference with $\mu_{j_0} := 0$ in the region where the likelihood puts mass, relatively to $s_{j_0}(\sigma) := \sigma/c_0$ with c large. From a resolution of conflict perspective, the aim is to find the same MAP as if that prior density of $\beta_{j_0}$was excluded and that is essentially what we obtain:

$$\begin{aligned}
\max_{\boldsymbol{\beta},\sigma} \pi(\boldsymbol{\beta}, \sigma|\mathbf{y}) &\propto \max_{\boldsymbol{\beta},\sigma} \pi(\sigma) \prod_{j=1}^{p} \frac{1}{s_j} g\Big(\frac{\beta_j - \mu_j}{s_j}\Big) \prod_{i=1}^{n} \frac{1}{\sigma} f\Big(\frac{y_i - \mathbf{x}_i^T\boldsymbol{\beta}}{\sigma}\Big) \\
&\overset{\text{I}}{\approx} \max_{\boldsymbol{\beta},\sigma} \pi(\sigma) \frac{1}{|\beta_{j_0}|} c_0 g(c_0) \prod_{\substack{j=1 \\ j\neq j_0}}^{p} \frac{1}{s_j} g\Big(\frac{\beta_j - \mu_j}{s_j}\Big) \prod_{i=1}^{n} \frac{1}{\sigma} f\Big(\frac{y_i - \mathbf{x}_i^T\boldsymbol{\beta}}{\sigma}\Big) \\
&\propto \max_{\boldsymbol{\beta},\sigma} \pi(\sigma) \frac{1}{|\beta_{j_0}|} \prod_{\substack{j=1 \\ j\neq j_0}}^{p} \frac{1}{s_j} g\Big(\frac{\beta_j - \mu_j}{s_j}\Big) \prod_{i=1}^{n} \frac{1}{\sigma} f\Big(\frac{y_i - \mathbf{x}_i^T\boldsymbol{\beta}}{\sigma}\Big)
\end{aligned}$$

where in step I, for $j = j_0$:

$$\begin{aligned}
\frac{1}{s_{j_0}} g\Big(\frac{\beta_{j_0} - \mu_{j_0}}{s_{j_0}}\Big) &= \frac{c_0}{\sigma} g\Big(\frac{|\beta_{j_0}|}{\frac{\sigma}{c_0}}\Big) \quad \text{since } g \text{ is symmetric} \\
&= \frac{c_0}{|\beta_{j_0}|} \cdot \frac{1}{\frac{\sigma}{|\beta_{j_0}|}} g\Big(\frac{c_0}{\frac{\sigma}{|\beta_{j_0}|}}\Big) \\
&\approx \frac{1}{|\beta_{j_0}|} c_0 g(c_0) \quad \text{By Proposition 2.1 and since } c_0 \text{ is large}
\end{aligned}$$

We see that the penalisation on β_{j_0}, i.e. $\frac{1}{|\beta_{j_0}|}$ is much less strong than $\exp(-c_0|\beta_{j_0}|)$, which is basically what we have when the prior is a Laplace. Then the prior may have virtually no impact when the least-square estimate is between 0.5 and 1, which is what we observed in our analysis.

2.3 Two Cases of Conflicting Prior Information

We mainly focus on the robustness of the estimation of σ and $\boldsymbol{\beta}$ in the presence of conflicting prior information. Firstly, we mathematically define what conflicting prior information means. Two cases are considered. We consider that a prior fits either in none of those two cases or exactly one of them.

2.3.1 Case 1

The first case corresponds to the situation where μ_j is significantly different from the values of β_j supported by the data, but s_j is not too small or too large. It can be viewed as analysing the difference $d_j := \beta_j - \mu_j$. Hence, from a theoretical perspective, we let the difference d_j approaches $+\infty$ or $-\infty$ when there exists conflicting prior information on β_j. And since β_j is the parameter we need to estimate, to mathematically represents this problem, our strategy is to let μ_j tends to infinity. Specifically, we set

$$\mu_j := a_j + b_j\omega \qquad j = 1, \ldots, p \tag{2.7}$$

where a_j, $b_j \in \mathbb{R}$. Let $\omega \to \infty$, then it is obvious that the conflicting prior information exists for β_j when $b_j \neq 0$ while the case $b_j = 0$ represents the opposite situation.

To represent the situation, two binary functions k_j and l_j are defined:

$$k_j = \begin{cases} 1 & \text{if no conflicting prior information on } \beta_j \\ 0 & \text{if there is conflicting prior information on } \beta_j \end{cases} \qquad l_j = 1 - k_j \quad \text{for } j = 1, \ldots, p$$

and $\sum_{j=1}^{p} k_j = k$, $\sum_{j=1}^{p} l_j = l$, then $l + k = p$ where k represents the number of β_j that has no conflicting prior information; l represents the number of having conflicting information.

Based on above, the prior without conflicting prior information can be defined as:

$$\pi^k(\boldsymbol{\beta}, \sigma) := \pi(\sigma) \prod_{j=1}^{p} \Big[\frac{1}{s_j}\, g\big(\frac{\beta_j - \mu_j}{s_j}\big)\Big]^{k_j} \tag{2.8}$$

Let $\pi^k(\boldsymbol{\beta}, \sigma|y_n)$ represents the joint posterior density of $\boldsymbol{\beta}$ and σ arising from the prior without conflicting information, And $m^k(\mathbf{y})$ is the corresponding marginal density. Following the equation (2.4), we have:

$$\pi^k(\boldsymbol{\beta}, \sigma|\mathbf{y}) = [m^k(\mathbf{y})]^{-1}\pi(\sigma) \prod_{j=1}^{p} \Big[\frac{1}{s_j}\, g\big(\frac{\beta_j - \mu_j}{s_j}\big)\Big]^{k_j} \prod_{i=1}^{n} \frac{1}{\sigma} f\Big(\frac{y_i - \mathbf{x}_i^T\boldsymbol{\beta}}{\sigma}\Big) \tag{2.9}$$

where $\boldsymbol{\beta} \in \mathbb{R}^p$, $\sigma > 0$, and

$$m^k(\mathbf{y}) = \int_{\mathbb{R}^p} \int_0^\infty L(\boldsymbol{\beta}, \sigma; \mathbf{y})\pi^k(\boldsymbol{\beta}, \sigma)\, d\sigma\, d\boldsymbol{\beta} \tag{2.10}$$

2.3.2 Case 2

For the second case, we set

$$s_j := t(\sigma)/(a_j + b_j\omega) \qquad j = 1, \dots, p \tag{2.11}$$

where t is a positive function of σ, and the difference d_j defined above is not too small or too large. Similarly, we let $\omega \to \infty$. If there exists conflicting prior information on β_j, then $b_j \neq 0$ and $s_j \to 0$, otherwise $b_j = 0$, then $s_j = \sigma/a_j$.

In this case, we define a new prior for presenting the **Resolution 2** in next Section:

$$\pi^d(\boldsymbol{\beta}, \sigma) := \pi(\sigma) \prod_{j=1}^{p} \Big[\frac{1}{s_j}\, g\big(\frac{\beta_j - \mu_j}{s_j}\big)\Big]^{k_j} \Big[\frac{1}{|\beta_j - \mu_j|}\Big]^{l_j} \tag{2.12}$$

The corresponding posterior density will be

$$\pi^d(\boldsymbol{\beta}, \sigma|\mathbf{y}) = [m^d(\mathbf{y})]^{-1}\pi(\sigma) \prod_{j=1}^{p} \Big[\frac{1}{s_j}\, g\big(\frac{\beta_j - \mu_j}{s_j}\big)\Big]^{k_j} \Big[\frac{1}{|\beta_j - \mu_j|}\Big]^{l_j} \prod_{i=1}^{n} \frac{1}{\sigma} f\Big(\frac{y_i - \mathbf{x}_i^T\boldsymbol{\beta}}{\sigma}\Big) \tag{2.13}$$

where $\boldsymbol{\beta} \in \mathbb{R}^p$, $\sigma > 0$, and

$$m^d(\mathbf{y}) = \int_{\mathbb{R}^p} \int_0^\infty L(\boldsymbol{\beta}, \sigma; \mathbf{y})\pi^d(\boldsymbol{\beta}, \sigma) d\sigma d\boldsymbol{\beta} \tag{2.14}$$

2.4 Resolution of conflicts

Two resolutions are presented below regarding the the two cases of conflicting prior information, respectively.

Resolution 1. (For Case 1)
If

(i) $zg(z) \in L_\theta(\infty)$ with $\theta > 1$, i.e. g is a LRVD

(ii) n > p + 1.

Then as $\omega \to \infty$, where ω is defined in equation (2.6), the results are as the follows:

(A)
$$\frac{m(\mathbf{y})}{\prod_{j=1}^{p} [g(\mu_j)]^{l_j}} \to m^k(\mathbf{y})$$

where $m^k(\mathbf{y})$ is defined in equation (2.10).

(B)
$$\pi(\boldsymbol{\beta}, \sigma|\mathbf{y}) \to \pi^k(\boldsymbol{\beta}, \sigma|\mathbf{y})$$

uniformly on $(\boldsymbol{\beta}, \sigma) \in [-\vartheta, \vartheta]^p \times [\frac{1}{\eta}, \eta]$, for any $\vartheta \geq 0$ and $\eta \geq 1$, where $\pi^k(\boldsymbol{\beta}, \sigma|\mathbf{y})$ is defined in equation (2.9).

Resolution 1 implies that if there is a conflict between the information carried by the data and the prior, the information carried by the prior will be discarded. Thus in this case the estimates $\widehat{\beta}_j$ will be equal to the OLS estimates $\widehat{\beta}_j^{OLS}$. If there is no conflict, the prior still works when we maximise the posterior density, which is exactly what we are aiming for.

Resolution 2. (For Case 2)
If

(i) $zg(z) \in L_\theta(\infty)$ with $\theta > 1$, i.e. g is a LRVD

(ii) n > p + 1.

Then as $\omega \to \infty$, where ω is defined in equation (2.6), the results are as the follows:

(C)

$$\frac{m(\mathbf{y})}{\prod_{j=1}^{p} [c_j\, g(c_j)]^{l_j}} \to m^d(\mathbf{y})$$

where $m^d(\mathbf{y})$ is defined in equation (2.14) and $c_j = a_j + b_j\omega$ is the denominator of s_j in Case 2.

(D)

$$\pi(\boldsymbol{\beta}, \sigma|\mathbf{y}) \to \pi^d(\boldsymbol{\beta}, \sigma|\mathbf{y})$$

uniformly on $(\boldsymbol{\beta}, \sigma) \in [-\vartheta, \vartheta]^p \times [\frac{1}{\eta}, \eta]$, for any $\vartheta \geq 0$ and $\eta \geq 1$, where $\pi^d(\boldsymbol{\beta}, \sigma|\mathbf{y})$ is defined in equation (2.13).

Resolution 2 implies that if there is a conflict between the information carried by the data and the prior, the prior carrying the information will be replaced by another prior that has much less conflict with the data, e.g. much less penalization on $\boldsymbol{\beta}$. Thus in this case the estimates $\widehat{\beta}_j$ will be close to the OLS estimates $\widehat{\beta}_j^{OLS}$. If there is no conflict, the prior still works when we maximise the posterior density, which is the same as the **Resolution 1**.

The proof of **Resolution 1** is given in Chapter 6, where we define two assumptions and also provide the ideas about how to prove the given assumptions. The proof of **Resolution 2** is similar with adding an additional term of $\left[\frac{1}{|\beta_j - \mu_j|}\right]^{l_j}$ in the prior, given as $\pi^d(\boldsymbol{\beta}, \sigma)$. Since the process of proving it is nearly the same as **Resolution 1** and we show how it comes from with a special case $\mu_j = 0$ in Section 3.3, we do not show the rigorous proof in this work. The hard part should also be the proof of the assumptions, similar as Resolution 1 and in this case, we need to proof that $\pi^d(\boldsymbol{\beta}, \sigma|\mathbf{y})$ is proper and also the interchangeability of the limit and the integral. With the additional term, we need to be more careful when β_j close to μ_j, which can be further worked on.

Chapter 3

Application to Lasso

3.1 Lasso Regression

3.1.1 Objective of Lasso

Lasso, i.e. Least Absolute Shrinkage and Selection Operator is proposed by Tibshirani (1996). It is a widely used regression analysis method with applications in many fields, which essentially minimizes the residual sum of squares subject to the sum of the absolute value of the coefficients being less than a constant. Lasso performs both variable selection like subset regression and regularization like Ridge Regression, and that it is desirable for enhancing the prediction accuracy and interpretability of the result model. The good property of variable section of lasso is due to the nature of L_1 penalty of its loss function comparing with the L_2 penalty of ridge that does not has this property.

Bacause of the good property and performance of lasso model, there are many generalisations of lasso. Tibshirani et al. (2005) proposed Fused Lasso with one more constraint than lasso, that is $\sum_{j=2}^{p} |\beta_j - \beta_{j-1}| \leq \eta$, where η is the tuning parameter. It is often used when the number of variables is greater than the number of obervations. Adaptive Lasso (Zou (2006)) adds a adaptive weight for each coefficient in the penalty term that is given by $\sum_{j=1}^{p} \omega_j |\beta_j|$ so that it can be used for penalizing different coefficients. Grouped Lasso (Yuan and Lin (2006)) with penalty term as $\sum_{j=1}^{p} \|\beta_j\|_{K_j}$, where K_j are the given positive definite matrices, is proposed for selecting grouped variables for prediction in linear regression.

To write the form of lasso mathematically, we consider $\mathbf{y} := (y_1, ..., y_n)^T \in \mathbb{R}^n$ to be a vector containing data points from a standardised dependent variables; $\mathbf{X} := (x_{ij}) \in \mathbb{R}^{n \times d}$ is a design matrix whose columns are data points from p standardised covariates. Our objective is to do prediction, i.e. using new $\widetilde{\mathbf{x}}_i := (\widetilde{x}_{i1}, ..., \widetilde{x}_{ip})^T$ to predict the values of new $\widetilde{y}_i$ by $\widetilde{\mathbf{x}}_i^T \widehat{\boldsymbol{\beta}}$, where $\widehat{\boldsymbol{\beta}} := (\widehat{\beta}_1, ..., \widehat{\beta}_p)^T \in \mathbb{R}^p$ is vector of the estimates of coefficients, which is obtained by solving the optimization problem of

$$\min_{\beta} \left\{ \frac{1}{n} \|\mathbf{y} - \mathbf{X}\boldsymbol{\beta}\|_2^2 \right\} \quad \text{subject to } \|\boldsymbol{\beta}\|_1 \leq t$$

where $\|\boldsymbol{\beta}\|_p := (\sum_{j=1}^{p} |\beta_j|^p)^{1/p}$ is the standard l^p-norm and t is a tuning parameter that determines the amount of regularisation.

More broadly, we use the Lagrangian form, given by:

$$\min_{\beta} \left\{ \frac{1}{n} \|\mathbf{y} - \mathbf{X}\boldsymbol{\beta}\|_2^2 + \lambda \|\boldsymbol{\beta}\|_1 \right\} \tag{3.1}$$

where $\lambda > 0$ is a hyper-parameter.

3.1.2 Bayesian Interpretation of Lasso

For the optimization problem in (3.1), the solutions can be viewed as a maximum a posteriori probability (MAP) estimate of $\boldsymbol{\beta}$ in the posterior density of the equation (2.4), where $f := \mathcal{N}(0,1)$ and g follows a Laplace distribution with $\mu_1 = \ldots = \mu_p = 0$, $s_1(\sigma) = \ldots = s_p(\sigma) = s := (2\sigma^2)/(\lambda n)$. Here, we set a non-informative prior on σ, $\pi(\sigma) \propto \frac{1}{\sigma}$. Indeed, we can perform a full Bayesian analysis on it since maximising the posterior with respect to $\boldsymbol{\beta}$ is equivalent to minimize the loss function in (3.1).

According to the equation (2.4), the posterior density can be written as:

$$\pi(\boldsymbol{\beta}, \sigma|\mathbf{y}) \propto \left(\prod_{j=1}^{p} \frac{1}{2s} \exp\left(-\frac{|\beta_j|}{s}\right)\right) \frac{1}{\sigma} \prod_{i=1}^{n} \frac{1}{\sqrt{2\pi}\sigma} \exp\left(-\frac{(y_i - \boldsymbol{x}_i^T\boldsymbol{\beta})^2}{2\sigma^2}\right) \tag{3.2}$$

where $\frac{1}{2s}\exp(-\frac{|\beta_j|}{s})$ is the PDF of Laplace with location 0 and scale s.

Essentially, we have

$$\max_{\boldsymbol{\beta}\in\mathbb{R}^p, \sigma>0} \pi(\boldsymbol{\beta}, \sigma|\mathbf{y}) \sim \min_{\beta}\{\frac{1}{n} \|\mathbf{y} - \mathbf{X}\boldsymbol{\beta}\|_2^2 + \lambda \|\boldsymbol{\beta}\|_1\} \tag{3.3}$$

Proof of (3.3)

Continue with equation (3.2)

$$\begin{aligned}
\max_{\boldsymbol{\beta}\in\mathbb{R}^p, \sigma>0} \pi(\boldsymbol{\beta}, \sigma|\mathbf{y}) &\propto \max_{\boldsymbol{\beta}\in\mathbb{R}^p, \sigma>0} h(\sigma) \exp\left(-\frac{n\lambda \|\beta\|_1}{2\sigma^2}\right) \exp\left(-\sum_{i=1}^{n} \frac{(y_i - \boldsymbol{x}_i^T\boldsymbol{\beta})^2}{2\sigma^2}\right) \\
&\propto \max_{\boldsymbol{\beta}\in\mathbb{R}^p, \sigma>0} h(\sigma) \exp\left(-\frac{1}{2\sigma^2} \|\mathbf{y} - \mathbf{X}\boldsymbol{\beta}\|_2^2 - \frac{n\lambda \|\beta\|_1}{2\sigma^2}\right) \\
&\propto \max_{\boldsymbol{\beta}\in\mathbb{R}^p, \sigma>0} h(\sigma) \exp\left(-\frac{n}{2\sigma^2} \left(\frac{1}{n} \|\mathbf{y} - \mathbf{X}\boldsymbol{\beta}\|_2^2 - \lambda \|\beta\|_1\right)\right)
\end{aligned}$$

where $h(\sigma) = (2\pi)^{-\frac{n}{2}} \left(\frac{4\sigma^2}{\lambda n}\right)^{-p} \sigma^{-n-1}$

Maximising that function with respect to $\boldsymbol{\beta}$ is equivalent to minimising the loss function $\frac{1}{n} \|\mathbf{y} - \mathbf{X}\boldsymbol{\beta}\|_2^2 + \lambda \|\boldsymbol{\beta}\|_1$

■

Therefore, a full Bayesian analysis can thus be performed.

3.1.3 Conflicting Prior Information in Lasso

In the read paper of Tibshirani (2011), Chris Holmes makes a comment about a potential problem about lasso directly connected to the main topic of this thesis: lasso does not allow to remove bias introduced by shrinking coefficients that should not be.

From a frequentist perspective, Holmes gives a different representation of the lasso estimator as follows

$$\{\tilde{\boldsymbol{\beta}}, \tilde{\boldsymbol{\eta}}\} = \underset{\boldsymbol{\beta},\boldsymbol{\eta}}{\mathrm{argmax}}\{l(\boldsymbol{\beta}) - \sum_j \eta_j^{-1}\beta_j^2\} \qquad \text{subject to} \sum_j \eta_j = t \tag{3.4}$$

where $l(\boldsymbol{\beta})$ has the form of a log-likelihood and t is a monotone function of λ in our representation. In this form, shrinkage of coefficients can also be realized, that is for the variable that is not relevant to the regression, $\eta_j \to 0$ and then $\tilde{\beta}_j = 0$. Essentially, as the sample size $n \to \infty$, we would wish $\eta_j \to \infty$ to remove bias for the variable that is relevant to the regression. But the problem is that there is a constraint in lasso, without setting $t \to \infty$.

From a Bayesian perspective, this issue can be seen from a differnet angle. As shown in the Bayesian interpretation of lasso (equation 3.2), the Laplace prior will put information around 0 and the tails of it are too slim. As a consequence, when the coefficient is significantly different from 0, there will be a conflict between the prior and the likelihood supported by the data, which is the conflicting prior information in lasso.

Regarding this problem, there are several researches studying other forms of penalty term of lasso. For example, Zou and Li (2008) proposed a new algorithm based on the local linear approximation (LLA) to maximize the penalized likelihood for a class of concave penalty functions while Zou (2006) put forward the Adaptive lasso where the adaptive weights are used for the penalty as mentioned above. However, both of them deal with this problem from a frequentist perspective rather than by studying on the prior. In the following, we continue this study by replacing the prior of lasso by a super heavy-tailed distribution that is presented in next section.

3.2 LPTL Distribution

The super heavy-tailed distribution, LRVD, is proposed to replace the Laplace prior in the Bayesian interpretation of lasso. In order to keep the variable selection property of lasso, the idea is to set the central part of this distribution to match exactly the Laplace density.

Hence, we propose a log-Pareto-tailed distribution (LPTL) with the central part of its density is exactly the Laplace for the prior, which is a super heavy-tailed distribution belonging to the family of LRVD. With a prespecified parameter $\rho \in (2P(1) - 1, 1) \approx (0.6321, 1)$, we denote it as LPTL($\rho$). More specifically, for the prior, we still have $\beta_j|\sigma \overset{d}{\sim} \frac{1}{2s}\, g(\frac{\beta_j}{s})$, but the function g has now been assumed to be

$$g(z) = \begin{cases} p(z) & if \quad |z| \leq \tau \\ p(\tau)\dfrac{\tau}{|z|}(\dfrac{log\tau}{log|z|})^{\psi+1} & if \quad |z| > \tau \end{cases} \tag{3.5}$$

where $p(.)$ is defined to be the density of a standard Laplace with location 0 and scale 1,

$\tau > 1$ and $\psi > 0$ are functions of ρ with

$$\begin{aligned} &\tau = P^{-1}((1+\rho)/2) := \{\tau : \mathbb{P}(-\tau \le Z \le \tau) = \rho \ for \ Z \overset{d}{\sim} p \\ &\psi = 2(1-\rho)^{-1} f(\tau)\, \tau \, log(\tau) \end{aligned} \tag{3.6}$$

where $P(\centerdot)$ and $P^{-1}(\centerdot)$ represents the cumulative distribution function (CDF) and inverse CDF of a standard Laplace distribution, respectively.

LPTL distribution is obtained by drawing inspiration from the LPTN that has been introduced in Section 1.2. The interpretations of the parameters of LPTL is as follows:

- τ: the quantile of LPTL distribution and controls the central density, a function of ρ;
- ψ: controls the tail decay, a function of ρ;
- ρ: the parameter we tune, which represents the mass of the central part that is exactly the standard Laplace density. As ρ increases, the function g approaches the Laplace, which is expected to give more similar results to lasso.

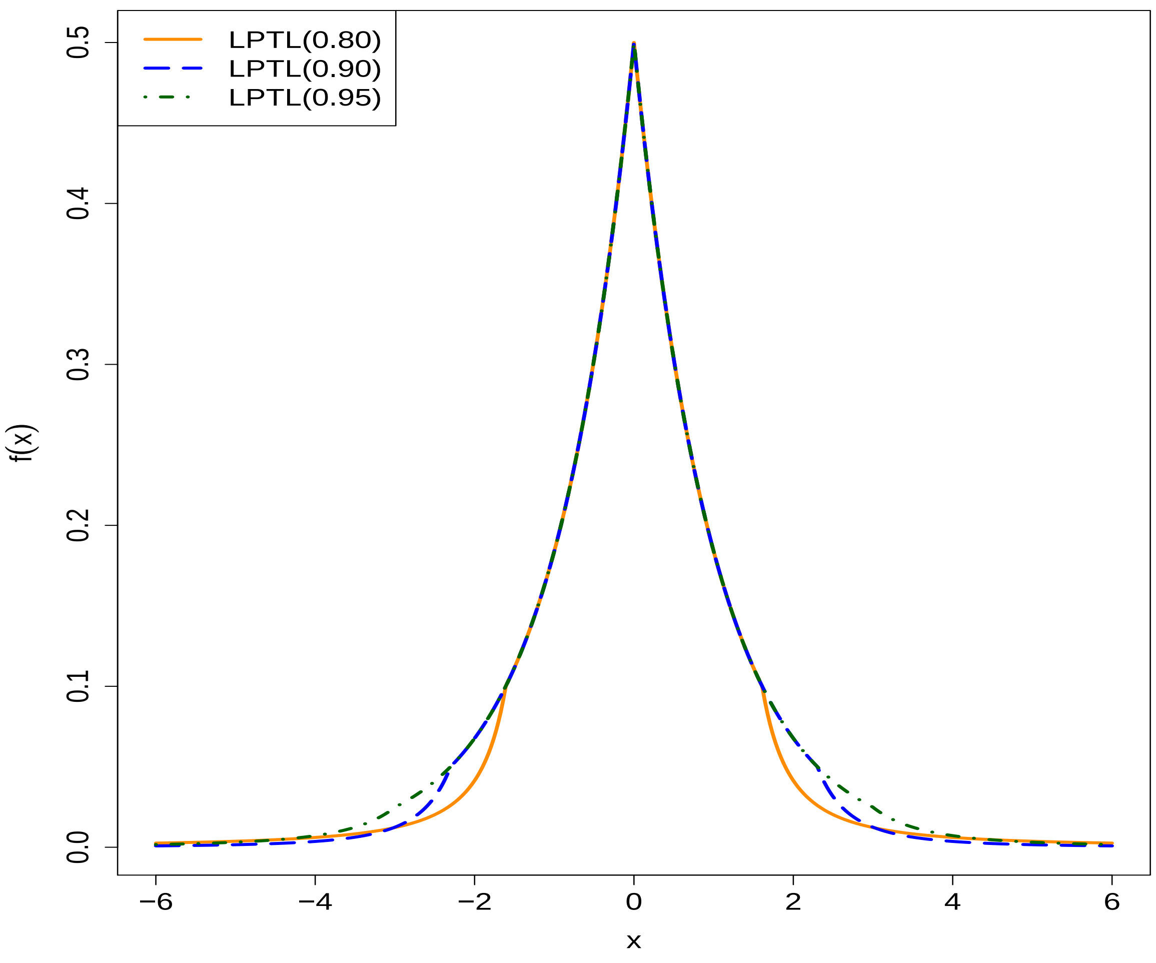

Figure 3.1: Plot of density of the LPTL(0.80), LPTL(0.90), LPTL(0.95).

In Figure 3.1, three different LPTL distributions with (0.80, 0.90, 0.95) of ρ are shown. From the plot, it is apparent that the mass of the central part still matches the (standard) Laplace density in order to remain the variable selection property of lasso. And with different values of ρ, the tail behaviour of LPTL varies so that we can control it and choose a relatively good value of ρ to deal with the conflicting prior information issue. $\rho = 0.95$ is a good value in our numerical analysis, which is the same as that of LPTN given by Gagnon et al. (2018) in the case of robustness against outliers.

To show the tail behaviour of the LPTL clearly, we enlarge the right tail of LPTL (0.90) and compare it with standard normal and standard Laplace density, shown in Figure 3.2. It is noticeable that the tail of LPTL is super heavy, which is obviously heavier than normal and also heavier than Laplace from some point around 5. Besides, it justifies that before the point around 3, the LPTL (0.9) overlaps the standard Laplace distribution. Furthermore, as shown in Figure 3.1, the LPTL distribution indeed have a strictly positive continuous PDF and is symmetric with respect to the origin, as we defined in Section 2.1, so that $z\,g(z)$ is monotone for $z > \tau$ as assumed in our framework.

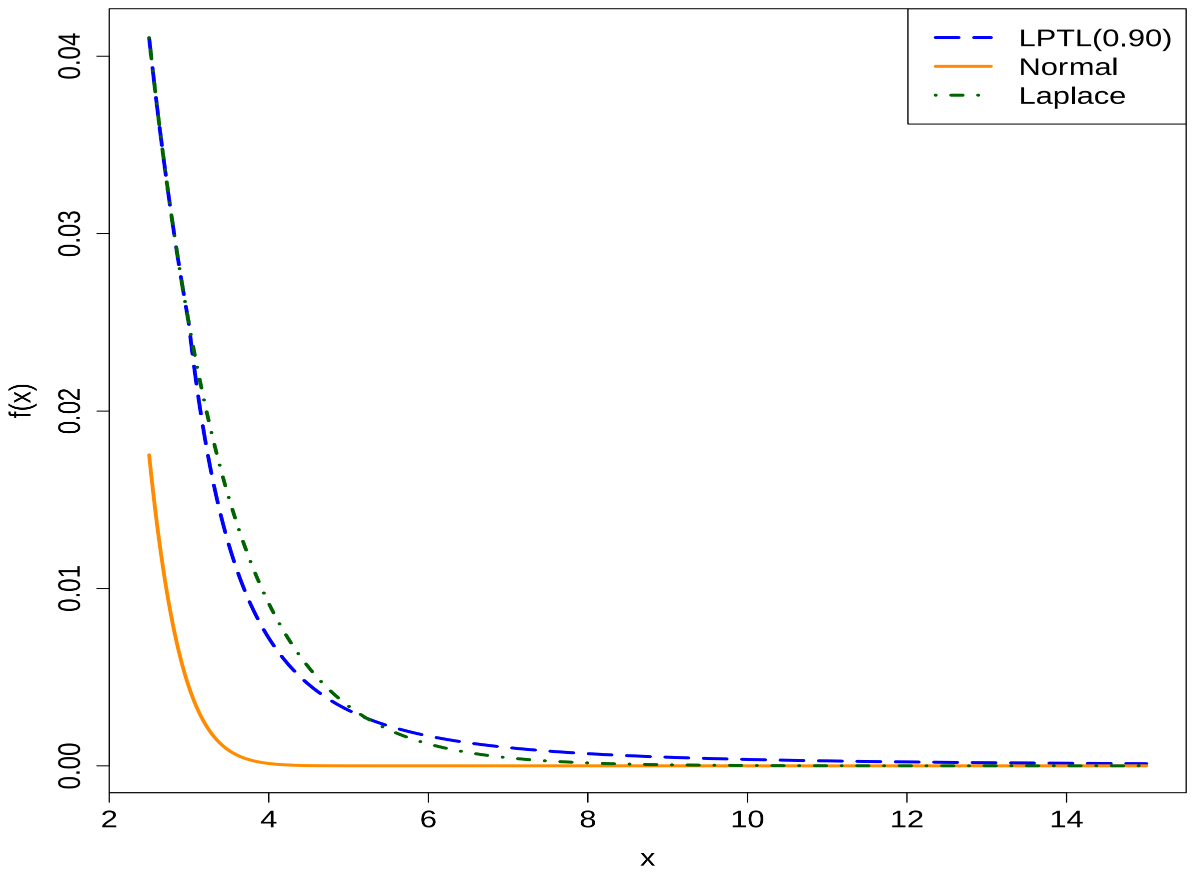

Figure 3.2: Plot of the behaviour of the right tails of LPTL(0.90), standard normal and standard Laplace.

3.3 Resolution of Conflicts for Lasso

In this section, we make a connection between the general framework and lasso case by presenting the limiting posterior density to show how resolution of conflicts translates into the case for lasso.

In lasso, when we standardised the design matrix and the response, the coefficients $\boldsymbol{\beta}$ are usually quite small, e.g. between -1 and 1. So it is not the case of the distance between the scale and the location of the slope being large as described in the **Case 1** of Section 2.3.1. But we notice that the $s(=\frac{2\sigma^2}{\lambda n})$ is always quite small since normally, n is large comparing with σ so when λ increase, s will become smaller and tends to 0. Therefore, it is a special case of the **Case 2** in Section 2.3.2 with $a_j = 0$ and $t(\sigma) = 2\sigma^2$ in the expression of s_j as well as $u_j = 0$ for $j = 1, \dots, p$. We now assume g is a LPTL distribution and $f := \mathcal{N}(0,1)$ in equation (2.4). Combining the idea given below the Proposition 2,1, as $\lambda n \to \infty$, we have the the numerator of limiting posterior density as follows

$$\begin{aligned}
\pi(\boldsymbol{\beta},\sigma)L(\boldsymbol{\beta},\sigma;\mathbf{y}) &= \pi(\sigma)\prod_{j=1}^{p}\frac{1}{s_j}\,g\Big(\frac{|\beta_j|}{s_j}\Big)\prod_{i=1}^{n}\frac{1}{\sigma}f\Big(\frac{y_i-\mathbf{x}_i^T\boldsymbol{\beta}}{\sigma}\Big)\\
&= \pi(\sigma)\prod_{j=1}^{p}\frac{\lambda n}{2\sigma^2}\cdot g\left(\frac{|\beta_j|}{\frac{2\sigma^2}{\lambda n}}\right)\prod_{i=1}^{n}\frac{1}{\sigma}f\Big(\frac{y_i-\mathbf{x}_i^T\boldsymbol{\beta}}{\sigma}\Big)\\
&= \pi(\sigma)\prod_{j=1}^{p}\frac{\lambda n}{|\beta_j|}\cdot\frac{1}{\frac{2\sigma^2}{|\beta_j|}}\cdot g\left(\frac{\lambda n}{\frac{2\sigma^2}{|\beta_j|}}\right)\prod_{i=1}^{n}\frac{1}{\sigma}f\Big(\frac{y_i-\mathbf{x}_i^T\boldsymbol{\beta}}{\sigma}\Big)\\
&\xrightarrow{\text{I}} \pi(\sigma)\prod_{j=1}^{p}\frac{1}{|\beta_j|}\cdot\lambda n\cdot g(\lambda n)\prod_{i=1}^{n}\frac{1}{\sigma}f\Big(\frac{y_i-\mathbf{x}_i^T\boldsymbol{\beta}}{\sigma}\Big)\quad\text{as}\quad \lambda n\to\infty
\end{aligned}$$

where step I is due to the Proposition 2.1 with $z = \lambda n$. For the limiting posterior density, $\lambda n\, g(\lambda n)$ will be cancel out with that at the denominator in the marginal, provided that the integral can be interchanged with the limit. Then the limiting posterior density for the lasso case is a special case of $\pi^d(\boldsymbol{\beta},\sigma|\mathbf{y})$ in **Case 2** with $\mu_j = 0$ and $k_j = 0$, given by

$$\pi^l(\boldsymbol{\beta},\sigma|\mathbf{y}) = \pi(\sigma)[m^l(\mathbf{y})]^{-1}\prod_{j=1}^{p}\frac{1}{|\beta_j|}\prod_{i=1}^{n}\frac{1}{\sigma}f\Big(\frac{y_i-\mathbf{x}_i^T\boldsymbol{\beta}}{\sigma}\Big) \tag{3.7}$$

where $m^l(\mathbf{y}) = \int_{\mathbb{R}^p}\int_0^\infty \pi(\sigma)\prod_{j=1}^{p}\frac{1}{|\beta_j|}\prod_{i=1}^{n}\frac{1}{\sigma}f\left(\frac{y_i-\mathbf{x}_i^T\boldsymbol{\beta}}{\sigma}\right)d\boldsymbol{\beta}d\sigma$

Essentially, it implies that in the limit, all prior information for $\boldsymbol{\beta}$ is conflicting. The reason is that in lasso case, s_j is so small that $\frac{|\beta_j|}{s}$ (the standardized β_j) is larger than τ (the quantile of LPTL), which means $\frac{|\beta_j|}{s}$ is located on the heavy tail (log-Pareto tail) rather than the central part of Laplace density. Hence, all prior will be regraded as conflicting information. Indeed, the prior we obtained in the limiting posterior ($\frac{1}{|\beta_j|}$) has much less penalisation on β_j than $\frac{\lambda n}{2\sigma^2}\exp(-\frac{\lambda n|\beta_j|)}{2\sigma^2}$, the laplace prior in lasso. We expect that in practice, it should be in between with siginificant parameters with priors behaving as $\frac{1}{|\beta_j|}$ and non-significant parameters with priors behaving like the usual Laplace.

3.4 Convexity of the Objective Function

In the retrospective paper of Tibshirani (2011), the convexity of the loss function of lasso and its computational advantage are mentioned both in the comment of Peter Bühlmann and the author's response. We now study on the convexity of the objective function of our approach.

First of all, we replace the Laplace prior in equation (3.2) by LPTL to write down the posterior density for our model.

$$\pi(\boldsymbol{\beta}, \sigma|\mathbf{y}) \propto \begin{cases} \left(\prod_{j=1}^{p} \frac{1}{2s} \exp(-\frac{|\beta_j|}{s})\right) \frac{1}{\sigma} \prod_{i=1}^{n} \frac{1}{\sqrt{2\pi}\sigma} \exp\left(-\frac{(y_i - \mathbf{x}_i^T \boldsymbol{\beta})^2}{2\sigma^2}\right) \quad if\ \frac{|\beta_j|}{s} < \tau \\ \left(\prod_{j=1}^{p} \frac{1}{2s} \exp(-\psi) \frac{\psi}{\frac{|\beta_j|}{s}} \left[\frac{log\psi}{log(\frac{|\beta_j|}{s})}\right]^{\psi+1}\right) \frac{1}{\sigma} \prod_{i=1}^{n} \frac{1}{\sqrt{2\pi}\sigma} \exp\left(-\frac{(y_i - \mathbf{x}_i^T \boldsymbol{\beta})^2}{2\sigma^2}\right) \\ \qquad if\ \frac{|\beta_j|}{s} > \tau \end{cases} \tag{3.8}$$

Next, we take $-\log$ of the posterior density to transform it to the loss function. Since we have shown in Section 3.1.2 that maximizing the posterior is equivalent to minimizing the loss function, we then discuss the convexity of loss function piecewise.

For the first case, i.e. $\frac{|\beta_j|}{s} < \tau$, it is exactly the lasso case and the loss function is convex. The second case seems to be complicated so we simplify it to two parameters: β and σ as well as one observation $y \sim \mathcal{N}(\widehat{\beta}^{OLS}, \sigma)$. The objective function becomes:

$$\begin{aligned} L(\beta, \sigma) &= -\log(\pi(\beta, \sigma|y)) \\ &\propto \frac{1}{2\sigma^2}(y - x\beta)^2 + 2\log\sigma + \log(2|\beta|) + (\psi+1)\log\left(\log\left(\frac{|\beta|\lambda}{2\sigma^2}\right)\right) \end{aligned} \tag{3.9}$$

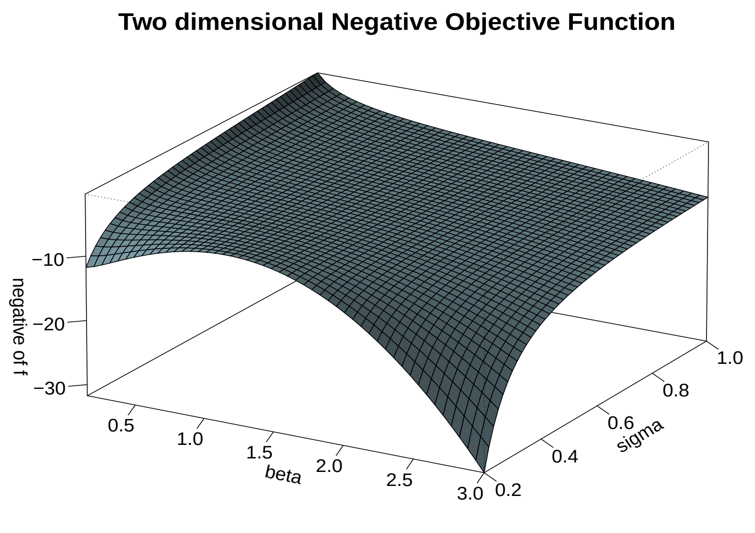

Figure 3.3: 3-D plot of negative objective function with respect to β and σ ($\lambda = 50$)

Then we fix $y = 1$, $x = 0.8$ to plot the loss function with respect to β and σ. To show the bottom of the plot clearly, we use the negative of the objection function (Figure 3.3). And to make sure it satisfy the condition, i.e. $\frac{|\beta_j|}{s} > \tau$, we take β from 0.15 to 3, σ from 0.2 to 1 and fix $\lambda = 50$. The plots of positive objective function and the one with $\lambda = 100$ are in **Appendix**.

Figure 3.3 indicates that the objective function of this simple case is non-convex. And we may infer that with $p+1$ dimensions of parameters and n observations, the objective function of our approach is till non-convex. To further study this case, we rewrite the loss function as a draft containing two parts:

$$\begin{aligned} L(\boldsymbol{\beta}, \sigma) &= -\log(\pi(\boldsymbol{\beta}, \sigma|\mathbf{y}) \\ &\propto -\log\{\text{likelihood}\} + \{-\log\{\text{prior density}\}\} \\ &\propto \text{RSS} + \text{Part 2} \end{aligned}$$

where RSS represents the residual sum of squares, which is convex with respect to $\boldsymbol{\beta}$ obviously. We then focus on the convexity of Part 2. To simplify, we fix $\sigma = 0.5$.

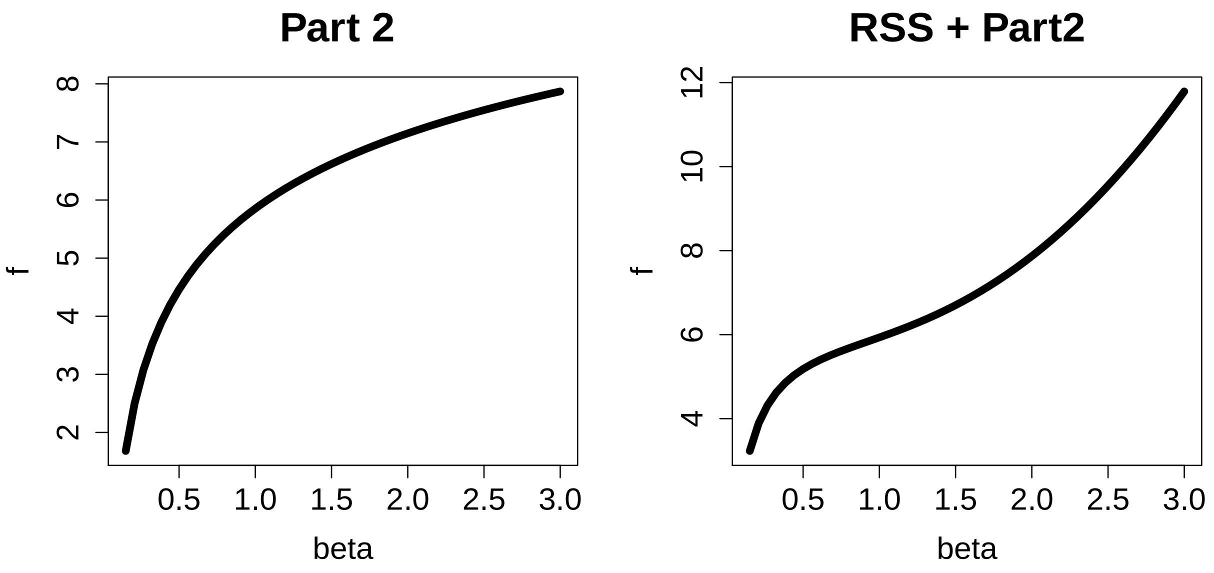

Figure 3.4: **Left:** Plot of "Part 2" function with respect to β; **Right:** Plot of "RSS + Part 2" function with respect to β.

Using the simple case above, i.e. one parameter and one observation, we obtain Figure 3.4. From it, it is clear that the "Part 2" function is concave but when the RSS term is added, the function starts to change towards convex. Note that $n = 1$ and $p = 1$ here, but normally $n \gg p$ in practice, so we can make an educated guess that when $n \gg p$, the function will be dominated by RSS and converges to a function with the convexity property. Therefore, when implementing our approach to the real data with large n, we can consider that the objective function is nearly convex. But strictly, it is not convex and the computation may be a downside

3.5 Lasso vs. LPTL Model

In this section, we investigate the difference between lasso and LPTL model by studying the relationships between their estimates and the OLS estimate respectively. Also we use a simulated data that contains $n = 20$ data points and 3 parameters (scale σ, intercept β_0, slope β_1) to visualize the difference between lasso and our approach, shown in Figure 3.5 below.

We set the coefficients estimated from the lasso model by $\widehat{\boldsymbol{\beta}}^{LASSO}$ while the coefficients estimated from the model with LPTL prior by $\widehat{\boldsymbol{\beta}}^{LPTL}$. As the reader may know, the regression coefficients $\boldsymbol{\beta}$ are estimated using ordinary least squares (OLS) method, corresponding to Bayesian estimates (Setting the non-informative prior on $\boldsymbol{\beta}$) and maximum likelihood estimates. This is noticed by looking at the density in Section 3.1.2 and by replacing the prior by $\pi(\boldsymbol{\beta} \mid \sigma) \propto 1$. The OLS estimate is given by:

$$\widehat{\boldsymbol{\beta}}^{OLS} = (\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\boldsymbol{y} \tag{3.10}$$

We now investigate the relationship between them to highlight the differences. To simplify, we assume that the columns of design matrix $\mathbf{X}$ are orthogonal. And from section 3.1, we know that the data points on each column are standardized, by that we mean that $\sum_{i=1}^{n} x_{ij} = 0$ and $\sum_{i=1}^{n} x_{ij}^2 = N$, for $j = 1, ..., p$. Combining the assumption of orthogonality of the columns, we have

$$\mathbf{X}^T\mathbf{X} = \begin{pmatrix} n & & \\ & \ddots & \\ & & n \end{pmatrix} = n\,\mathbf{I} \tag{3.11}$$

Using the above assumptions, we get the following relationships.

(i) The relationships between $\widehat{\boldsymbol{\beta}}^{LASSO}$ and $\widehat{\boldsymbol{\beta}}^{OLS}$ is:

$$\widehat{\beta}_j^{LASSO} = \begin{cases} \widehat{\beta}_j^{OLS} \max\{0, 1 - \dfrac{\lambda}{2|\widehat{\beta}_j^{OLS}|}\} & if\, \widehat{\beta}_j^{OLS} \neq 0 \\ 0 & if\, \widehat{\beta}_j^{OLS} = 0 \end{cases} \tag{3.12}$$

From the relationship, we find the fact that $\widehat{\beta}_j^{LASSO}$ is exactly 0 when $|\widehat{\beta}_j^{OLS}| < \frac{\lambda}{2}$, which shows the variable selection feature of lasso. Also, depending on the relationship between $\widehat{\beta}_j^{OLS}$ and $\widehat{\beta}_j^{LASSO}$ is multiplied by a factor less than one, it shows that there is an impact of the conflict of information. In Figure 3.5 (red line), we choose $\lambda = 1$ (Left plot) and $\lambda = 2$ (Right plot) to visualize this relationship using numerical optimization, i.e. MAP estimation of lasso. From both plots, the red lines shows that the threshold is exactly $\frac{\lambda}{2}$, i.e. 0.5 in the left plot and 1 in the right plot, and beyond this threshold, $\widehat{\beta}_j^{LASSO}$ increases linearly with $\widehat{\beta}_j^{OLS}$, which is exactly what we expect in the formula.

(ii) The relationship between $\widehat{\boldsymbol{\beta}}^{LPTL}$ and $\widehat{\boldsymbol{\beta}}^{OLS}$ is:

$$\begin{cases} \widehat{\beta}_j^{LPTL} = \widehat{\beta}_j^{OLS} \max\{0, 1 - \dfrac{\lambda}{2|\widehat{\beta}_j^{OLS}|}\} & for\ \dfrac{|\widehat{\beta}_j^{LPTL}|}{s(\widehat{\sigma})} \leq \tau \,\&\, \widehat{\beta}_j^{OLS} \neq 0 \\ \widehat{\beta}_j^{LPTL} + sign(\widehat{\beta}_j^{OLS}) \dfrac{\widehat{\sigma}^2 \left(\log\left(\frac{|\widehat{\beta}_j^{LPTL}|}{s(\widehat{\sigma})}\right) + \psi + 1 \right)}{n\,|\widehat{\beta}_j^{LPTL}| \log\left(\frac{|\widehat{\beta}_j^{LPTL}|}{s(\widehat{\sigma})}\right)} = \widehat{\beta}_j^{OLS} & for\ \dfrac{|\widehat{\beta}_j^{LPTL}|}{s(\widehat{\sigma})} > \tau \,\&\, \widehat{\beta}_j^{OLS} \neq 0 \\ \widehat{\beta}_j^{LPTL} = 0 & for\ \widehat{\beta}_j^{OLS} = 0 \end{cases} \tag{3.13}$$

This formula shows that the relationship between $\widehat{\beta}_j^{LPTL}$ and $\widehat{\beta}_j^{OLS}$ is piecewise, mainly in two cases. In the first case, i.e. no conflicting prior information, $\widehat{\beta}_j^{LPTL}$ has the same formula as $\widehat{\beta}_j^{LASSO}$. When there exists conflicting information on β_j (second case), the formula of relationship seems quite complicated. However, it can be simplified as:

$$\widehat{\beta}_j^{LPTL} + sign(\widehat{\beta}_j^{OLS})\frac{M}{n} = \widehat{\beta}_j^{OLS}, \quad \text{where } M = \frac{\widehat{\sigma}^2 \left(\log\left(\frac{|\widehat{\beta}_j^{LPTL}|}{s(\widehat{\sigma})}\right) + \psi + 1 \right)}{|\widehat{\beta}_j^{LPTL}| \log\left(\frac{|\widehat{\beta}_j^{LPTL}|}{s(\widehat{\sigma})}\right)}$$

Normally, n is quite large comparing to M, thus $\frac{M}{n} \to 0$ so that $\widehat{\beta}_j^{LPTL} \approx \widehat{\beta}_j^{OLS}$, reflecting that the conflict is resolved. The relationship also visualized in Figure 3.5 (black line), where the jump point indicates a threshold that separate these two cases, beyond which $\widehat{\beta}_j^{LPTL}$ is nearly equal to $\widehat{\beta}_j^{OLS}$, otherwise it behaves the same as $\widehat{\beta}_j^{LASSO}$, satisfying the relationship and achieving our goal.

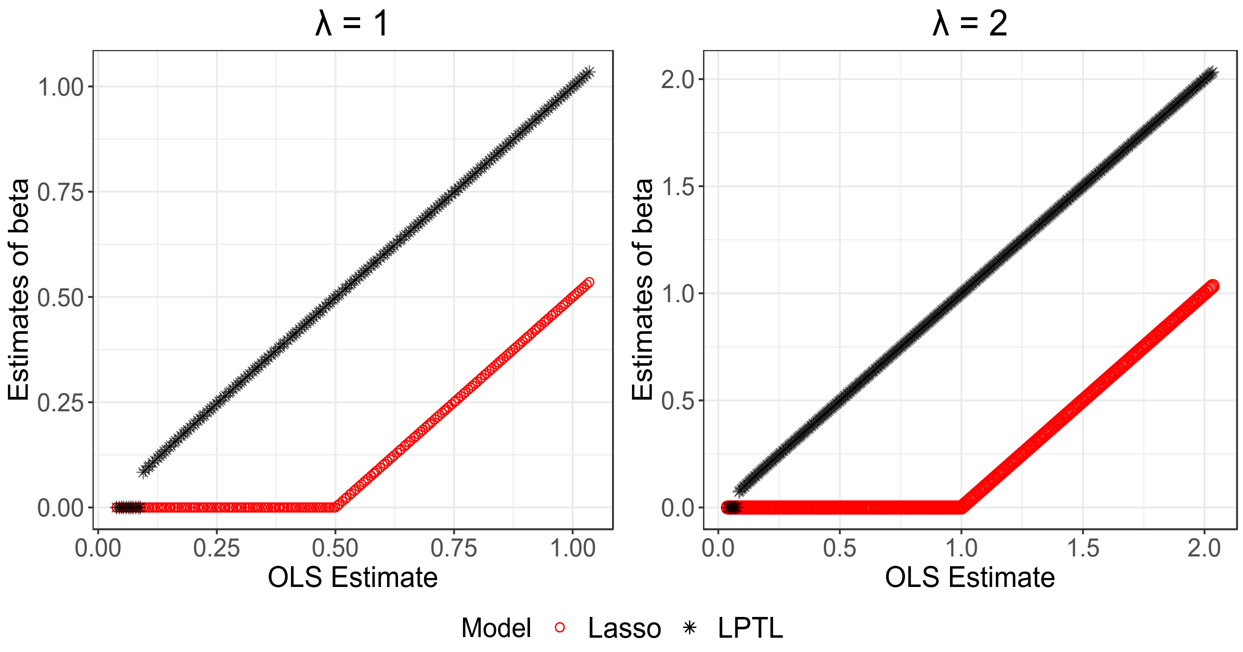

Figure 3.5: **Left:** Plot of Lasso estimate vs. LPTL estimate ($\lambda = 1$); **Right** Plot of Lasso estimate vs. LPTL estimate ($\lambda = 2$).

We combine LPTL estimate and Lasso estimate in on plot in Figure 3.5 to compare them. For small values of $\widehat{\beta}_j^{OLS}$, $\widehat{\beta}_j^{LPTL}$ and $\widehat{\beta}_j^{LASSO}$ are both equal to 0 while as $\widehat{\beta}_j^{OLS}$ increases, $\widehat{\beta}_j^{LPTL}$ will jump at some point and reach a value close to OLS estimate and remain this feature as $\widehat{\beta}_j^{OLS}$ increases, indicating the conflicts are resolved. Beyond that point, however, lasso estimate still remains 0 until $\frac{\lambda}{2}$, the threshold, beyond which it starts to increase linearly with OLS estimate but always shows an impact of "shrinkage". These are the main differences between LPTL estimates and Lasso estimates. Also, we notice that the change of λ has more impact on the threshold of Lasso estimate than the jump point of LPTL estimate, possibly because the jump point of LPTL estimate also depends on σ. In general, the figure shows a good result of LPTL model that attains our objective. In the real data analysis part, we will continue to compare these two models using a real dataset.

The proofs of the two formula (3.12), (3.13) are given in the following.

Proof of (3.12)

From (3.1), we have the loss function for lasso

$$\begin{aligned}
L(\boldsymbol{\beta}) &= \frac{1}{n}\left\|\mathbf{y}-\mathbf{X}\boldsymbol{\beta}\right\|_2^2+\lambda\left\|\boldsymbol{\beta}\right\|_1 \\
&= (\mathbf{y}^T\mathbf{y}-2\boldsymbol{\beta}^T\mathbf{X}^T\mathbf{y}+\boldsymbol{\beta}^T\mathbf{X}^T\mathbf{X}\boldsymbol{\beta})+\lambda\left\|\boldsymbol{\beta}\right\|_1 \\
&\overset{\text{I}}{\propto} -\frac{2}{n}\boldsymbol{\beta}^T\cdot n\widehat{\boldsymbol{\beta}}^{OLS}+\boldsymbol{\beta}^T\boldsymbol{\beta}+\lambda\sum_{j=1}^{p}|\beta_j| \\
&= \sum_{j=1}^{p} -2\beta_j\widehat{\beta}_j^{OLS}+\beta_j^2+\lambda|\beta_j|
\end{aligned}$$

In step I, based on equation (3.10) and (3.11), we have $\mathbf{X}^T\mathbf{X}=n\,\mathbf{I}$ and $\mathbf{X}^T\boldsymbol{y}=n\,\widehat{\boldsymbol{\beta}}^{OLS}$.

Let $L_j=-2\beta_j\widehat{\beta}_j^{OLS}+\beta_j^2+\lambda|\beta_j|$ and differentiate it with respect to β_j (for $\beta_j\neq 0$), we get

$$\frac{\partial L_j}{\partial\beta_j}=-2\widehat{\beta}_j^{OLS}+2\beta_j+\lambda\frac{\beta_j}{|\beta_j|}$$

Setting $\left.\frac{\partial L_j}{\partial\beta_j}\right|_{\beta_j=\widehat{\beta}_j^{LASSO}}=0$, we have

$$\widehat{\beta}_j^{LASSO}=\widehat{\beta}_j^{OLS}-sign(\widehat{\beta}_j^{LASSO})\cdot\frac{\lambda}{2}$$

If $\widehat{\beta}_j^{LASSO}>0$, we must have$\widehat{\beta}_j^{OLS}>\frac{\lambda}{2}>0$, otherwise $\widehat{\beta}_j^{LASSO}=0$. Similarly, if $\widehat{\beta}_j^{LASSO}<0$, then $\widehat{\beta}_j^{OLS}<-\frac{\lambda}{2}<0$, otherwise $\widehat{\beta}_j^{LASSO}=0$. It also shows that $\widehat{\beta}_j^{LASSO}$ and $\widehat{\beta}_j^{OLS}$ have the same sign, i.e. $sign(\widehat{\beta}_j^{LASSO})=sign(\widehat{\beta}_j^{OLS})$, finally we have

$$\widehat{\beta}_j^{LASSO}=\begin{cases}\widehat{\beta}_j^{OLS}\max\{0,1-\dfrac{\lambda}{2|\widehat{\beta}_j^{OLS}|}\} & if\ \widehat{\beta}_j^{OLS}\neq 0\\[2ex] 0 & if\ \widehat{\beta}_j^{OLS}=0\end{cases}$$

End of Proof ■

Proof of (3.13)

We derive (3.13) based on the posterior density of (3.8). Since the posterior density is piecewise because of its LPTL prior, we discuss by splitting it into the following two cases.

Case 1. $\frac{|\beta_j|}{s} < \tau$, i.e. no conflicting prior information on β_j.

In this case, we have

$$\pi(\boldsymbol{\beta}, \sigma|\mathbf{y}) \propto \left(\prod_{j=1}^{p} \frac{1}{2s} \exp(-\frac{|\beta_j|}{s})\right) \frac{1}{\sigma} \prod_{i=1}^{n} \frac{1}{\sqrt{2\pi\sigma}} \exp(-\frac{(y_i - \boldsymbol{x}_i^T \boldsymbol{\beta})^2}{2\sigma^2})$$

It is exactly the same as the posterior density of lasso, so we just replace the $\widehat{\beta}_j^{LASSO}$ by $\widehat{\beta}_j^{LPTL}$ in the formula (3.12). Thus,

$$\widehat{\beta}_j^{LPTL} = \begin{cases} \widehat{\beta}_j^{OLS} \max\{0, 1 - \frac{\lambda}{2|\widehat{\beta}_j^{OLS}|}\} & if\, \widehat{\beta}_j^{OLS} \neq 0 \\ 0 & if\, \widehat{\beta}_j^{OLS} = 0 \end{cases}$$

Case 2. $\frac{|\beta_j|}{s} > \tau$, i.e. there exists conflicting prior information on β_j.

In this case, we have

$$\pi(\boldsymbol{\beta}, \sigma|\mathbf{y}) \propto \left(\prod_{j=1}^{p} \frac{1}{2s} \exp(-\psi) \frac{\psi}{\frac{|\beta_j|}{s}} \left[\frac{log\psi}{log(\frac{|\beta_j|}{s})}\right]^{\psi+1}\right) \frac{1}{\sigma} \prod_{i=1}^{n} \frac{1}{\sqrt{2\pi\sigma}} \exp(-\frac{(y_i - \boldsymbol{x}_i^T \boldsymbol{\beta})^2}{2\sigma^2})$$

Similar as what we did in Section 3.1.2, we firstly transform from maximizing $\pi(\boldsymbol{\beta}, \sigma|\mathbf{y})$ to minimizing the loss function by $-\log$,

$$\begin{aligned} L(\boldsymbol{\beta}, \sigma) &= -\log \pi(\boldsymbol{\beta}, \sigma|\mathbf{y}) \\ &\propto \frac{1}{2\sigma^2}(\mathbf{y}^T\mathbf{y} - 2\boldsymbol{\beta}^T\mathbf{X}^T\mathbf{y} + \boldsymbol{\beta}^T\mathbf{X}^T\mathbf{X}\boldsymbol{\beta}) + (n+1)\log\sigma + p\log(2s) \\ &\qquad + \sum_{i=1}^{p} \log\left(\frac{|\beta_j|}{s}\right) + (\psi+1)\log\left(\log\left(\frac{|\beta_j|}{s}\right)\right) \\ &\propto \frac{1}{2\sigma^2}(-2\boldsymbol{\beta}^T\mathbf{X}^T\mathbf{y} + \boldsymbol{\beta}^T\mathbf{X}^T\mathbf{X}\boldsymbol{\beta}) + (n+1)\log\sigma + p\log(2s) \\ &\qquad + \sum_{i=1}^{p} \log\left(\frac{|\beta_j|}{s}\right) + (\psi+1)\log\left(\log\left(\frac{|\beta_j|}{s}\right)\right) \\ &= \frac{1}{2\sigma^2}(-2n\boldsymbol{\beta}^T\widehat{\boldsymbol{\beta}}^{OLS} + n\boldsymbol{\beta}^T\boldsymbol{\beta}) + (n+1)\log\sigma + p\log(2s) \\ &\qquad + \sum_{i=1}^{p} \log\left(\frac{|\beta_j|}{s}\right) + (\psi+1)\log\left(\log\left(\frac{|\beta_j|}{s}\right)\right) \\ &= \sum_{j=1}^{p} -\frac{n}{\sigma^2}\widehat{\beta}_j^{\,OLS}\beta_j + \frac{n}{2\sigma^2}\beta_j^2 + \log\left(\frac{|\beta_j|}{s}\right) + (\psi+1)\log\left(\log\left(\frac{|\beta_j|}{s}\right)\right) \\ &\qquad + \frac{1}{p}(n+1)\log\sigma + \log(2s) \end{aligned}$$

Let

$$L_j = -\frac{n}{\sigma^2}\widehat{\beta}_j^{OLS}\beta_j + \frac{n}{2\sigma^2}\beta_j^2 + \log\left(\frac{|\beta_j|}{s}\right) + (\psi+1)\log\left(\log\left(\frac{|\beta_j|}{s}\right)\right)$$
$$+\frac{1}{p}(n+1)\log\sigma + \log(2s)$$

Differentiate L_j with respect to β_j (for $\beta_j \neq 0$),

$$\frac{\partial L_j}{\partial \beta_j} = -\frac{n}{\sigma^2}\widehat{\beta}_j^{OLS} + \frac{n}{\sigma^2}\beta_j + \frac{1}{\beta_j} + \frac{\psi+1}{\beta_j \log(\frac{|\beta_j|}{s})}$$

Evaluate at $\beta_j = \widehat{\beta}_j^{LPTL}$ and $\sigma = \hat{\sigma}$, we have $\frac{\partial L_j}{\partial \beta_j}\Big|_{\beta_j=\widehat{\beta}_j^{LPTL},\, \sigma=\hat{\sigma}} = 0$, i.e.

$$\widehat{\beta}_j^{LPTL} + sign(\widehat{\beta}_j^{LPTL})\frac{\widehat{\sigma}^2\left(\log\left(\frac{|\widehat{\beta}_j^{LPTL}|}{s(\widehat{\sigma})}\right)+\psi+1\right)}{n\,|\widehat{\beta}_j^{LPTL}|\,\log\left(\frac{|\widehat{\beta}_j^{LPTL}|}{s(\widehat{\sigma})}\right)} = \widehat{\beta}_j^{OLS}$$

If $\widehat{\beta}_j^{LPTL} > 0$, then

$$\widehat{\beta}_j^{OLS} > \frac{\widehat{\sigma}^2\left(\log\left(\frac{|\widehat{\beta}_j^{LPTL}|}{s(\widehat{\sigma})}\right)+\psi+1\right)}{n\,|\widehat{\beta}_j^{LPTL}|\,\log\left(\frac{|\widehat{\beta}_j^{LPTL}|}{s(\widehat{\sigma})}\right)} > 0$$

as in this case, $\frac{|\beta_j|}{s} > \tau > 1$, i.e. $\log\left(\frac{|\beta_j|}{s(\widehat{\sigma})}\right) > 0$ Similarly, If $\widehat{\beta}_j^{LPTL} < 0$, $\widehat{\beta}_j^{OLS} < 0$

Therefore, $\widehat{\beta}_j^{LPTL}$ and $\widehat{\beta}_j^{OLS}$ has the same sign. And Finally we have

$$\begin{cases} \widehat{\beta}_j^{LPTL} + sign(\widehat{\beta}_j^{OLS})\dfrac{\widehat{\sigma}^2\left(\log\left(\frac{|\widehat{\beta}_j^{LPTL}|}{s(\widehat{\sigma})}\right)+\psi+1\right)}{n\,|\widehat{\beta}_j^{LPTL}|\,\log\left(\frac{|\widehat{\beta}_j^{LPTL}|}{s(\widehat{\sigma})}\right)} = \widehat{\beta}_j^{OLS} & if\, \widehat{\beta}_j^{OLS} \neq 0 \\ 0 & if\, \widehat{\beta}_j^{OLS} = 0 \end{cases}$$

Combining Case 1 and Case 2, we obtain the following relationship:

$$\begin{cases} \widehat{\beta}_j^{LPTL} = \widehat{\beta}_j^{OLS}\max\{0, 1-\dfrac{\lambda}{2|\widehat{\beta}_j^{OLS}|}\} & for\, \dfrac{|\widehat{\beta}_j^{LPTL}|}{s(\widehat{\sigma})} \leq \tau \,\&\, \widehat{\beta}_j^{OLS} \neq 0 \\ \widehat{\beta}_j^{LPTL} + sign(\widehat{\beta}_j^{OLS})\dfrac{\widehat{\sigma}^2\left(\log\left(\frac{|\widehat{\beta}_j^{LPTL}|}{s(\widehat{\sigma})}\right)+\psi+1\right)}{n\,|\widehat{\beta}_j^{LPTL}|\,\log\left(\frac{|\widehat{\beta}_j^{LPTL}|}{s(\widehat{\sigma})}\right)} = \widehat{\beta}_j^{OLS} & for\, \dfrac{|\widehat{\beta}_j^{LPTL}|}{s(\widehat{\sigma})} > \tau \,\&\, \widehat{\beta}_j^{OLS} \neq 0 \\ \widehat{\beta}_j^{LPTL} = 0 & for\, \widehat{\beta}_j^{OLS} = 0 \end{cases}$$

End of Proof ■

Chapter 4

Real Data Analysis

4.1 Dataset Description

National Basketball Association (NBA) is a professional basketball league in the United States (and Canada), which brings together the best basketball players in the world. Also, among any other professional sports league, the average salary of NBA is the highest. To a large extent, the player's salary is decided by their performance in each season.

Thus, to empirically study the performance of the proposed approach and to establish a connection between NBA players' salary and their performance, we analyse the the latest player's salary and performance data of 2018/19 season. The salary dataset is from the "2018-19 NBA Player Contracts" on the Basketball-Reference (2018a) website, which provides the official NBA data for current NBA seasons. The players' performance dataset is from "2018-19 NBA Player Stats: Advanced" (Basketball-Reference (2018b)).

First of all, we notice that there are duplicate player's name, i.e. duplicate rows, because some players may transfer from one team to another in one year. For such players, we remain his performance data of the whole season but only use his current team as his "team" data. To illustrate clearly, we give an example in Table 4.1, where TOT means one player transfer teams in one season and the row including TOT contains the whole performance data of this player. For instance, C.J Miles transfer from team MEM to TOR, so we use the first row as his performance data but replace TOT with TOR (current team) as his team value. And we delete duplicate rows, i.e. 2nd and 3rd rows in this example.

	Player name	**Position**	**Team**	**Number of games**
1	C.J. Miles	SF	TOT	53
2	C.J. Miles	SF	TOR	40
3	C.J. Miles	SF	MEM	13

Table 4.1: Example of one player who transfer teams

Since there are two data sets: "salary" and "performance", we combine them together according to the name of players. There is only one row containing missing values; we exclude it. In total, the integrated dataset consists 480 rows of observations and 25 features. The descriptionof all the variables is given in Table 4.2.

Variable	**Description**
Age	Player's age.
Team	Player's current team, 30 teams in total.
Position	Player's position on the filed (11 levels). Usually we use 5 positions: PG (Point Guard), SG (shooting Guard), SF (Small Forward), PF (Power Forward) and C (Center). Some players are suitable for more than one position, e.g. SG-PF.
G	The total games the player took in 18/19 season.
MP	The total minutes on the field, each game has 48 minutes.
PER	Player Efficiency Rating, which is an important index that calculates the player's accomplishment to indicate performance. The league average is 15.
TS.	True Shooting Percentage. Shooting efficiency, taking into account 2-point/3-point field goals, and free throw.
X3PAr	3-Point Attempt Rate.
FTr	Free Throw Attempt Rate.
TRB.	Total Rebound Percentage. The percentage of a player grabbing the rebounds.
ORB.	Offensive Rebound Percentage. Offensive means team is in the attack.
DRB.	Defensive Rebound Percentage. Defensive means team is on the defensive.
AST.	Assist Percentage. The percentage of teammate goals that a player assisted.
STL.	Steal Percentage. The percentage of opponent possessions that end with a steal by the player.
BLK.	Block Percentage. The percentage of opponent attempts blocked by the player.
TOV.	Turnover Percentage. An estimate of turnovers per 100 plays
USG.	Usage Percentage. The percentage of team plays used by a player.
WS	Win Shares. The number of wins contributed by a player.
OWS	Offensive Win Shares. Contributed by the player's offense ability.
DWS	Defensive Win Shares.
WS.48	Win Shares Per 48 Minutes.
BPM	Box Plus/Minus. A kind of estimate (box score) of the points per 100 possessions a player contributed, translated to an average team.
OBPM	Offensive Box Plus/Minus.
DBPM	Defensive Box Plus/Minus.
VORP	Value over Replacement Player. A kind of estimate (box score) of the points per 100 possessions that a player contributed above a replacement player, translated to an average team.

Table 4.2: Description of 25 variables.

4.2 Exploratory Analysis

After reconstructing the data set, we do the data exploratory analysis in order to visualize some patterns among data and have a overview of the connection between salary and some key features such as PER and MP before modelling.

The correlation plot below indicates that most of the variables have low correlations. But some variables such as **X3PAr**, **TRB**, **WS**, **BPM** and **VORP** have large correlations with some others. To avoid multicollinearity, we exclude these five variables. From the meaning of them given in Table 4.2, **X3PAr** has a negative relation with **FTr**. **TRB** is a sum of the offensive (**ORB**) and the defensive performance (**DRB**), similarly for **WS** and **BPM**. **VORP** is related to three BPM variables from its description. Therefore, they are the redundant information and it is reasonable to delete them. The variable **Position** also performs bad and we replace it with two new dummy variables, described in the following. The new correlation matrix is shown in **Appendix**. The correlations between features and the response indicates that around 10 features are closely related with salary such as **MP** and **AST** but the variables with complicated formulas such as **OWS** have low correlations with salary.

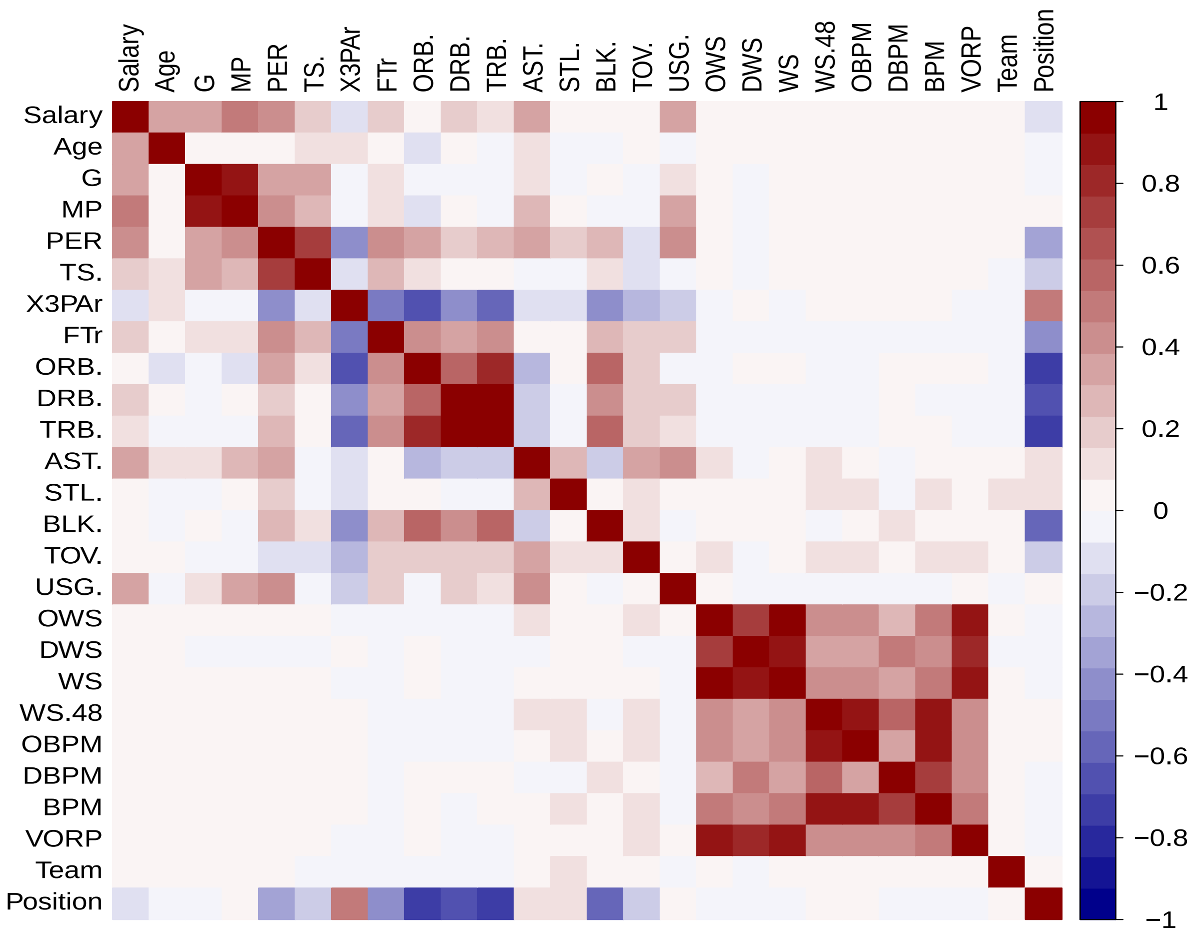

Figure 4.1: Correlation plot of the response and the features in the dataset.

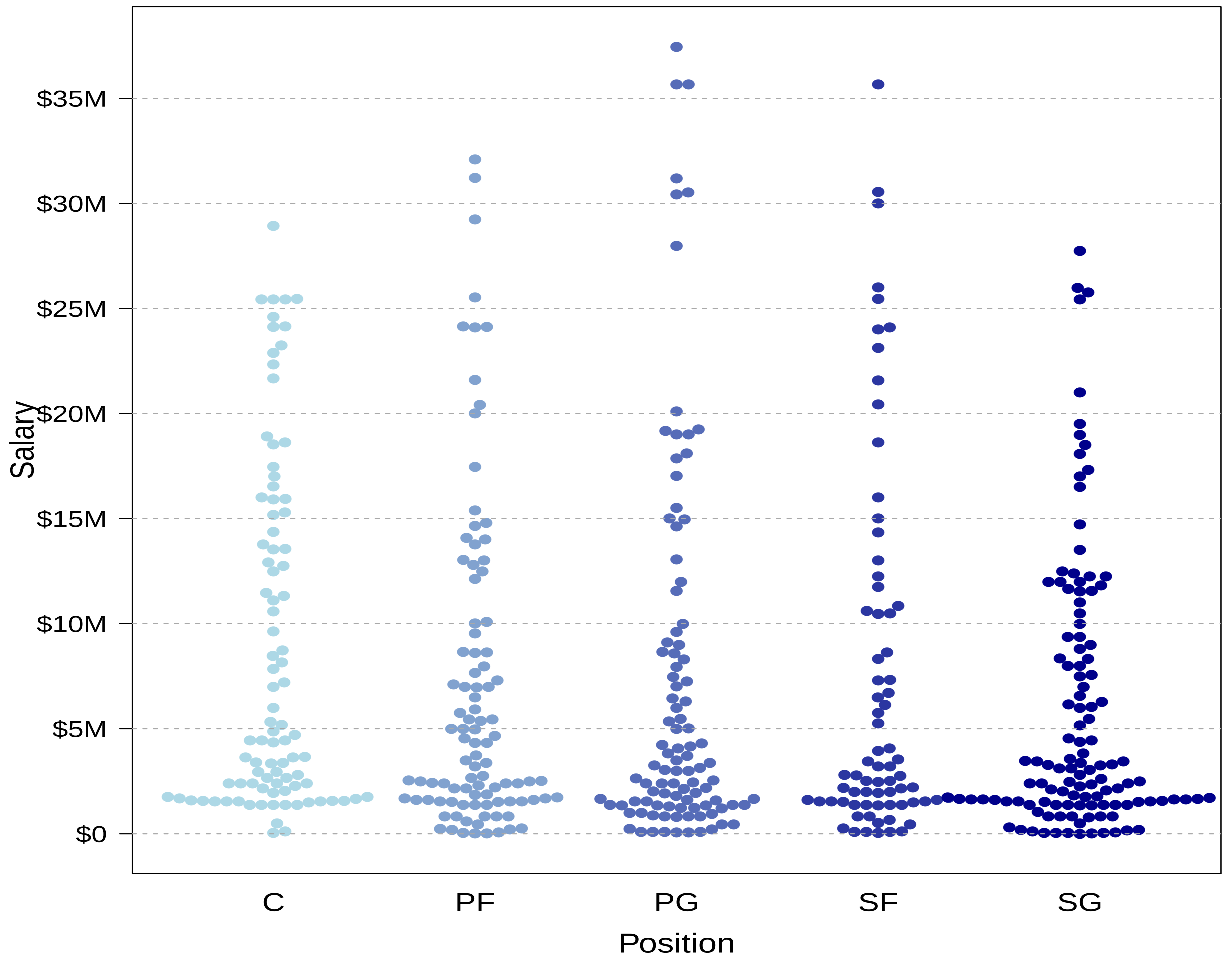

Figure 4.2: Stacked plot of the distribution of salary on five main positions.

As we described, **Position** has 11 levels. To visualize it more clearly, we first reduce the number of levels to 5. The way we do it is to select the former one of the combined position. For example, for a player with combined position "PF-C", we choose the former one "PF" as his position data. Then there are 5 main positions left and the stacked plot (Figure 4.2) is quite clear. This figure shows that the five positions have similar salary distributions. Then we calculate the average salary of each position (Table 4.3), which indicates that position C has the highest average salary while SG has the lowest and the other three are on the same level. Based on it, we create two new dummy variable (**Pos.C**, **Pos.SG**) to replace **Position** variable, where **Pos.C** take value "1" if the position of the player is "C", otherwise 0 and **Pos.SG** set "1" for "SG", otherwise 0. If both of them are 0, it means the position is one of the other three positions: "PF", "SF", "PG".

Position	C	PF	SF	SG	PG
Number of Player	91	97	73	120	99
Average Salary ($)	8,809,944	6,942,628	7,418,001	5,907,966	7,200,504

Table 4.3: Number of Player and average salary of Each Position

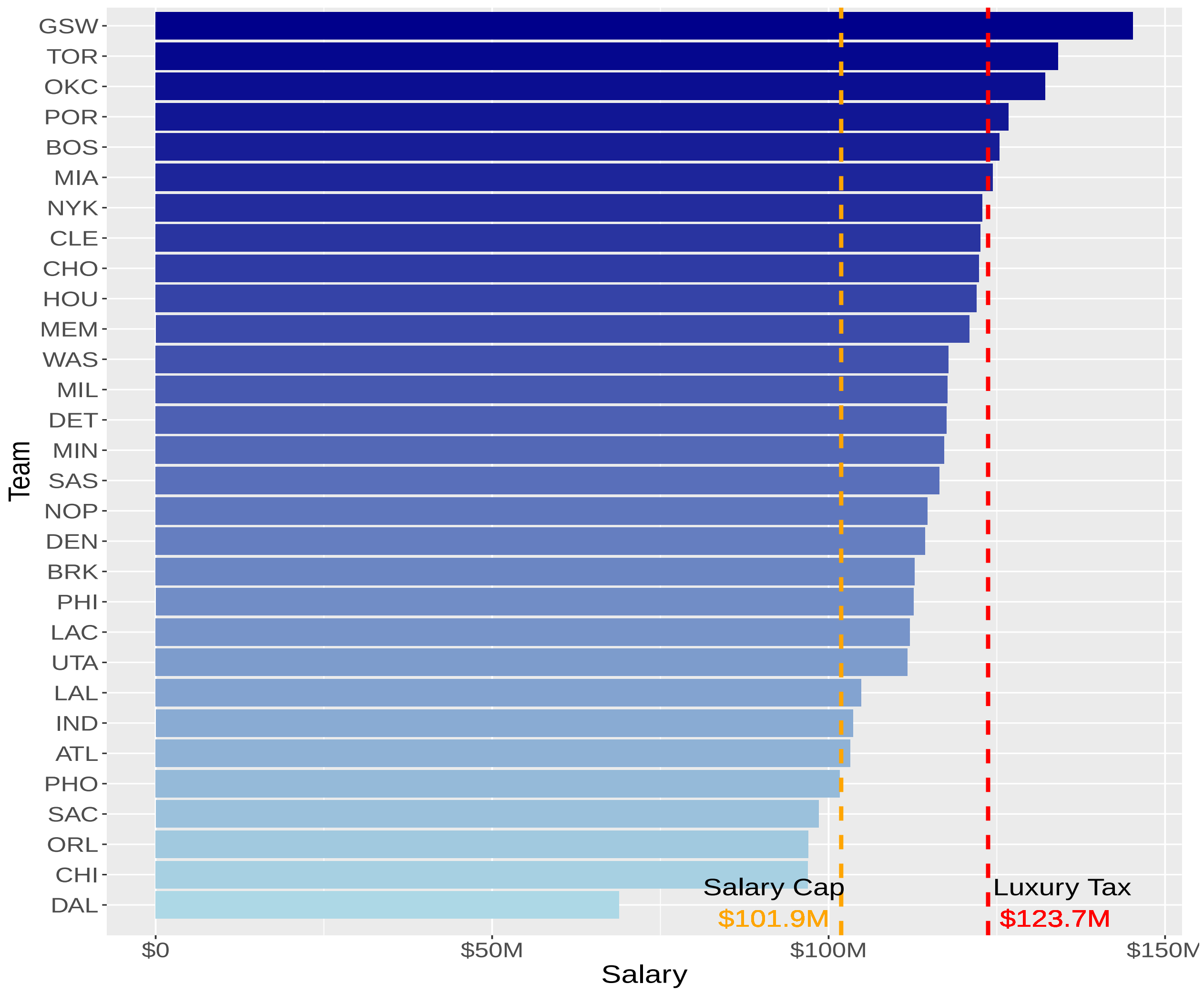

Figure 4.3: Bar plot of the total salary of each team. Orange and red lines represents the "Salary Cap" and "Luxury Tax" of NBA 2018/19 season, respectively.

In Figure 4.3, each team's total salary of all players is shown clearly, from which we can see that team GSW (Golden State Warriors) has the highest total salary significantly, the second is the TOR (Toronto Raptors) and the last is DAL (Dallas). For the performance of the team, DAL was at the end of all the teams not surprisingly. But interestingly, Raptors win the NBA 2018/19 Championship and Warriors lost to Raptors in the Finals. From this story, we can notice that there exists some relationship between the salary and team's success but salary cannot decide the results and sometimes amazing happens.

The orange and red lines indicates the "Salary Cap" ($101.9M) and "Luxury Tax" ($123.7M) of 2018/19 season respectively. NBA league controls every team's total salary in order to avoid some team gathering star players by setting these two lines. If some team's total salary is above the "Salary Cap", it will be reduced privileges of signing players while if "Luxury Tax" is exceeded, such team has to pay a huge penalty to the league. Here, we use the "Salary Cap" to construct a new dummy variables to replace variable **Team**. In detail, if a player is in some team whose total salary is above "Salary Cap" that means the team data of this player is more relevant to higher total salary, we put value "1", otherwise "0".

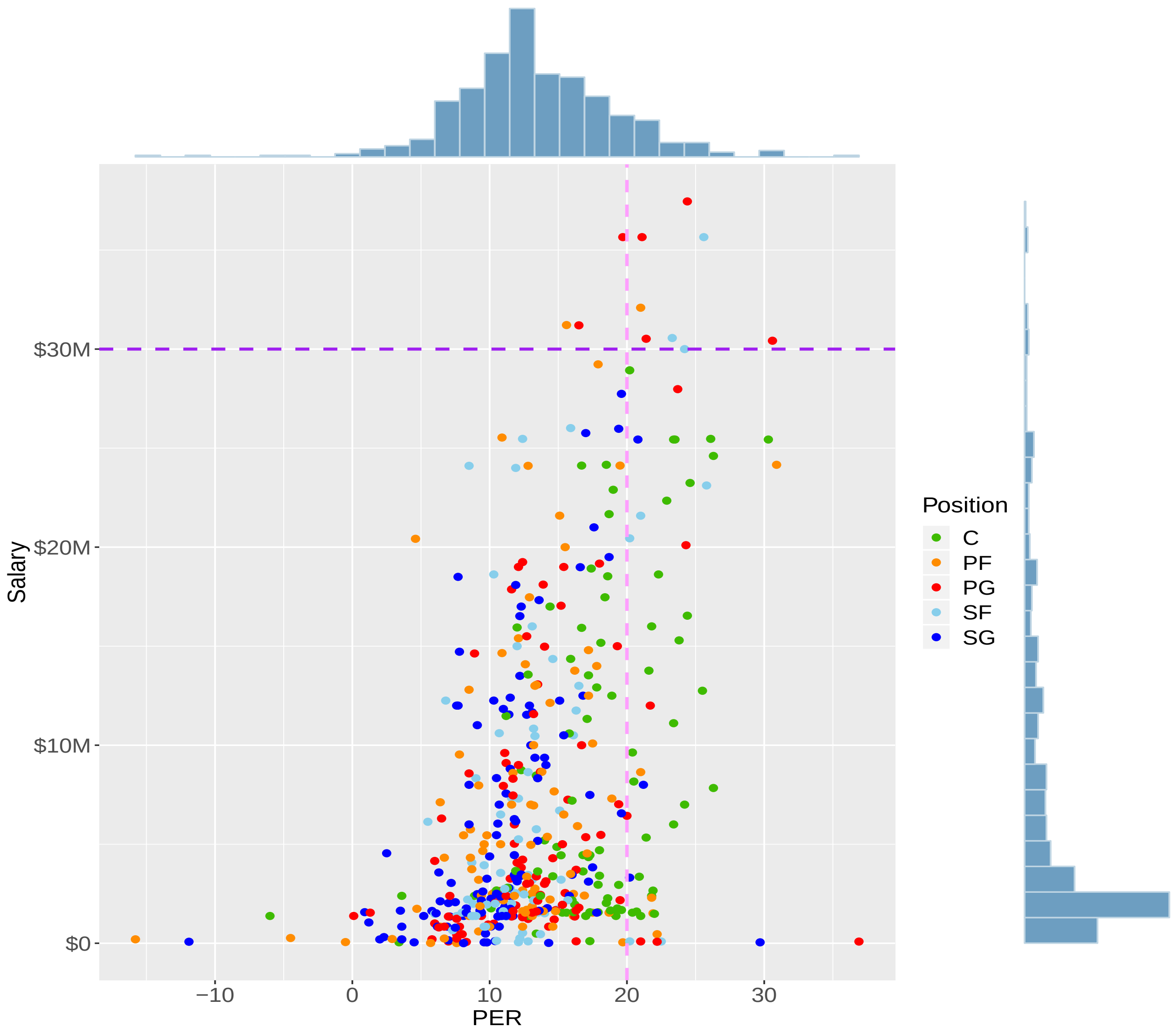

Figure 4.4: Scatter plot of salary against PER for each player.

As described above, **PER** is a crucial index to measure players' performance. Here, we use scatter plot to visualize the pattern between salary and PER. Form Figure 4.4, the players with top salary usually have high PER (> 20) but it is also noticeable that some players have high PER but middle-level or low salary. It is possible since on the one hand, PER is not only relevant to salary but also positions, e.g. the plot shows that most "C" (green dots) has higher PER than SG (blue dots). On the other hand, some players may made great progress in this year and their salary is low currently but may be raised in the future. Overall, salary has connection with PER.

In Figure 4.5, the relationship between salary and "Minutes Played" is presented, from which we can notice that for most players who has less minutes on the filed, it is more probable for them to get less paid. In addition, a few players have fewer MP but high salary, possibly because these points represent star players who got injured in the season. It is a common scenario due to the fact that star player are always heavily defended and have a higher likelihood of injury.

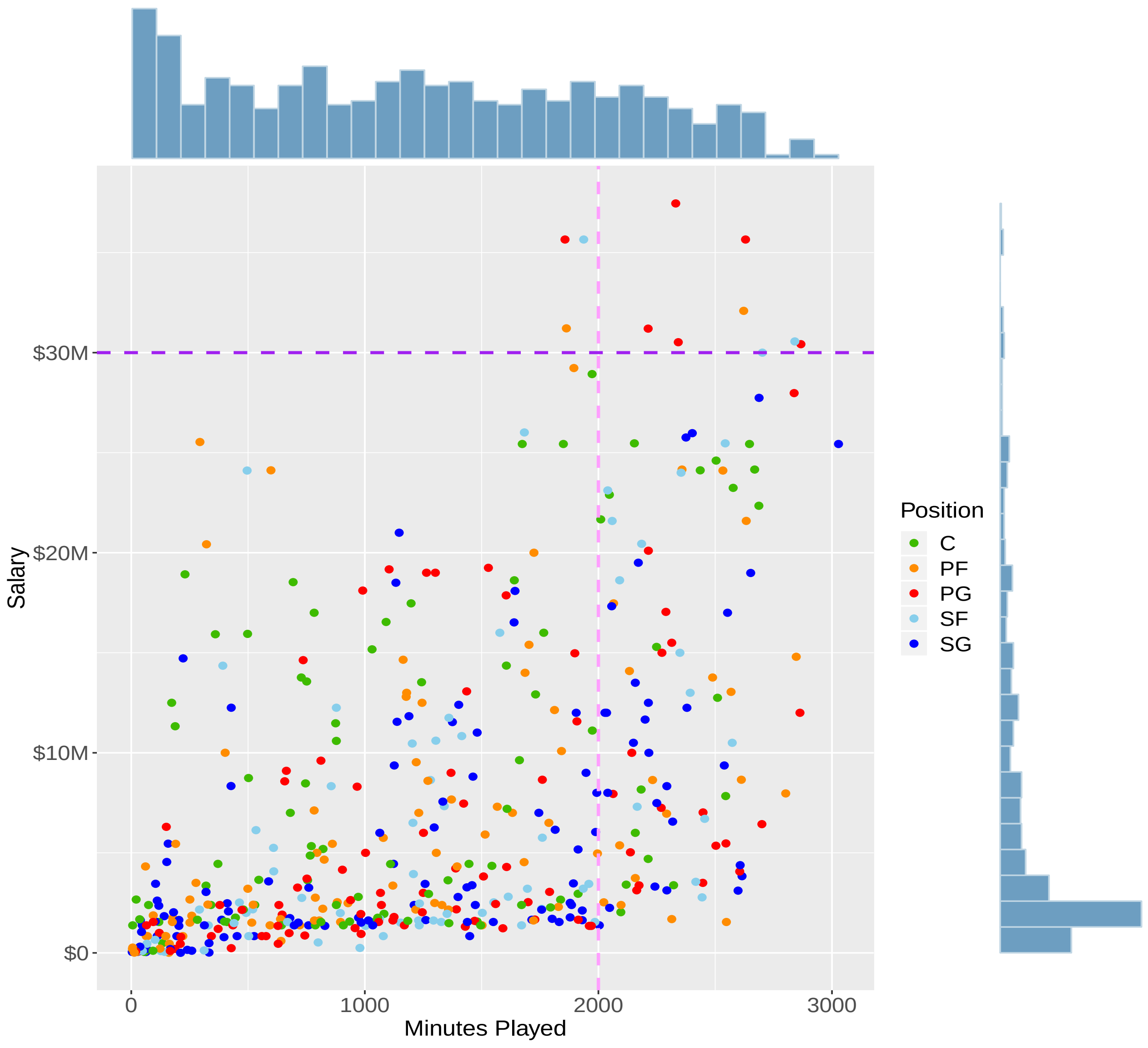

Figure 4.5: Scatter plot of salary against Minutes Played for each player.

Brief Summary

After data exploratory analysis, We exclude the variables: **X3PAr**, **TRB**, **WS**, **BPM** and **VORP** to avoid multicollinearity. And what we left is a data set containing 480 observations and 21 variables with variables **Team** and **Position** being replaced by three dummy variables: **Team.d**, **Pos.C** and **Pos.SG**. Furthermore, we have seen that there exist respective relationships between salary and PER, "Minutes Player" of each player, which may be reflected in the process of model fitting. Next, we would use this reconstructed dataset to do performance comparison and analysis of different competitive methods.

4.3 Competitors

To measure the performance of the proposed approach, we compare it to 4 widely-used regression methods: best subset selection, lasso regression, ridge regression, elastic net. We also apply our approach in the case where $f := LPTN$, which allows to evaluate the presence of outliers. All methods are described in detail below.

In the analysis, we restrict our attention on MAP estimate to simplify and the posterior densities are given below. We tune the parameter λ using cross-validation as done for ridge, lasso and elastic net. For best subset selection, we select the number of variables through the same cross-validation procedure. The statistical analysis is thus in a sense frequentist.

4.3.1 Best Subset Selection

Best subset selection is a regression method that compares all the potential models with different number of variables. The optimization problem is as follows:

$$\min_{\beta}\{\frac{1}{n}\left\|\mathbf{y}-\mathbf{X}\boldsymbol{\beta}\right\|_2^2+\lambda\left\|\boldsymbol{\beta}\right\|_0\} \tag{4.1}$$

where $\lambda > 0$ is a tuning parameter and $\left\|\boldsymbol{\beta}\right\|_0$ is the l_0 norm of $\boldsymbol{\beta}$, representing the number of non-zero entries of the vector $\boldsymbol{\beta}$.

In our case, there are 21 variables so this approach compares models with 1,2,...,21 variables and choose the best one. Here, we use the `regsubsets` function in R to choose the best number of variables, which is equivalent to tuning λ.

4.3.2 Ridge Regression

Ridge regression is a well-known approach that shrinks the coefficients of the correlated variables towards each other, so is a useful to deal with dataset with correlated predictors. It performs a regularization using the l_2 norm penalty.

$$\min_{\beta}\left\{\frac{1}{n}\left\|\mathbf{y}-\mathbf{X}\boldsymbol{\beta}\right\|_2^2+\lambda\left\|\boldsymbol{\beta}\right\|_2^2\right\} \tag{4.2}$$

where $\lambda > 0$ is a tuning parameter and $\left\|\boldsymbol{\beta}\right\|_2$ is the l_2 norm of $\boldsymbol{\beta}$.

The posterior density of ridge we use in the procedure of MAP estimation is:

$$\pi(\boldsymbol{\beta},\sigma|\mathbf{y}) \propto \left(\prod_{j=1}^{p}\frac{1}{\sqrt{2\pi}\sigma_1}\exp\left(-\frac{\beta_j^2}{2\sigma_1^2}\right)\right)\frac{1}{\sigma}\prod_{i=1}^{n}\frac{1}{\sqrt{2\pi}\sigma}\exp\left(-\frac{(y_i-\mathbf{x}_i^T\boldsymbol{\beta})^2}{2\sigma^2}\right) \tag{4.3}$$

where $\sigma_1 = \frac{\sigma}{\sqrt{\lambda n}}$.

As mentioned above, lasso has Bayesian interpretation with Laplace prior. Based on this form, ridge can also be regarded as maximizing a posterior density with a Gaussian prior.

4.3.3 Lasso Regression

Lasso has been mentioned above but it is still worth mentioning the way we choose the λ sequence for lasso. `glmnet` is a quite useful package in R that performs Lasso regression, which uses the coordinate descent algorithm to perform this process (details in Friedman et al. (2010)). In that paper, there is also a method to choose the λ sequence for lasso, which firstly compute $\lambda_{max} = \max_{l} |\langle \boldsymbol{x}_l, \boldsymbol{y} \rangle|/n$, and select a minimum value by setting $\lambda_{min} = \epsilon \lambda_{max}$, then construct the λ sequence from λ_{max} to $\lambda_{min}$with K values. Here, we set $\epsilon = 0.001$, K = 100.

4.3.4 Elastic Net

Elastic Net (Zou and Hastie (2005)) is a combination of lasso regression and ridge regression described above. The objective function is as follows:

$$\min_{\beta}\{\frac{1}{n} \|\mathbf{y} - \mathbf{X}\boldsymbol{\beta}\|_2^2 + \lambda_1 \|\boldsymbol{\beta}\|_1 + \lambda_2 \|\boldsymbol{\beta}\|_2^2\} \tag{4.4}$$

where $\lambda_1 > 0$ and $\lambda_2 > 0$ are tuning parameters.

In Zou and Hastie (2005), the authors also present the following alternative form of the objective:

$$\min_{\beta}\{\frac{1}{n} \|\mathbf{y} - \mathbf{X}\boldsymbol{\beta}\|_2^2 + (1-\alpha)\lambda \|\boldsymbol{\beta}\|_1 + \alpha\lambda \|\boldsymbol{\beta}\|_2^2\} \tag{4.5}$$

where $\alpha = \frac{\lambda_2}{\lambda_1+\lambda_2}$ and λ are the tuning parameters. The term $(1-\alpha) \|\boldsymbol{\beta}\|_1 + \alpha \|\boldsymbol{\beta}\|_2^2$ is called the elastic net penalty, which is a convex combination of the lasso and ridge penalty.

From (4.5), the posterior density we use in the procedure of MAP estimation is:

$$\begin{aligned}\pi(\boldsymbol{\beta}, \sigma|\mathbf{y}) \propto \left(\prod_{j=1}^{p} \frac{1}{\sqrt{2\pi}\sigma_1} \exp\left(-\frac{\beta_j^2}{2\sigma_1^2}\right) \frac{1}{2s_1} \exp\left(-\frac{|\beta_j|}{s_1}\right)\right) \cdot \frac{1}{\sigma} \cdot \\ \prod_{i=1}^{n} \frac{1}{\sqrt{2\pi}\sigma} \exp\left(-\frac{(y_i - \mathbf{x}_i^T\boldsymbol{\beta})^2}{2\sigma^2}\right)\end{aligned} \tag{4.6}$$

where $\sigma_1 = \frac{\sigma}{\sqrt{\alpha\lambda n}}$ and $s_1 = \frac{2\sigma^2}{(1-\alpha)\lambda n}$.

The way we tune the hyper-parameter α and λ is based on the form (4.5) by setting $\alpha = 0.1, 0.2, ..., 0.9$ while λ sequence is the same as that in lasso described above.

4.4 Cross Validation and Measure of Comparison

In this study, we carry out nested cross validation to tune the best hyper-parameters and check the performance of each method. More precisely, we use a 5-fold outer loop and in each outer loop, we divide the dataset to training set and test set. And on each training set we do the 10-fold inner cross-validation to find the best hyper-parameters and refit on the training set to obtain the best model, and then measure its performance on the test set. Doing so we will get the performance of the best model for each of the 5 test sets, so finally we average the performance of them and use it as the final performance for one method. The whole process is shown in **Algorithm 1**.

Here, we do not use the convenient `glmnet` package to fit the model for Lasso, Ridge and Elastic Net. The reason is that we cannot apply it on our LPTL model. Thus, to ensure the fitting processes of all the models are the same, i.e. the difference is only from the model, the MAP estimation of each model are conducted through optimization procedures, i.e. using `optimx` function based on Broyden–Fletcher–Goldfarb–Shanno (BFGS) algorithm. Usually, there are three optimisers being widely used when we use `optim` function: Nelder-Mead, CG and BFGS, where CG represents the conjugate gradient algorithm. The reason that we choose BFGS is shown in Section 4.5.2.

Algorithm 1: Nested Cross Validation for Checking the Performance

```
Input: NBA Data: $D$, Number of folds of outer, inner loops: $N_{outer}$, $N_{inner}$, Sequence
       of hyper-parameters: $\lambda_{seq}$
Output: mean of MSE for one method

1  for i = 1,..., $N_{outer}$ do
2      Separate $D$ to $D_{train,i}$ and $D_{test,i}$
3      for j = 1,..., $N_{inner}$ do
4          Separate $D_{train,i}$ to $\widetilde{D}_{train,j,i}$ and $\widetilde{D}_{valid,j,i}$
5          Fit models on $\widetilde{D}_{train,j}$ using $\lambda_{seq}$, then we obtain K models (K is the number of
             $\lambda$ in the sequence) // Implement Parallel Computing on $\lambda_{seq}$ and the
             optimiser we use is "BFGS".
6          Evaluate each model on the validation set $\widetilde{D}_{valid,j}$ and obtain the $\widetilde{MSE}_{j,k}$ for
             each model, k = 1,...,K.
7      Calculate mean($\widetilde{MSE}_k$) = $\frac{1}{N_{inner}} \sum_{j=1}^{N_{inner}} \widetilde{MSE}_{j,k}$ for K models
8      Choose the $\lambda_k$ with minimum mean($\widetilde{MSE}_k$) as the $\widetilde{\lambda}_{best}$
9      Refit on $D_{train,i}$ to get the best model $M_i$ for $i$ th outer fold
10     Evaluate $M_i$ on $D_{test,i}$ and obtain the $MSE_i$ for current outer fold.
11 i.e. return mean($MSE_i$) = $\frac{1}{N_{outer}} \sum_{i=1}^{N_{outer}} MSE_i$
```

The measure we use to compare models and methods is the prediction Mean-square error (MSE). Specifically, when we measure performance of a model on the test set, the predicted responses $\mathbf{y}_{pred}$ is obtained from applying the model on the predictors and Mean-Square Error of this model is $\frac{1}{n}\sum_{i=1}^{n}(\mathbf{y}_{test,i} - \mathbf{y}_{pred,i})^2$, where $\mathbf{y}_{test}$ is the vector of the response of the test set. Undoubtedly, the lower is MSE, the higher is the accuracy of the predictions made by a model.

There are two advantages of the nested cross validation procedure. Firstly, nested cross validation is much more stable than cross validation because it has 5 MSEs and we use the average of them while cross validation just have one MSE, which is less robust. For instance, it's likely that for one test set, lasso performs better than ridge but in a different test set, the result is opposite. So taking average of 5 MSEs makes the result more reliable. Second, nested cross validation makes full use of data. The whole dataset is divided to 5 folds and each time we use one fold as the test set and the other 4 folds as the training set. Conducting this procedure provides us 5 performance results just based on one dataset. Therefore, the dataset is fully utilized in this study.

In addition to the performance of each method, we also obtain the final model, estimating the parameter on the entire data set. The best model is obtained from implementing cross validation on the whole dataset and the process is similar to the inner cross validation described in nested cross validation with training data replaced by the whole dataset, as shown in **Algorithm 2** below. Note that N_{fold} of the **Algorithm 2** should be same as N_{inner} of the **Algorithm 1**. Doing so we can acquire the best model for each method and its estimated parameters as well as the best hyper-parameters.

Algorithm 2: Cross Validation for Obtaining the Final Model

Input: NBA Data: D, Number of folds: N_{folds}, Sequence of hyper-parameters: λ_{seq}
Output: The best model with estimated parameters

```
1 for j = 1,..., $N_{folds}$ do
2     Separate $D$ to $D_{train,j}$ and $D_{test,j}$
3     Fit models on $D_{train,j}$ using $\lambda_{seq}$, then we obtain K models (K is the number of $\lambda$ in
          the sequence) // Implement Parallel Computing on $\lambda_{seq}$ and the optimiser we use
          is "BFGS".
4     Evaluate each model on the test set $D_{test,j}$ and obtain the $MSE_{j,k}$ for each model,
        where k = 1,...,K.
5 Calculate mean($MSE_k$) = $\frac{1}{N_{fold}}\sum_{j=1}^{N_{fold}} MSE_{j,k}$ for K models
6 Choose the $\lambda_k$ with minimum mean($MSE_k$) as the $\lambda_{best}$
7 Refit on $D$ to obtain the best model $M$
8 return $M$ and its estimated parameters.
```

4.5 Optimization of Algorithm

Regarding the computational aspect of the two algorithms, a large amount of computation exists. There are two ways of speeding up the computation: 1. Parallel computing for evaluating the different values for λ; 2. Providing the gradients to the numerical optimiser.

4.5.1 Parallel Computing

Parallel computing is defined as the execution of several calculations simultaneously, which is one of the most common methods to speed up computations, especially when we cannot define new algorithms with a better computational complexity. In our case, the time complexity is $O(N_{outer}N_{inner}K)$ for **Algorithm 1** and $O(N_{folds}K)$ for **Algorithm 2**, where K is length of the λ sequence we determined. Undoubtedly, K is quite large such as 100 in lasso that produces a lot of computations.

Thus, we consider implementing parallel computing on λ sequence for both algorithms. In detail, we create 5 clusters where each cluster carries out model fitting for 20 λs (say 100 λs in total). And at the beginning of the algorithm, we export dataset, initial parameters, functions and packages that we need to use to the 5 clusters together in order to reduce a lot of time of data transmission, i.e. overhead. Hence, through parallel computing, the algorithm is accelerated a lot.

4.5.2 `Optimx` with Gradient

As mentioned above, we use `optimx` package with BFGS method to obtain the estimates through MAP. Actually, `optimx` is much faster than the well-known `optim` function. However, it needs to calculate the gradient of the objective function. For each method, we have shown its posterior density. For convenience, we take "log" of those densities and and calculate the gradients with respect to $\boldsymbol{\beta}$ for the competitors and our approach. With the gradients, `optimx` speeds up the whole procedure significantly. In addition to the benefit of speeding up the computation, `optimx` also set tolerance (10^{-12}) to make the estimation stable.

There are two reasons for us to choose BFGS algorithm instead of other two optimisers. Firstly, when we check the estimates of lasso obtained through Nelder–Mead, the results do not show the shrinkage property significantly. We infer that it is because that Nelder–Mead is a heuristic search method and the estimates will converges to the true $\boldsymbol{\beta}$ iteratively but there are more than 20 parameters in this study so it may need thousands of iterations to realize the convergence. As for CG, it does not need to calculate the gradient and the result of estimates of lasso is much better than Nelder–Mead. But since we use `optimx` with gradient, choosing BFGS will achieve both higher accuracy and speed than CG.

4.6 Analysis of Results

In this chapter, we analyse, illustrate and compare the results produced by our approach. The results of our approach depend on the choice of ρ. In Section 4.6.1, we present the results associated to the optimal value with $\rho = 0.97$. In Section 4.6.2, we show that they are not so sensitive to a slight change of value when the parameter in a range around 0.95, as it is the case for the parameter of the LPTN in robustness against outliers.

4.6.1 Performance Analysis

According to the procedures presented in Section 4.4, we obtain the performance based on MSE of each method on each fold and their average performance in Table 4.4. The table shows that our models (LPTL, LPTL+LPTN) represent a significant improvement comparing with others. Subset regression performs worst while Ridge, Lasso and Elastic Net are essentially at the same level. For some folds, there may exist exceptions. For instance, Lasso/Elastic Net are better than Ridge for Fold 1 but for Fold 5, it turns out to be the opposite, which embodies the superiority of nested cross validation that produces multiple performance results and ensures our results are stable and reliable. Figure 4.6 provides a visual representation of the differences between the our approach and the other methods.

Model	**Fold 1**	**Fold 2**	**Fold 3**	**Fold 4**	**Fold 5**	**Average**
Subset Selection	0.525	0.418	0.519	0.571	0.555	**0.518**
Ridge	0.518	0.410	0.515	0.552	0.547	**0.508**
Lasso	0.506	0.405	0.521	0.555	0.560	**0.509**
Elastic Net	0.507	0.404	0.521	0.555	0.561	**0.510**
LPTL	0.462	0.381	0.473	0.527	0.500	**0.469**
LPTL + LPTN	0.465	0.380	0.477	0.527	0.495	**0.469**

Note: $\rho_{LPTL} = 0.97$, $\rho_{LPTN} = 0.95$ *for the LPTL and LPTL + LPTN models*

Table 4.4: Summary of Performance (MSE) of Methods on each fold.

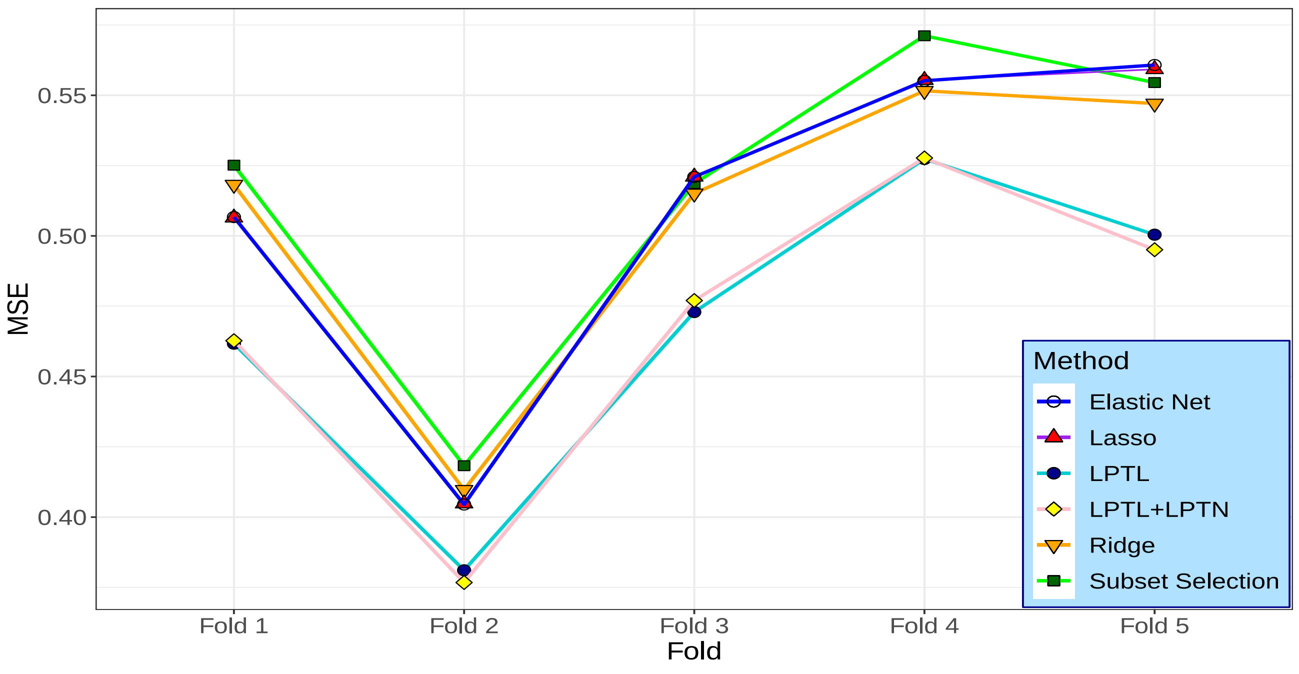

Figure 4.6: Performance (MSE) of methods on each fold.

Variables	Subset Selection	Ridge	Lasso	Elastic Net	LPTL	LPTL + LPTN	OLS
(Intercept)	-	-	-	-	-	-	-
Age	0.336	0.327	0.321	0.325	0.327	0.326	0.327
G	-0.492	-0.473	-0.380	-0.437	-0.486	-0.486	-0.487
MP	0.831	0.784	0.712	0.756	0.796	0.796	0.797
PER	0.301	0.499	0.248	0.387	0.516	0.515	0.516
TS	-0.166	-0.276	-0.111	-0.202	-0.286	-0.287	-0.287
FTr	0.094	0.082	0.075	0.081	0.080	0.081	0.082
ORB	-	-0.130	-0.020	-0.076	-0.134	-0.132	-0.136
DRB	-	0.047	0.060	0.058	0.040	0.040	0.045
AST	-	-0.124	-	-0.058	-0.131	-0.132	-0.133
STL	-	-0.047	-	-0.029	-0.046	-0.046	-0.050
BLK	-	-0.053	-0.012	-0.033	-0.050	-0.048	-0.055
TOV	0.060	0.130	0.040	0.087	0.135	0.135	0.136
USG	-	-0.002	0.051	0.016	-	0.018	-0.007
OWS	-	0.002	-	-	-	-	0.002
DWS	-	0.008	-	0.005	-	-	0.007
WS.48	-	0.017	-	-	-	-	0.018
OBPM	-	-0.037	-0.006	-0.017	-0.030	-0.031	-0.038
DBPM	-	0.019	0.009	0.019	-0.002	-	0.019
Team.d	0.064	0.064	0.054	0.060	0.060	0.059	0.064
Pos.C	-	0.015	-	-	-	-	0.016
Pos.SG	-	-0.008	-	-	-	-	-0.009

Note: "-" represents 0 (to three decimal places)

Table 4.5: Summary of Best Estimates $\widehat{\boldsymbol{\beta}}$ for each Method

In Table 4.5, we list the estimates $\widehat{\boldsymbol{\beta}}$ for the best model of each method, which are the parameters we are mainly focus on. In the last column, we add the OLS estimates as a benchmark. As expected, the models with the super heavy-tailed priors resolve conflicts which translate into estimates closer to $\boldsymbol{\beta}^{OLS}$ when it is significantly different from 0. At the same time, they perform variable selection. These characteristics lead to more accurate predictions. From the table, we could see most estimates of LPTL and LPTL+LPTN models shows this phenomenon, which indicates our models performs quite well and are consistent with our expectation.

For other methods, the best model of Subset Selection method selects 8 variables and six estimates are close to $\widehat{\beta}_j^{OLS}$, which seems not good enough. Ridge does not perform variable selection, which is verified by the estimates we obtained in the table while nearly all of the Lasso estimates are smaller than $\widehat{\beta}_j^{OLS}$, which conforms to its shrinkage property. Furthermore, Elastic Net is indeed a combination of Lasso and Ridge, both from the size and the number of the estimates.

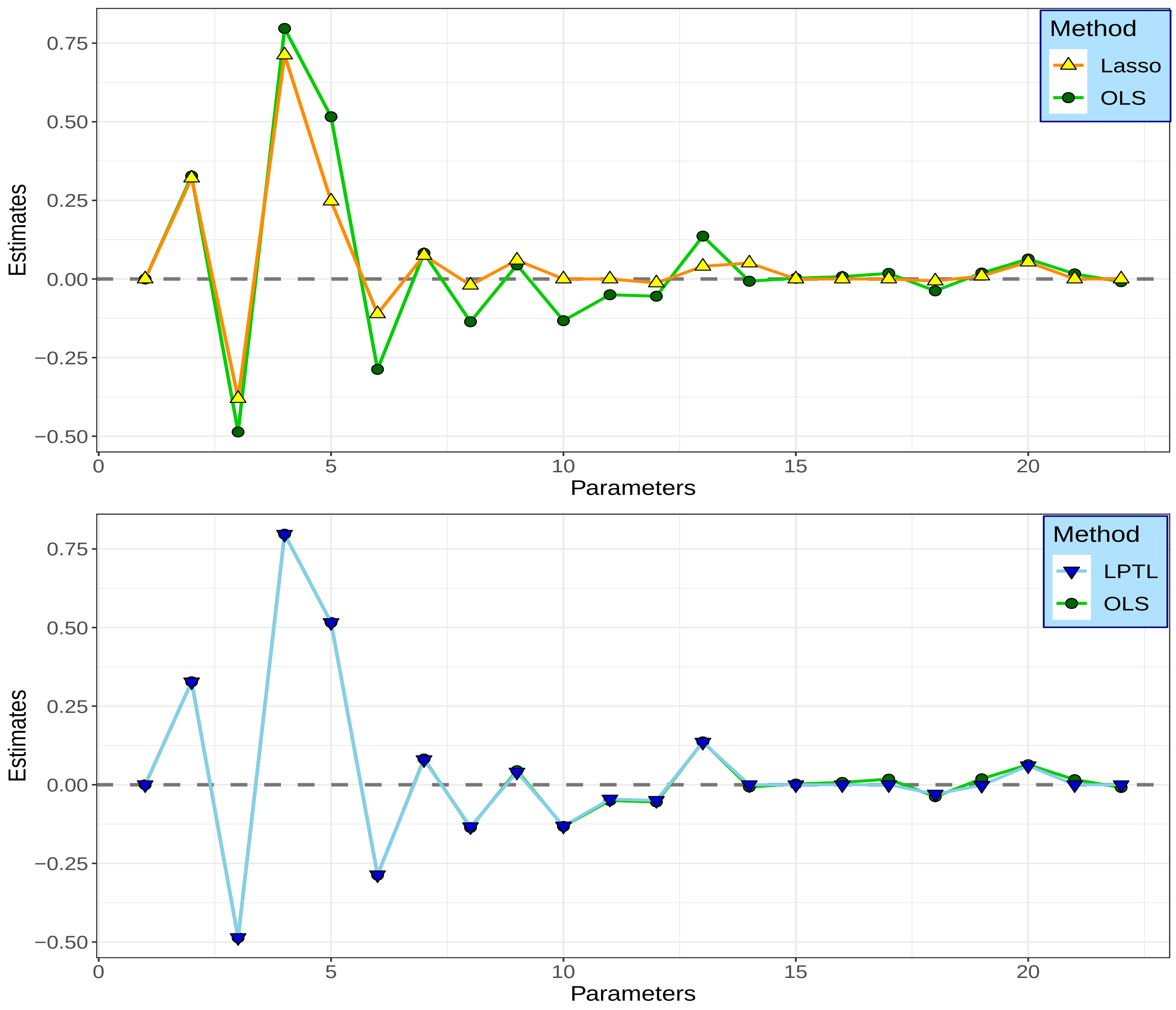

Figure 4.7: **Top:** Plot of comparison between the estimates of best lasso and OLS; **Bottom:** Plot of comparison between the estimates of best LPTL model and OLS.

If we come back to our main competitor, lasso, we visually represent in Figure 4.7 the differences between the estimates of it and LPTL by comparing them with OLS estimate separately. From the top plot of Figure 4.7, it is obvious that the the range of the "track" (orange line) of Lasso estimates is within the OLS estimates. However, the bottom plot shows that the line of estimates of LPTL (blue line) nearly perfectly covers[1] the line of OLS estimates (green line) when the estimates are significantly different from 0. Nevertheless, when OLS estimates close to 0, the LPTL estimates always lie on the x-axis (triangle label), i.e. equals to 0 exactly. Therefore, we have seen the big difference of the estimates "track" of Lasso and LPTL based on OLS estimates, which achieves our expectation in Section 3.3 that for significant parameters, priors behaving as $\frac{1}{|\beta_j|}$ so that the penalisation on $\boldsymbol{\beta}$ is much weaker than that of Laplace prior and the estimates are close to $\widehat{\beta}_j^{OLS}$, while the non-significant parameters with priors behaving like the usual Laplace are shrunk to 0 in this case.

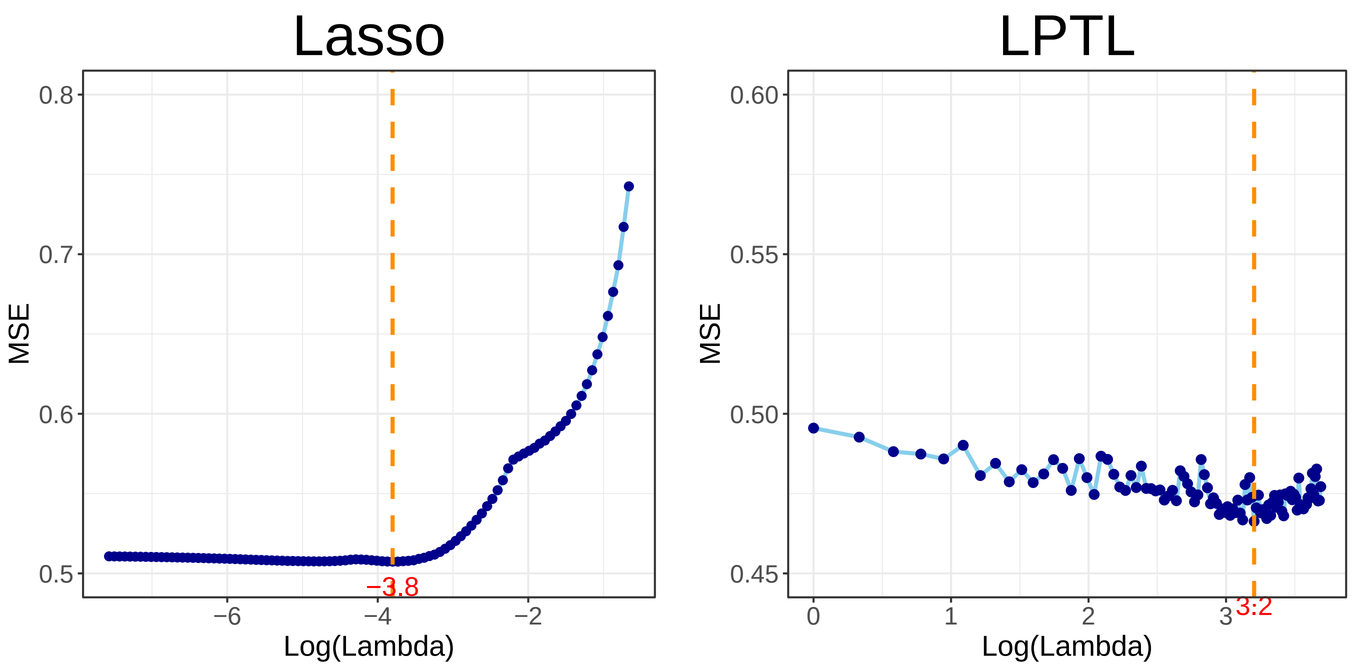

Figure 4.8: **Left:** MSE of lasso model against log(λ), fitting on the whole dataset; **Right:** MSE of LPTL model against log(λ).

In addition to the great performance of LPTL model, the good property it also possesses is the insensitivity against λ. We present the plots (Figure 4.8) of MSE against $\log(\lambda)$ for lasso models and LPTL models to illustrate it. Firstly, we can see that the best lasso model takes $\log(\lambda) = -3.8$ while the best LPTL model takes $\log(\lambda) = 3.1$, indicating a huge difference. Second, looking at the variation of MSE, it is noticeable that MSE of lasso model varies dramatically as λ increases and is beyond 0.70 at the end. However, the MSE of LPTL model always stays within a small range between 0.45 and 0.50. The difference shows that LPTL model is insensitive to the λ, which may be due to that in out approach, we are allowed to penalise more because we only penalise the non-significant coefficients.

In my view, it is meaningful in practice. When people use lasso in practical regression problem, the main objective usually is to obtain a good-fitting model with selected variables. But there exists a trade-off between the good prediction and variable selection. In other

[1]Since we use `ggplot2` that has the characteristic of "layers overlapping" and we plot LPTL+LPTN estimates line after OLS estimates line, the green line is covered by blue line.

words, when fewer variables are expected, the hyper-parameter λ will be tuned larger, then normally, the prediction performance will be worse as shown in the left plot of Figure 4.8. But in our approach, it's not an issue because of its insensitivity property of λ and we can obtain a model with better prediction performance with selected variables, which indicates its robustness from another angle.

4.6.2 Tuning ρ for LPTL and LPTN

We now presented the relatively optimal ρ_{LPTL} and ρ_{LPTN} for our proposed models. Here, the two hyper-parameters are tuned separately, that is we tune ρ_{LPTL} using LPTL model first and then fix it to tune ρ_{LPTN} using LPTL+LPTN model. More precisely, we consider several values for ρ_{LPTL} and ρ_{LPTN} from 0.93 to 0.98 with 0.01 gap. Then we carry out the same nested cross validation procedure described above to calculate the mean of MSE to measure the performance of the model for each ρ.

The results are shown in Figure 4.9, where the left plot gives an optimal $\rho_{LPTL} = 0.97$ and right plot indicates $\rho_{LPTN} = 0.95$ when fixing $\rho_{LPTL} = 0.97$. It is noticeable that for the LPTL plot (Left), the scale of the y-axis is within a small range, which shows that mean(MSEi) is not sensitive when ρ is in a range around 0.95, as it is the LPTN case of robustness against outliers. Therefore, setting it to 0.95 as well thus seems to be a suitable guideline and a appropriate value for avoiding over-fitting in the future study.

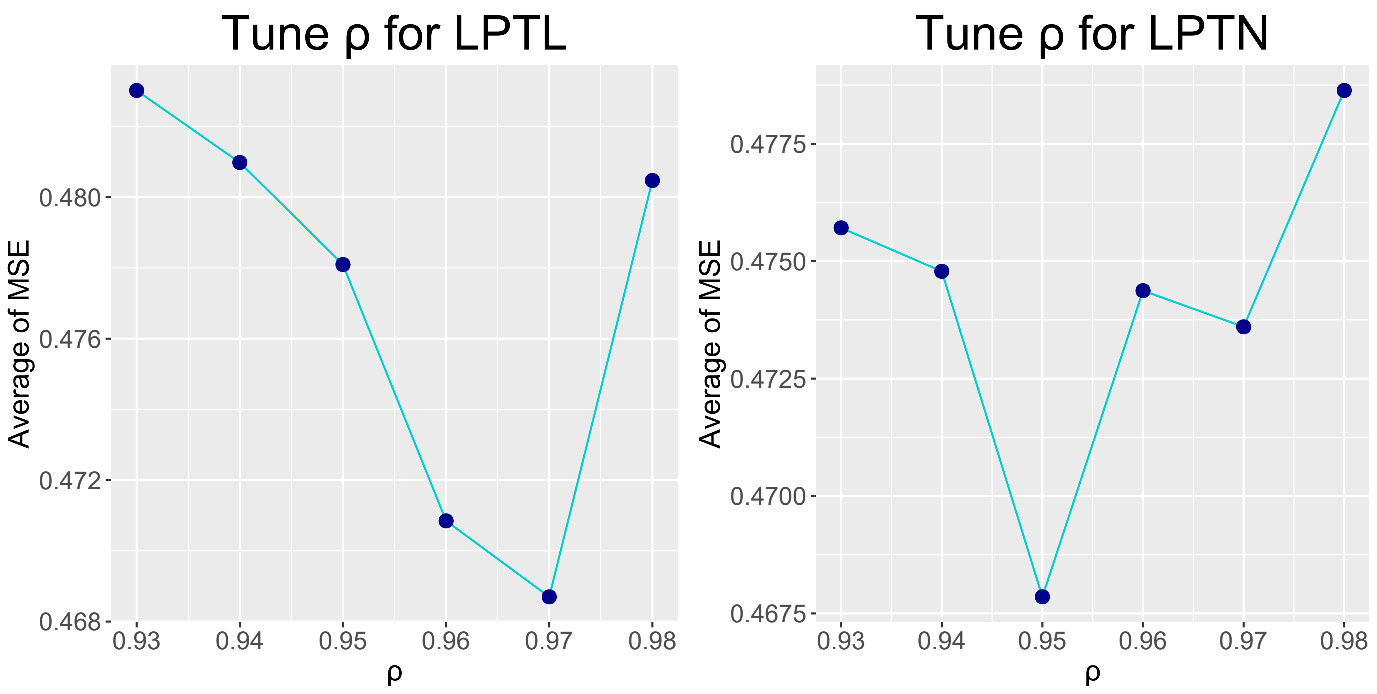

Figure 4.9: **Left:** MSE of LPTL model against ρ; **Right:** MSE of LPTL+LPTN model against ρ when ρ_{LPTL} is fixed at 0.97.

4.6.3 Outlier Detection

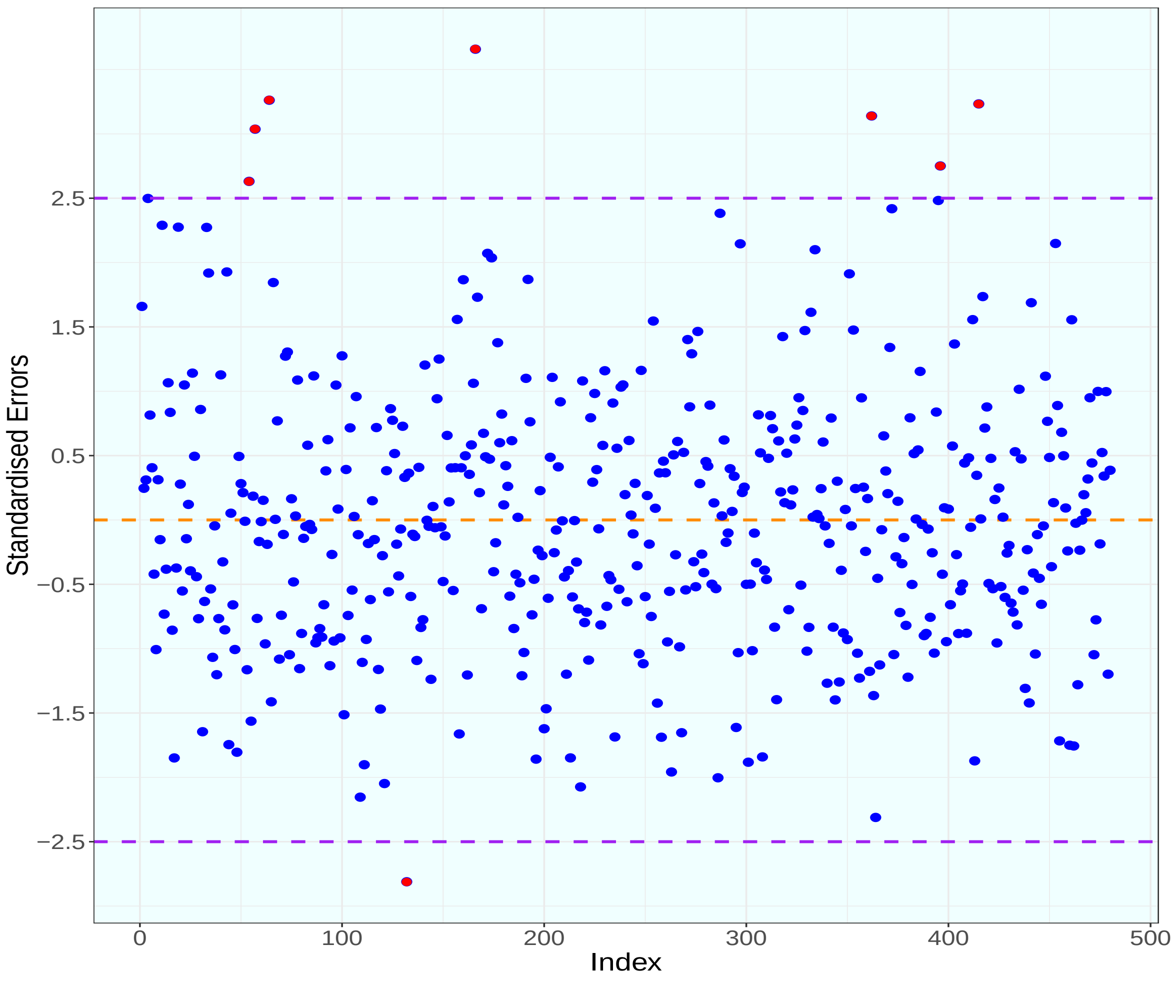

Figure 4.10: Standardised errors against under the LPTL+LPTN model.

From Table 4.4, the performance of LPTL+LPTN is nealy the same as LPTL model, which means there is few outliers in the dataset. As introduced in Gagnon et al. (2018), LPTN model is more effective than normal for outlier detection. Thus, we use LPTL+LPTN model to detect the outliers in the dataset. Since the estimates for $\boldsymbol{\beta}$ and σ have been obtained, we substitute them into the formula for the standardised error $z_i = (y_i - \mathbf{x}_i\boldsymbol{\beta})/\sigma$ to compute the standardised error for each observation. And by Gagnon et al. (2018) and Gervini et al. (2002), 2.5 represents a reasonable value to flag potential outliers.

In Figure 4.10, there are 8 observations in our dataset whose standardised errors are large than 2.5. As mentioned by (Gagnon et al., 2018), with a normal, the probability to observe errors more extreme than -2.5 or 2.5 is 1 $\%$. And in our case, there are around 500 observations, so it is acceptable to have 5 observations with the absolute value of error larger than 2.5. There are 5 observations with error larger than 3: 57, 64, 166, 362, 415. By checking in the dataset, we find all of these five observations have salary larger than $\$24,000,000$ so it is reasonable to have relatively higher errors. Basically, they are not outliers. It is consistent with our results of the LPTL+LPTN performance. Furthermore, the graph indicates that the linear fit is appropriate.

Chapter 5

Conclusion and Discussion

5.1 Summary of Study

In this study, we encounter a problem of conflicting information coming from the prior in linear regression. The purpose of this work is to construct a wholly robust model based on the resolution of conflicting prior information, and then apply it to lasso regression.

We firstly separate the conflicting prior information in linear regression to two cases according to the behaviour of the location and scale parameter of the coefficients, respectively. By replacing the prior distribution by the LRVD, a super heavy-tailed distribution, two resolutions (**Resolution 1**, **Resolution 2**) are presented regarding the two cases of conflicting prior information, leading to different limiting posterior densities. When there exists conflicting information on β_j, the limiting posterior density of **Resolution 1** discards the prior of β_j while the other replaces the prior by $\frac{1}{|\beta_j - \mu_j|}$, both of which indicating a robustness against conflicting prior information. When there is no conflicting prior information, the Laplace prior still works in both cases.

Regarding the conflicting prior information for lasso, we propose a LPTL distribution that belongs to the family of LRVD. Based on it, a resolution of conflicts for lasso is presented, which is a special case of **Resolution 2**, where the prior behaves as $\frac{1}{|\beta_j|}$ in the limit that has much less penalization on β_j comparing with the Laplace prior in lasso. The resolution leads to a wholly robust lasso against conflicting prior information and is expected that in practice, the estimates of this model will behaved as close to OLS estimate with siginificant parameters while with nonsignificant parameters, the estimates are shrunk to 0, which is exactly what our results indicate.

To empirically evaluate the performance of the wholly robust model, we bring in four widely-used regression models: best subset selection, ridge, lasso and elastic net to compete with. The results show that our model succeed in beating other approaches. Also, the estimates generated from the model meet our expectation that the proposed model not only performs variable selection like lasso but also resolves conflicts, which translate into estimates closer to $\widehat{\boldsymbol{\beta}}^{OLS}$ for significant parameters, indicating the validity and superiority of the wholly robust model. Furthermore, adding LPTN distribution to the our approach makes it feasible for outlier detection and the results indicate that the dataset has few outliers.

5.2 Further Work

Further wotk can be conducted based on this study. First of all, even though the proofs of **Resotion 1** given two assumptions are given and the ideas of the assumptions are presented, the rigorous proof of the assumptions of **Resolution 1** as well as a complete proof regarding **Resolution 2** can be further worked on. As mentioned in Chapter 6, the hard part mainly comes from the proof of the finiteness of the marginal in both cases. Based on the idea, if there is a way to use the change of variables on both the scalar β_j and the vector $\boldsymbol{\beta}$ so that we can combine the prior and the likelihood in the integral, it may be figured out.

Second, we visualize the convexity of the objective function of the LPTL model and show it is not convex. The limitation is that we only consider the case of two parameters: β, σ and actually, the objective function has $p + 1$ parameters: $\sigma, \beta_1, \ldots \beta_p$. Also, the rigorous proof of the convexity can be conducted such as checking if the Hessian matrix is symmetric positive semi-definite by using the sylvester criterion, though it is almost certain that the loss function is non-convex. Due to this fact, the computation is a downside.

Focusing on the computational aspects, `optimx` function based on BFGS in our algorithm has shown a great improvement of the computation speed comparing with `optim` function, but other functions and optimisers can be discussed further such as the taylor-made optimiser. Besides, Markov chain Monte Carlo (MCMC) algorithms may also be conducted to obtain the posterior estimates and carry out other analysis using posterior means and medians in this context, which can be further studied to perfect our models.

Chapter 6

Proofs and Ideas

6.1 Proof of Resolution 1 (A)

First of all, we make two assumptions:

(i) $\pi(\boldsymbol{\beta}, \sigma|\mathbf{y})$ and $\pi^k(\boldsymbol{\beta}, \sigma|\mathbf{y})$ are proper given $n > p + 1$.

(ii) We can interchange the limit and integral at step II of the proof.

Then

$$\begin{aligned}
\lim_{\omega\to\infty} \frac{m(\mathbf{y})}{m^k(\mathbf{y}) \prod_{j=1}^{p} [g(\mu_j)]^{l_j}} &\overset{\text{I}}{=} \lim_{\omega\to\infty} \frac{m(\boldsymbol{y}_n)}{m^k(\mathbf{y}) \prod_{j=1}^{p} [g(\mu_j)]^{l_j}} \int_{\mathbb{R}^p}\int_0^\infty \pi(\boldsymbol{\beta}, \sigma|\mathbf{y}) d\sigma d\boldsymbol{\beta} \\
&= \lim_{\omega\to\infty} \int_{\mathbb{R}^p}\int_0^\infty \frac{m(\mathbf{y})}{m^k(\mathbf{y}) \prod_{j=1}^{p} [g(\mu_j)]^{l_j}} \frac{\pi(\boldsymbol{\beta}, \sigma)L(\boldsymbol{\beta}, \sigma; \mathbf{y})}{m(\mathbf{y})} d\sigma d\boldsymbol{\beta} \\
&= \lim_{\omega\to\infty} \int_{\mathbb{R}^p}\int_0^\infty \frac{L(\boldsymbol{\beta}, \sigma; \mathbf{y}) \cdot \prod_{j=1}^{p} [\frac{1}{s_j}\, g(\frac{\beta_j-\mu_j}{s_j})]^{k_j+l_j}}{m^k(\mathbf{y}) \cdot \prod_{j=1}^{p} [g(\mu_j)]^{l_j}} d\sigma d\boldsymbol{\beta} \\
&= \lim_{\omega\to\infty} \int_{\mathbb{R}^p}\int_0^\infty \pi^k(\boldsymbol{\beta}, \sigma|\mathbf{y}) \cdot \prod_{j=1}^{p} \Big[\frac{\frac{1}{s_j}\, g(\frac{\beta_j-\mu_j}{s_j})}{g(\mu_j)}\Big]^{l_j} d\sigma d\boldsymbol{\beta} \\
&\overset{\text{II}}{=} \int_{\mathbb{R}^p}\int_0^\infty \pi^k(\boldsymbol{\beta}, \sigma|\mathbf{y}) \cdot \lim_{\omega\to\infty} \prod_{j=1}^{p} \Big[\frac{\frac{1}{s_j}\, g(\frac{\mu_j-\beta_j}{s_j})}{g(\mu_j)}\Big]^{l_j} d\sigma d\boldsymbol{\beta} \\
&\overset{\text{III}}{=} \int_{\mathbb{R}^p}\int_0^\infty \pi^k(\boldsymbol{\beta}, \sigma|\mathbf{y}) d\sigma d\boldsymbol{\beta} \\
&\overset{\text{IV}}{=} 1
\end{aligned}$$

Step I: By the assumption (i), $\pi(\boldsymbol{\beta}, \sigma|\mathbf{y})$ is proper, i.e. the marginal $m(\mathbf{y})$ is finite, then

$$\int_{\mathbb{R}^p}\int_0^\infty \pi(\boldsymbol{\beta}, \sigma|\mathbf{y}) d\sigma d\boldsymbol{\beta} = \int_{\mathbb{R}^p}\int_0^\infty \frac{\pi(\boldsymbol{\beta}, \sigma)L(\boldsymbol{\beta}, \sigma; \mathbf{y})}{m(\mathbf{y})} d\sigma d\boldsymbol{\beta} = 1$$

Step II: By the assumption (ii), we interchange the limit and integral. Besides, based on the setting in equation 2.7 (Section 2.3.1): $\mu_j = a_j + b_j\omega, \quad j = 1, ..., p$, and b_j = 0 when there is no conflicting prior information, i.e. $k_j = 1, l_j = 0$. Hence, $\pi^k(\boldsymbol{\beta}, \sigma|\boldsymbol{y}_n)$ does not depend on ω and we could put it in front of the limit. Also, since g is symmetric (by condition (i): g LRVD), then we have $g(\frac{\beta_j-\mu_j}{s_j}) = g(\frac{\mu_j-\beta_j}{s_j})$.

Step III: By the condition (i): g is LRVD, then based on the Proposition 2.1 (location-scale invariance), we have

$$\lim_{\omega\to\infty}\prod_{j=1}^{p}\Big[\frac{\frac{1}{s_j}g(\frac{\mu_j-\beta_j}{s_j})}{g(\mu_j)}\Big]^{l_j} = \prod_{j=1}^{p}\Big[\lim_{\omega\to\infty}\frac{\frac{1}{s_j}g(\frac{\mu_j-\beta_j}{s_j})}{g(\mu_j)}\Big]^{l_j} = \prod_{j=1}^{p}\Big[\lim_{\mu_j\to\infty}\frac{\frac{1}{s_j}g(\frac{\mu_j-\beta_j}{s_j})}{g(\mu_j)}\Big]^{l_j} = 1$$

Step IV: By the assumption (i), $\pi^k(\boldsymbol{\beta}, \sigma|\mathbf{y})$ is proper. ■

6.1.1 Idea of Assumption (a)

To prove $\pi(\boldsymbol{\beta}, \sigma|\mathbf{y})$ and $\pi^k(\boldsymbol{\beta}, \sigma|\mathbf{y})$ are proper is equivalent to prove $m(\mathbf{y})$, $m^k(\mathbf{y})$ are finite. Gagnon et al. (2018) has proved that if n > p + 1, in the case where the prior is bounded, i.e. $\pi^b(\boldsymbol{\beta}, \sigma)$ is bounded by $\max(C, C\sigma^{-1})$ (C can be any positive constant), then

$$\int \pi^b(\boldsymbol{\beta}, \sigma)\prod_{i=1}^{n}\frac{1}{\sigma}f(\frac{y_i - \mathbf{x}_i^T\boldsymbol{\beta}}{\sigma})d\sigma d\boldsymbol{\beta} \text{ is finite} \tag{6.1}$$

We draw the inspiration from his work and give the strategy of the proof that focus on the prior in our case. The full proof could be generalised by considering the integral of the prior and the likelihood at the same time. But since adding the likelihood will result in the vector $\boldsymbol{\beta}$ occurring in the integrand, where the β_j also exists, so that the change of variables (see below) seems not straightforward and need to be further studied.

Firstly, something needed to be assumed on the prior as Gagnon did. We did not mention before to simplify but actually, we need to the prior of the scale to be bounded by $\max(B, B\sigma^{-1})$, where B can be any positive constant. Then our prior is:

$$\pi(\boldsymbol{\beta}, \sigma) = \pi(\boldsymbol{\beta}|\sigma)\pi(\sigma) = \left(\prod_{j=1}^{p}\frac{1}{s_j(\sigma)}g(\frac{\beta_j - \mu_j}{s_j(\sigma)})\right)\pi(\sigma)$$

In the integral, g is a LRVD, σ can take any positive value and $s_j(\sigma)$ is a positive function of σ as defined in Section 2.1, so $s_j(\sigma)$ can also take any positive value. To simplify, we consider the prior on $\boldsymbol{\beta}|\sigma$ and $\pi(\sigma)L(\boldsymbol{\beta}, \sigma; \mathbf{y})$ is exactly the same thing as the integrand in (6.1) that has been done. Here, we consider $s_j(\sigma)$ in two cases to show our idea.

Case 1. $\delta N^{-1} \leq s_j(\sigma) < \infty$, i.e. $\frac{1}{s_j(\sigma)}$ is bounded.

where $\delta > 0$ is a constant that can be chosen as small as we can and $N > 0$ is a constant such that

$$|x| \geq |z| \geq N \Rightarrow g(x) \leq g(z) \text{ and } |x|f(x) \leq |z|f(z) \tag{6.2}$$

based on the assumptions of g, which also implies that $g(z)$ is bounded and we define

$$D := \max\{\sup_{z\in\mathbb{R}} f(z),\ \sup_{z\in\mathbb{R}} |z|f(z)\} \tag{6.3}$$

where $D > 0$ is a constant. In this case, the prior is obviously bounded, since

$$\pi(\boldsymbol{\beta}|\sigma) \leq (ND/\delta)^p$$

Case 2. $0 < s_j(\sigma) < \delta N^{-1}$, i.e. $\frac{1}{s_j(\sigma)}$ tends to infinity.

The proof of this case is much more technical that we cannot give the complete proof but the idea will show a feasible way to do it. The thing that we need to be careful in this case is that when β_j is close to μ_j, which is a $\frac{0}{0}$ form of the ratio $\frac{\beta_j-\mu_j}{s_j(\sigma)}$. The integral of our prior with respect to $\boldsymbol{\beta}$ can be written as:

$$\int_{\mathbb{R}} \frac{1}{s_1(\sigma)} g(\frac{\beta_1-\mu_1}{s_1(\sigma)})d\beta_1 \ldots \int_{\mathbb{R}} \frac{1}{s_p(\sigma)} g(\frac{\beta_p-\mu_p}{s_p(\sigma)})d\beta_p \tag{6.4}$$

We focus on jth item and the others are similar. The strategy is to use the change of variable and integrate over β_j, i.e. let $t_j = \frac{\beta_j-\mu_j}{s_j(\sigma)}$, then $\beta_j = t_j \cdot s_j(\sigma) + \mu_j$ and $d\beta_j = s_j(\sigma)dt_j$ so we can cancel out $s_j(\sigma)$. The jth item becomes:

$$\int_{-\infty}^{\infty} g(t_j)dt_j < \infty \tag{6.5}$$

Since g is a bounded PDF.

The hard part is that in our marginal, the prior and the likelihood occur simultaneously in the integrand and $\boldsymbol{\beta}$ in the likelihood is a vector, so it's hard to integrate our prior and the likelihood with respect to β_j and $\boldsymbol{\beta}$ respectively at the same time. Also using the change of variables for β_j and $\boldsymbol{\beta}$ simultaneously is difficult. Maybe there are some tricks to do the integration like this. If we can do so, our prior in both two cases is bounded and combining what (Gagnon et al., 2018) did, the marginal will be finite with respect to $\boldsymbol{\beta}$ and σ.

6.1.2 Idea of Assumption (b)

For the interchangeability of the integral and the limit, We need to show that the integrand of step II is bounded by a function of σ and $\boldsymbol{\beta}$ and it does not depend on ω. The integrand is

$$\pi^k(\boldsymbol{\beta},\sigma|\boldsymbol{y}_n) \cdot \prod_{j=1}^{p} \left[\frac{\frac{1}{s_j} g(\frac{\mu_j-\beta_j}{s_j})}{g(\mu_j)}\right]^{l_j} \tag{6.6}$$

$\pi^k(\boldsymbol{\beta},\sigma|\boldsymbol{y}_n)$ is a function of $\boldsymbol{\beta}$ and σ obviously. And from Section 6.1.1, it is proper so is integrable. In addition, as we discussed in Step II, it does not depend on ω since when $k_j = 1$, the coefficient of ω is 0.

For the second term $\prod_{j=1}^{p} \left[\frac{\frac{1}{s_j} g(\frac{\mu_j-\beta_j}{s_j})}{g(\mu_j)}\right]^{l_j}$, the idea is that because $\prod_{j=1}^{p} g(\mu_j)^{-1}$ is fixed and bounded as shown above, basically, we have nearly the same thing as in Section 6.1.1, which can be figured out.

■

6.2 Proof of Resolution 1 (B)

Rigorously, we consider the set $(\boldsymbol{\beta}, \sigma)$ such that $\pi^k(\boldsymbol{\beta}, \sigma) \neq 0$

$$\begin{aligned}
\lim_{\omega\to\infty} \frac{\pi(\boldsymbol{\beta}, \sigma|\mathbf{y})}{\pi^k(\boldsymbol{\beta}, \sigma|\mathbf{y})} &= \lim_{\omega\to\infty} \frac{m^k(\mathbf{y})}{m(\mathbf{y})} \times \frac{\pi(\boldsymbol{\beta}, \sigma)}{\pi^k(\boldsymbol{\beta}, \sigma)} \times \frac{L(\boldsymbol{\beta}, \sigma; \mathbf{y})}{L(\boldsymbol{\beta}, \sigma; \mathbf{y})} \\
&\overset{\text{I}}{=} \lim_{\omega\to\infty} \frac{m^k(\mathbf{y}) \prod_{j=1}^{p} [g(\mu_j)]^{l_j}}{m(\mathbf{y})} \times \frac{\pi(\boldsymbol{\beta}, \sigma)}{\pi^k(\boldsymbol{\beta}, \sigma) \prod_{j=1}^{p} [g(\mu_j)]^{l_j}} \\
&= \lim_{\omega\to\infty} \frac{m^k(\mathbf{y}) \prod_{j=1}^{p} [g(\mu_j)]^{l_j}}{m(\mathbf{y})} \times \lim_{\omega\to\infty} \frac{\pi(\boldsymbol{\beta}, \sigma)}{\pi^k(\boldsymbol{\beta}, \sigma) \prod_{j=1}^{p} [g(\mu_j)]^{l_j}} \\
&\qquad\qquad\qquad\qquad\qquad \text{By product rule of limit} \\
&= 1 \times \lim_{\omega\to\infty} \frac{\pi(\boldsymbol{\beta}, \sigma)}{\pi^k(\boldsymbol{\beta}, \sigma) \prod_{j=1}^{p} [g(\mu_j)]^{l_j}} \quad \text{Base on Resolution 1 (A)} \\
&= \lim_{\omega\to\infty} \frac{\pi^k(\boldsymbol{\beta}, \sigma) \prod_{j=1}^{p} \left[\frac{1}{s_j} g\left(\frac{\beta_j - \mu_j}{s_j}\right)\right]^{l_j}}{\pi^k(\boldsymbol{\beta}, \sigma) \prod_{j=1}^{p} [g(\mu_j)]^{l_j}} \\
&= \lim_{\omega\to\infty} \prod_{j=1}^{p} \left[\frac{\frac{1}{s_j} g\left(\frac{\beta_j - \mu_j}{s_j}\right)}{g(\mu_j)}\right]^{l_j} \quad \text{Since} \quad \pi^k(\boldsymbol{\beta}, \sigma) \neq 0 \\
&= 1 \quad \text{By Proposition 2.1: Location-scale Invariance property}
\end{aligned}$$

Step I: g is a LRVD, so above 0 and $g(\mu_j) \neq 0$.

Indeed, based on assumption (a), $\pi^k(\boldsymbol{\beta}, \sigma|\mathbf{y})$ is proper and bounded, then

$$\begin{aligned}
\lim_{\omega\to\infty} \left|\pi(\boldsymbol{\beta}, \sigma|\mathbf{y}) - \pi^k(\boldsymbol{\beta}, \sigma|\mathbf{y})\right| &= \lim_{\omega\to\infty} \pi^k(\boldsymbol{\beta}, \sigma|\mathbf{y}) \left|\frac{\pi(\boldsymbol{\beta}, \sigma|\mathbf{y})}{\pi^k(\boldsymbol{\beta}, \sigma|\mathbf{y})} - 1\right| \\
&= \pi^k(\boldsymbol{\beta}, \sigma|\mathbf{y}) \lim_{\omega\to\infty} \left|\frac{\pi(\boldsymbol{\beta}, \sigma|\mathbf{y})}{\pi^k(\boldsymbol{\beta}, \sigma|\mathbf{y})} - 1\right| \\
&= 0
\end{aligned}$$

where $\pi^k(\boldsymbol{\beta}, \sigma|\mathbf{y})$ does not depend on w.

$\blacksquare$

References

Andrade, J. A. A. and O'Hagan, A. (2011). Bayesian robustness modelling of location and scale parameters. *Scandinavian Journal of Statistics*, 38(4):691–711.

Basketball-Reference (2018a). 2018-19 NBA Player Contracts. `https://www.basketball-reference.com/contracts/players.html`. [Online; accessed 19-August-2019].

Basketball-Reference (2018b). 2018-19 NBA Player Stats: Advanced. `https://www.basketball-reference.com/leagues/NBA_2019_advanced.html`. [Online; accessed 19-August-2019].

Box, G. E. and Tiao, G. C. (1968). A bayesian approach to some outlier problems. *Biometrika*, 55(1):119–129.

De Finetti, B. (1961). The bayesian approach to the rejection of outliers. In *Proceedings of the fourth Berkeley Symposium on Probability and Statistics*, volume 1, pages 199–210. University of California Press Berkeley.

Desgagné, A. et al. (2013). Full robustness in bayesian modelling of a scale parameter. *Bayesian Analysis*, 8(1):187–220.

Desgagné, A. et al. (2015). Robustness to outliers in location–scale parameter model using log-regularly varying distributions. *The Annals of Statistics*, 43(4):1568–1595.

Dickey, J. M. (1975). Bayesian alternatives to the f-test and least-squares estimate in the normal linear model. *Studies in Bayesian econometrics and statistics*, pages 515–554.

Friedman, J., Hastie, T., and Tibshirani, R. (2010). Regularization paths for generalized linear models via coordinate descent. *Journal of statistical software*, 33(1):1.

Gagnon, P., Desgagné, A., Bédard, M., et al. (2018). A new bayesian approach to robustness against outliers in linear regression. *Bayesian Analysis*.

Garthwaite, P. H., Dickey, J. M., et al. (1992). Elicitation of prior distributions for variable-selection problems in regression. *The Annals of Statistics*, 20(4):1697–1719.

Gervini, D., Yohai, V. J., et al. (2002). A class of robust and fully efficient regression estimators. *The Annals of Statistics*, 30(2):583–616.

Hampel, F. R. (1971). A general qualitative definition of robustness. *The Annals of Mathematical Statistics*, pages 1887–1896.

Hill, B. M. (1974). On coherence, inadmissibility and inference about many parameters in the theory of least squares. *Studies in Bayesian econometrics and statistics*, pages 555–584.

Huber, P. J. et al. (1973). Robust regression: asymptotics, conjectures and monte carlo. *The Annals of Statistics*, 1(5):799–821.

Karamata, J. (1930). Sur un mode de croissance reguliere des functions, mathematica (cluj), 4 (1930), 38-53. *Karamata384Mathematica (Cluj)*.

Peña, D., Zamar, R., and Yan, G. (2009). Bayesian likelihood robustness in linear models. *Journal of Statistical Planning and Inference*, 139(7):2196–2207.

Raftery, A. E., Madigan, D., and Hoeting, J. A. (1997). Bayesian model averaging for linear regression models. *Journal of the American Statistical Association*, 92(437):179–191.

Rousseeuw, P. and Yohai, V. (1984). Robust regression by means of s-estimators. In *Robust and nonlinear time series analysis*, pages 256–272. Springer.

Rousseeuw, P. J. (1984). Least median of squares regression. *Journal of the American statistical association*, 79(388):871–880.

Rousseeuw, P. J. (1985). Multivariate estimation with high breakdown point. *Mathematical statistics and applications*, 8(283-297):37.

Tibshirani, R. (1996). Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society: Series B (Methodological)*, 58(1):267–288.

Tibshirani, R. (2011). Regression shrinkage and selection via the lasso: a retrospective. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 73(3):273–282.

Tibshirani, R., Saunders, M., Rosset, S., Zhu, J., and Knight, K. (2005). Sparsity and smoothness via the fused lasso. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 67(1):91–108.

Walter, G. and Augustin, T. (2010). Bayesian linear regression—different conjugate models and their (in) sensitivity to prior-data conflict. In *Statistical Modelling and Regression Structures*, pages 59–78. Springer.

West, M. (1984). Outlier models and prior distributions in bayesian linear regression. *Journal of the Royal Statistical Society: Series B (Methodological)*, 46(3):431–439.

Yohai, V. J. et al. (1987). High breakdown-point and high efficiency robust estimates for regression. *The Annals of Statistics*, 15(2):642–656.

Yuan, M. and Lin, Y. (2006). Model selection and estimation in regression with grouped variables. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 68(1):49–67.

Zou, H. (2006). The adaptive lasso and its oracle properties. *Journal of the American statistical association*, 101(476):1418–1429.

Zou, H. and Hastie, T. (2005). Regularization and variable selection via the elastic net. *Journal of the royal statistical society: series B (statistical methodology)*, 67(2):301–320.

Zou, H. and Li, R. (2008). One-step sparse estimates in nonconcave penalized likelihood models. *Annals of statistics*, 36(4):1509.

Appendix A

Additional Figures

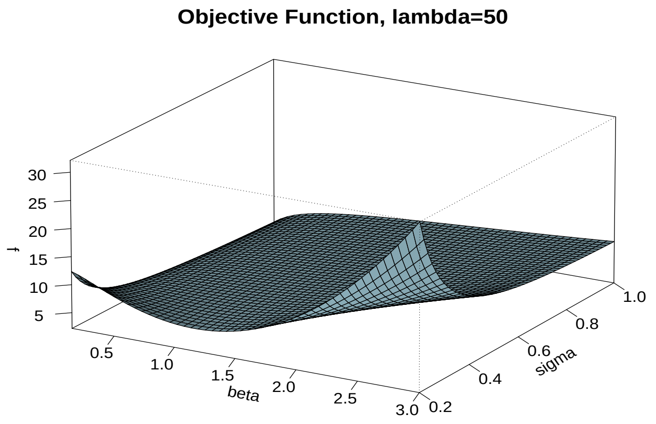

Figure A.1: 3-D plot of objective function with respect to β and σ, $\lambda = 50$

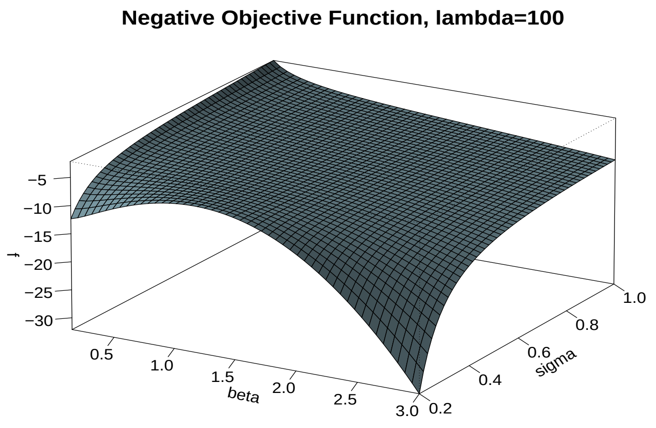

Figure A.2: 3-D plot of negative objective function with respect to β and σ, $\lambda = 100$

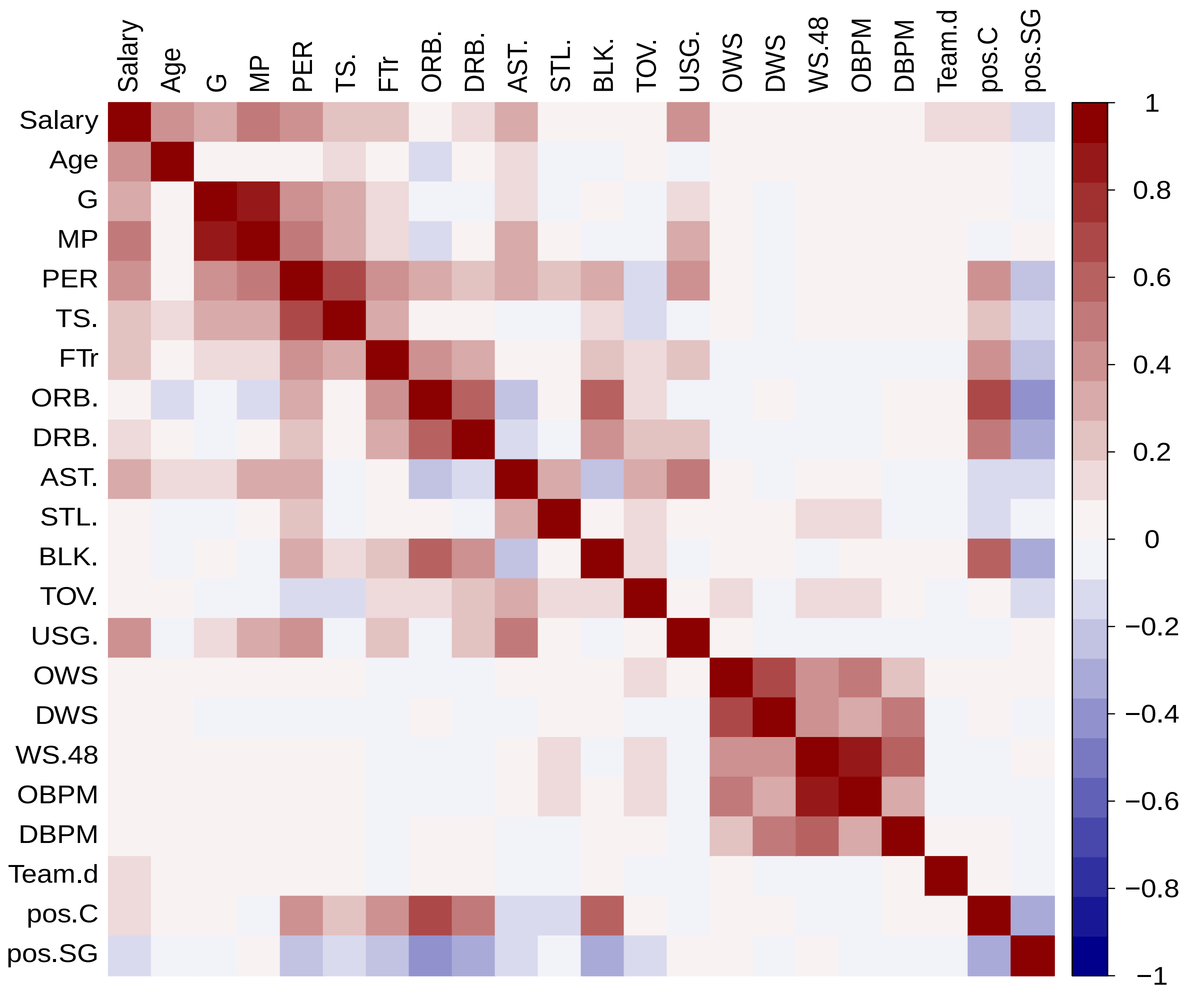

Figure A.3: Correlation plot of the updated dataset

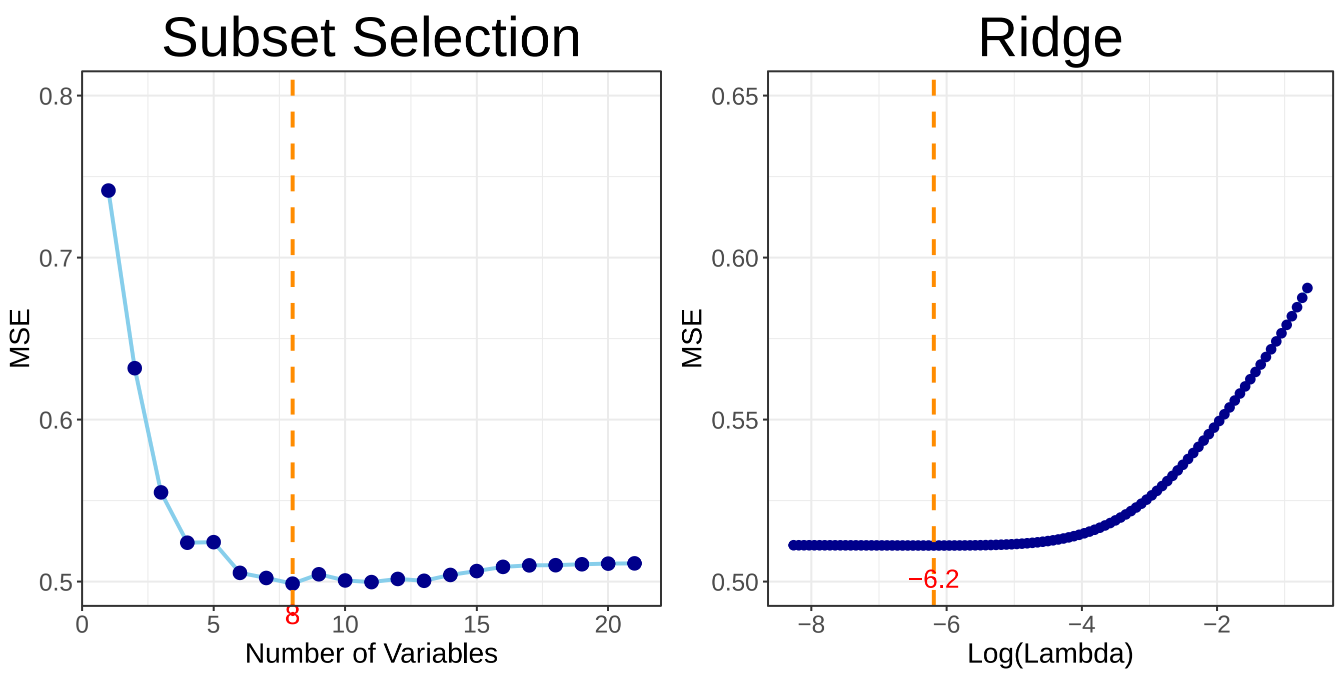

Figure A.4: **Left:** Plot of MSE of Best subset selection against number of variables; **Right:** Plot of MSE of Best subset selection against log(λ).

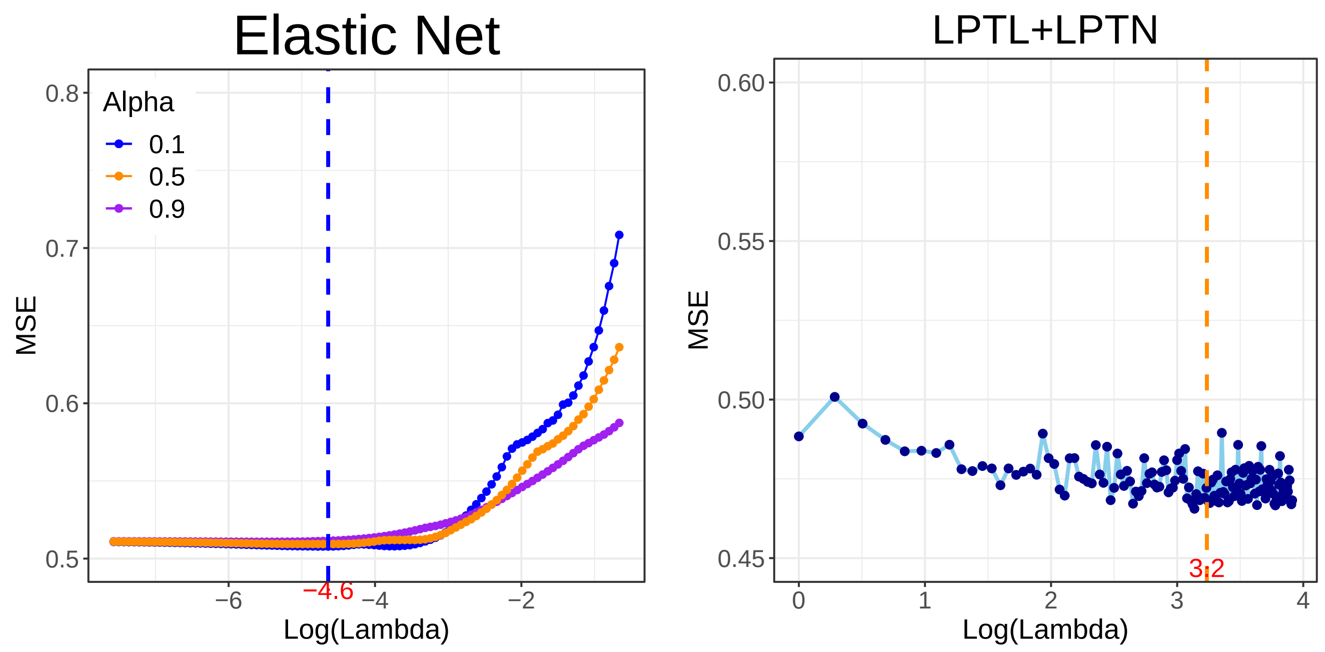

Figure A.5: **Left:** Plot of MSE of elastic net against log(Lambda) with 3 α values: 0.1, 0.5, 0.9; **Right:** Plot of MSE of best LPTL+LPTN model against log(Lambda).

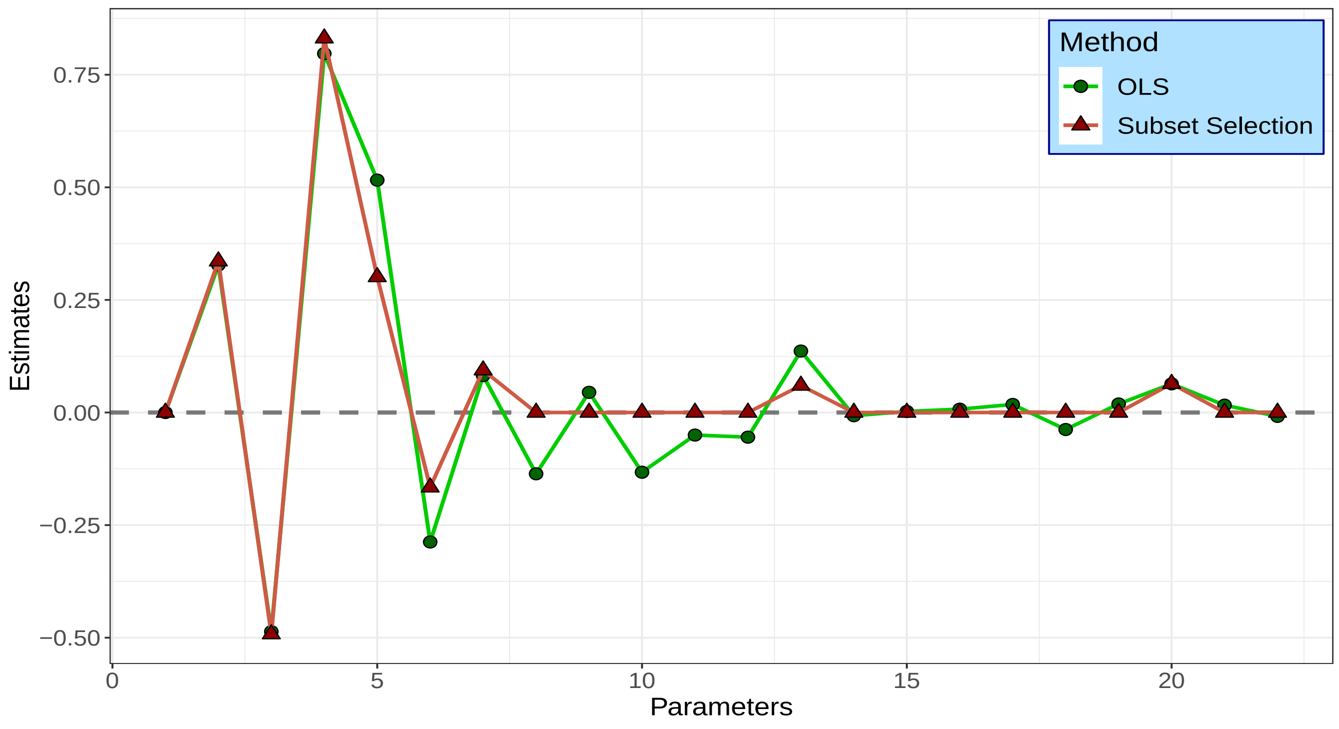

Figure A.6: Plot of best subset selection estimates vs. OLS estiamtes.

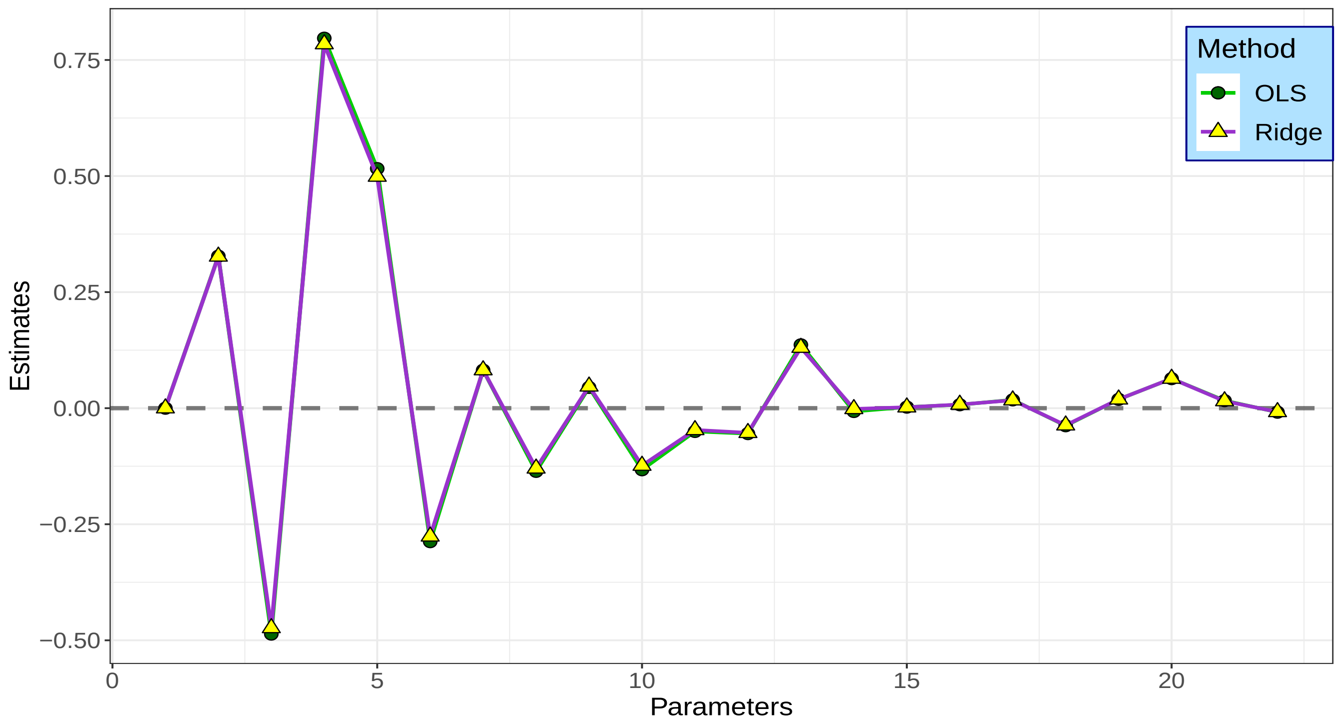

Figure A.7: Plot of best ridge estimates vs. OLS estiamtes.

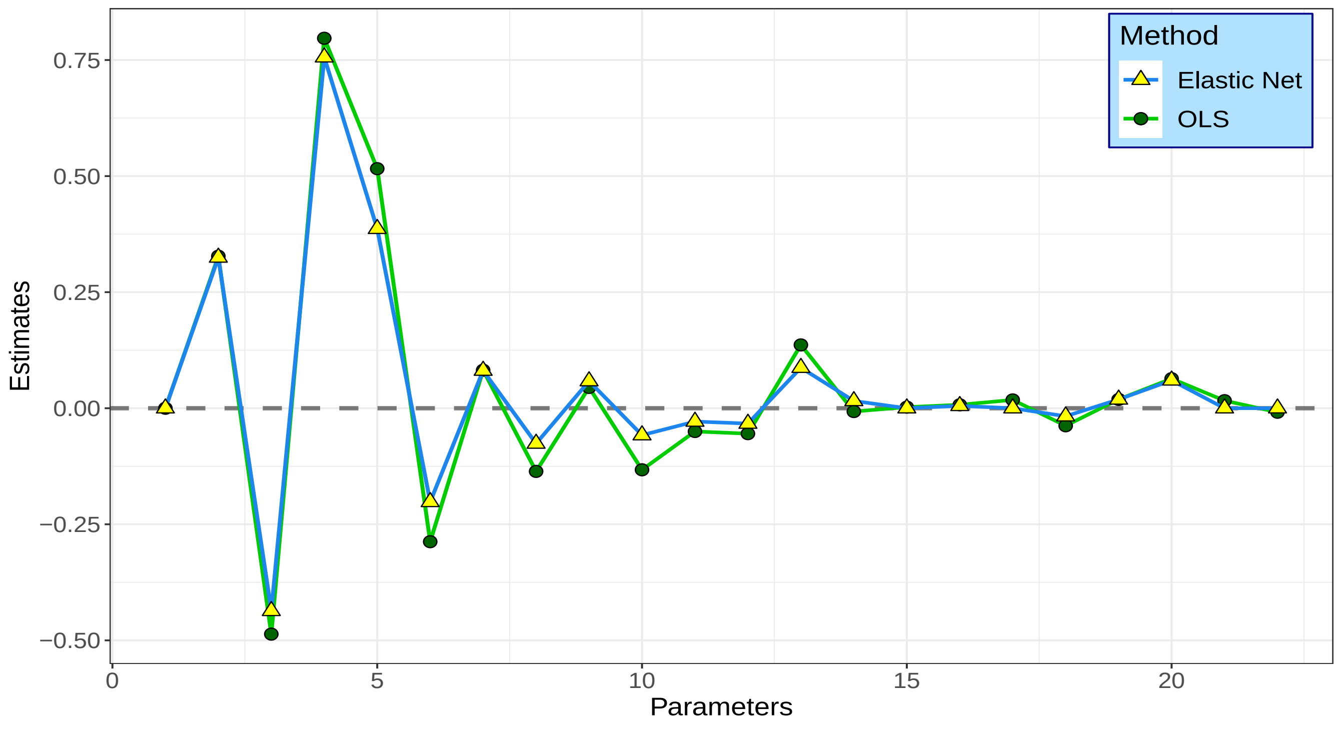

Figure A.8: Plot of best elastic net estimates vs. OLS estiamtes.

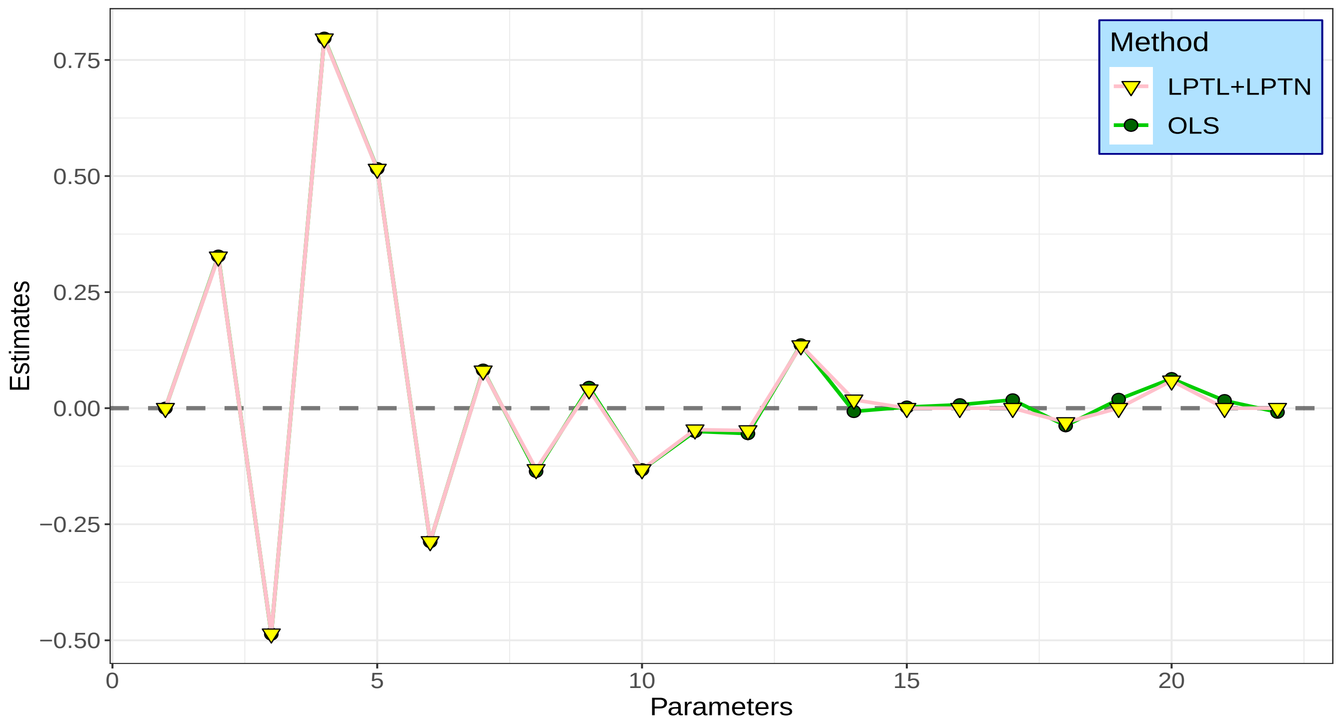

Figure A.9: Plot of best LPTL+LPTN estimates vs. OLS estiamtes.

Appendix B

Software: R codes

```
########################## Densities and likelihoods ##########################

# Density of LPTL distribution
dLPTL <- function(y, tau_LPTL, psi_LPTL, mu = 0, b, log.d = FALSE){
  # tau must be larger than 1, psi_LPTL = lambda_LPTL + 1
  # the scale of the slope (beta) must be greater than 0
  if(b > 0){

    z <- abs(y-mu)/b
    tails <- as.numeric(z <= tau_LPTL)

    # avoid undefined value of log(log(z)) if tails = 1
    z_floor <- apply(cbind(z), 1, max, 1.001)

    logf <- tails*dlaplace(z, log = TRUE) + (1 - tails)*(dlaplace(tau_LPTL, log =
       TRUE) + log(tau_LPTL) - log(z_floor) + psi_LPTL*log(log(tau_LPTL)) - psi_
       LPTL*log(log(z_floor))) - log(b)

    if (log.d == TRUE) {res <- logf} else {res <- exp(logf)}

    return(res)
  }
  else{if (log.d == TRUE) {return(-Inf)} else {return(0)}}
}

# Log posterior density of ridge
logpiRidge <- function(parameters_k, y, C_k, lambda_pri){
  n = length(y)
  sigma = parameters_k[1]/(sqrt(n*lambda_pri))# the scale of the slope

  z <- abs((y - C_k %*% parameters_k[2:length(parameters_k)]/ parameters_k[1])

  logL <- sum(dnorm(z, log = TRUE) - log(parameters_k[1]))

  # log prior(Gaussian) on beta
  log_prior <- sum(dnorm(parameters_k[2:length(parameters_k)], sd = sigma, log =
     TRUE))

  return(log_prior + logL - log(parameters_k[1]))
}
```

```r
# Log posterior density of lasso
logpiLasso <- function(parameters, y, C_k, lambda_pri){

  n = length(y)
  b = 2*(parameters[1]^2)/(lambda_pri*n) # the scale of the slope (beta)

  z <- abs((y - C_k %*% parameters[2:length(parameters)]) / parameters[1])

  logL <- sum(dnorm(z, log = TRUE) - log(parameters[1]))# log likelihood

  # log prior(Laplace) on beta
  log_prior <- sum(dlaplace(parameters[2:length(parameters)], s = b, log = TRUE))

  return(logL + log_prior - log(parameters[1]))
}

# Log prior density of elastic net
d_elastic.net = function(y, b, sigma, mu = 0, log.d = FALSE){

  if(b > 0 && sigma > 0){

    z1 <- abs(y - mu)/b
    z2 <- abs(y- mu)/sigma

    logf <- dlaplace(z1, log = TRUE) - log(b) + dnorm(z2, log = TRUE) - log(sigma)

    if (log.d == TRUE) {res <- logf} else {res <- exp(logf)}

    return(res)
  }
  else{if (log.d == TRUE) {return(-Inf)} else {return(0)}}
}

# Log posterior density of elastic net
logpiEN <- function(parameters_k, y, C_k, lambda_pri, alpha){

  n = length(y)

  # parameters of the prior in elastic net
  b = 2*parameters_k[1]^2/(lambda_pri*n*(1-alpha))
  sigma = parameters_k[1]/(sqrt(n*lambda_pri*alpha))


  z <- abs((y - C_k %*% parameters_k[2:length(parameters_k)]) / parameters_k[1])

  logL <- sum(dnorm(z, log = TRUE) - log(parameters_k[1]))# log likelihood

  # log prior on beta
  log_prior <- sum(d_elastic.net(parameters_k[2:length(parameters_k)], b=b, sigma
     = sigma, log.d = TRUE))

  return( log_prior + logL -log(parameters_k[1]))
}

# Log posterior density of LPTL
logpiLPTL <- function(parameters_k, y, C_k, lambda_pri, tau_LPTL, lambda_LPTL){
  sigma_k <- parameters_k[1]
  beta_k <- as.matrix(parameters_k[-1])

  n = length(y)
  b = (2*(sigma_k^2)/(lambda_pri*n))# the scale of the slope (beta)

  if(sigma_k > 0 & b > 0){

    z <- abs((y - C_k %*% beta_k)) / sigma_k

    logL <- sum(dnorm(z, log = TRUE) - log(sigma_k))# log likelihood

    # log prior (LPTL) on beta
    log_prior <- sum(dLPTL(beta_k, tau_LPTL = tau_LPTL, psi_LPTL = lambda_LPTL+1,
       b = b, log.d = TRUE))

    return(log_prior + logL - log(sigma_k))
  }
  else{return(Inf)}
}

# Log posterior density of LPTL+LPTN
logpiLPTLLPTN <- function(parameters_k, y, C_k, lambda_pri, tau_LPTN, psi_LPTN,
   tau_LPTL, lambda_LPTL){

  sigma_k <- parameters_k[1]
  beta_k <- as.matrix(parameters_k[-1])

  n = length(y)
  b = (2*(sigma_k^2)/(lambda_pri*n))

  z <- abs((y - C_k %*% beta_k)) / sigma_k

  tails <- as.numeric(z <= tau_LPTN) # LPTN tau

  # avoid undefined value of log(log(z)) if tails = 1
  z_floor <- apply(cbind(z), 1, max, 1.001)

  logL <- sum(tails * dnorm(z, log = TRUE) + (1 - tails)*(dnorm(tau_LPTN, log =
     TRUE) + log(tau_LPTN) - log(z_floor) + psi_LPTN * log(log(tau_LPTN)) - psi_
     LPTN * log(log(z_floor))) - log(sigma_k))

  # log prior(LPTL) on beta
  log_prior <- sum(dLPTL(beta_k, tau_LPTL = tau_LPTL, psi_LPTL = lambda_LPTL+1, b
     = b, log.d = TRUE))

  return(log_prior + logL - log(sigma_k))
}

#--------------------------Functions in  Section 3.4 ---------------------------
# Postive objective function of the simple case in
simple_obj<-function(beta,sigma){

  term1 <- 1/(2*sigma^2)*(y- x*beta)^2
  term2 <- (n+1)*log(sigma)+log(4*sigma^2/(lambda*n))
  term3 <- log(abs(beta)*lambda*n/(2*sigma^2))
  term4 <- (lambda_LPTL+1)*log(log(abs(beta)*lambda*n/(2*sigma^2)))
  (term1+term2+term3+term4)
}

# Negative objective function of the simple case in Section 3.4
simple_obj_neg<-function(beta,sigma){

  term1 <- 1/(2*sigma^2)*(y- x*beta)^2
  term2 <- (n+1)*log(sigma)+log(4*sigma^2/(lambda*n))
  term3 <- log(abs(beta)*lambda*n/(2*sigma^2))
  term4 <- (lambda_LPTL+1)*log(log(abs(beta)*lambda*n/(2*sigma^2)))

  -(term1+term2+term3+term4)
}

# A part of the objective funciton in Section 3.4
part_obj<-function(beta,sigma){

  term1<-0
  term2<-(n+1)*log(sigma)+log(4*sigma^2/(lambda*n))
  term3<-log(abs(beta)*lambda*n/(2*sigma^2))
  term4<-(lambda_LPTL+1)*log(log(abs(beta)*lambda*n/(2*sigma^2)))

  (term1+term2+term3+term4)
}

#--------------------------Functions in  Section 3.5 ---------------------------
# Negative log posterior of lasso for the optim R function
negpost_Lasso <- function(parameters, y, X, lambda_pri){

  n = length(y)
  b = 2*(parameters[1]^2)/(lambda_pri*n)# Scale of Laplace

  if(parameters[1] > 0 && b > 0){

    z <- abs((y - X %*% parameters[2:length(parameters)]) / parameters[1])

    logL <- sum(dnorm(z, log = TRUE) - log(parameters[1]))

    # log prior(Laplace) on beta
    log_prior <- sum(dlaplace(parameters[2:length(parameters)], s = b, log = TRUE)
       )

    return(log(parameters[1]) - log_prior - logL)
  }
  else{return(Inf)}
}

# Function to find the global maximum using several starting points
find_MAP_Lasso <- function(nb_start = 1, initial, sd_initial, y, X,lambda_pri){

  MAP <- optim(initial, negpost_Lasso, gr = NULL, y = y, X = X, lambda_pri =
     lambda_pri, method = "Nelder-Mead", control = list(maxit = 40000, reltol
     =10^(-12) ))

  print(list(MAP = round(MAP$par, 2), negpost = MAP$value))

  if (nb_start > 1){
    for (i in 2:nb_start){

      MAP_poss_best <- optim(initial + rnorm(length(initial), 0, sd_initial),
                             negpost_Lasso, gr = NULL, lambda_pri = lambda_pri,
                             y = y, X = X, method = "Nelder-Mead",
                             control = list(maxit = 40000, reltol=10^(-12) ))

      if (MAP_poss_best$value < MAP$value){

        MAP <- MAP_poss_best
        print(list(MAP = round(MAP$par, 2), negpost = MAP$value))
      }
    }
  }
  return(MAP$par)
}

# Negative log posterior of LPTL for the optim R function
negpost_LPTL <- function(parameters, y, X, lambda_pri, tau_LPTL, psi_LPTL){

  n=length(y)
  b=(2*(parameters[1]^2)/(lambda_pri*n))

  if(parameters[1] > 0 & b > 0){

    z <- abs((y - X %*% parameters[2:length(parameters)]) / parameters[1])

    logL <- sum(dnorm(z, log = TRUE) - log(parameters[1]))

    # log prior(LPTL) on beta
    log_prior <- sum(dLPTL(parameters[2:length(parameters)], tau_LPTL = tau_LPTL,
                           psi_LPTL = psi_LPTL, b = b, log.d = TRUE))

    return(log(parameters[1]) - log_prior - logL)
  }
  else{return(Inf)}
}

# Function to find the global maximum using several starting points
find_MAP_LPTL <- function(nb_start , initial, sd_initial, y, X,
                                 lambda_pri, tau_LPTL, psi_LPTL)
{
  MAP <- optim(initial, negpost_LPTL, gr = NULL, y = y, X = X, lambda_pri = lambda
      _pri,tau_LPTL = tau_LPTL, psi_LPTL = psi_LPTL, method = "Nelder-Mead",
     control = list(maxit = 40000, reltol=10^(-12) ))

  print(list(MAP = round(MAP$par, 2), negpost = MAP$value))

  if (nb_start > 1){

    for (i in 2:nb_start){

      MAP_poss_best <- optim(initial + rnorm(length(initial), 0, sd_initial),
          negpost_LPTL, gr = NULL, lambda_pri = lambda_pri, y = y, X = X, tau_LPTL
          = tau_LPTL, psi_LPTL = psi_LPTL, method = "Nelder-Mead", control = list(
          maxit = 40000, reltol=10^(-12) ))
```

```r
      if (MAP_poss_best$value < MAP$value){

        MAP <- MAP_poss_best
        print(list(MAP = round(MAP$par, 2), negpost = MAP$value))
      }
    }
  }
  return(MAP$par)
}

#################### Gradient for the optimx R function ######################

# Gradient of the log posterior density for ridge
gradientRidge <- function(parameters_k, y, C_k, lambda_pri)
{

  n = length(y)

  sigma_k <- parameters_k[1]
  beta_k <- as.matrix(parameters_k[-1])

  d = length(beta_k)
  b_k <- 2*sigma_k^2/(lambda_pri*n)

  # Gradient with respect to beta
  term1 <- -n*lambda_pri*beta_k*sigma_k^(-2)
  term2 = sigma_k^(-2) * (t(C_k) %*% (y - C_k %*% beta_k))

  gradient_beta <- as.matrix(term1) + term2

  # Gradient with respect to sigma
  term_sigma1 = sum(-1/sigma_k + n*lambda_pri*beta_k^2*sigma_k^(-3))-1/sigma_k
  term_sigma2 = 1/sigma_k*(-n + sigma_k^(-2)*sum((y - C_k %*% beta_k)^2))

  gradient_sigma <- term_sigma1 + term_sigma2

  return(c(gradient_sigma, gradient_beta))
}

# Gradient of the log posterior density for lasso
gradientLasso <- function(parameters_k, y, C_k, lambda_pri)
{
  n = length(y)

  sigma_k <- parameters_k[1]
  beta_k <- as.matrix(parameters_k[-1])

  d = length(beta_k)
  b_k <- 2*sigma_k^2/(lambda_pri*n)

  # Gradient with respect to beta
  term1 <- rep(NA, length(beta_k))

  for(j in 1: (length(beta_k)))
  {
    term1[j] <- -beta_k[j]/(abs(beta_k[j])*b_k)
  }
  term2 = sigma_k^(-2) * (t(C_k) %*% (y - C_k %*% beta_k))

  gradient_beta <- as.matrix(term1) + term2

  # Gradient with respect to sigma
  term_sigma1 = -(2*d+1)/sigma_k+ sum((lambda_pri*n*abs(beta_k)/sigma_k^3))
  term_sigma2 = 1/sigma_k*(-n + sigma_k^(-2)*sum((y - C_k %*% beta_k)^2))

  gradient_sigma <- term_sigma1 + term_sigma2

  return(c(gradient_sigma, gradient_beta))
}

# Gradient of the log posterior density for elastic net
gradientEN <- function(parameters_k, y, C_k, lambda_pri, alpha)
{
  n = length(y)

  sigma_k <- parameters_k[1]
  beta_k <- as.matrix(parameters_k[-1])

  d = length(beta_k)
  b_k <- 2*sigma_k^2/(lambda_pri*n)

  # Gradient with respect to beta
  term1 <- rep(NA, length(beta_k))
  for(j in 1:length(beta_k))
  {
    term1[j] <- -n*lambda_pri*alpha*beta_k[j]*sigma_k^(-2) -
      n*lambda_pri*(1-alpha)*beta_k[j]/(2*sigma_k^2*abs(beta_k[j]))
  }
  term2 = sigma_k^(-2) * (t(C_k) %*% (y - C_k %*% beta_k))

  gradient_beta <- as.matrix(term1) + term2

  # Gradient with respect to sigma
  term_sigma1 = sum(-3/sigma_k + n*lambda_pri*alpha*beta_k^2*sigma_k^(-3)+
                      n*lambda_pri*(1-alpha)*abs(beta_k)*sigma_k^(-3))-1/sigma_k
  term_sigma2 = 1/sigma_k*(-n + sigma_k^(-2)*sum((y - C_k %*% beta_k)^2))
  gradient_sigma <- term_sigma1 + term_sigma2

  return(c(gradient_sigma, gradient_beta))
}

# Gradient of the log posterior density for LPTL model
gradientLPTL <- function(parameters_k, y, C_k, lambda_pri, tau_LPTL, lambda_LPTL){

  n = length(y)

  sigma_k <- parameters_k[1]
  beta_k <- as.matrix(parameters_k[-1])

  d = length(beta_k)
  b_k <- 2*sigma_k^2/(lambda_pri*n)

  t_k <- abs(beta_k) / b_k
  tails_LPTL <- as.numeric(t_k <= tau_LPTL)
  # Gradient with respect to beta
  term1 <- rep(NA, length(beta_k))
  term2 <- rep(NA, length(beta_k))

  for(j in 1: (length(beta_k)))
  {
    term1[j] <- (1 - tails_LPTL[j])*(-1/beta_k[j] - (lambda_LPTL+1)/(beta_k[j]*log
       (abs(beta_k[j])/b_k)))
    term2[j] <-  tails_LPTL[j]*(-beta_k[j]/(abs(beta_k[j])*b_k))
  }
  term3 = sigma_k^(-2) * (t(C_k) %*% (y - C_k %*% beta_k))

  gradient_beta <- as.matrix(term1) + as.matrix(term2) +term3

  # Gradient with respect to sigma
  term_sigma1 = -(2*d+1)/sigma_k + 2/(sigma_k)*sum((1-tails_LPTL)*(1 + (lambda_
     LPTL+1)/(log(abs(beta_k)/b_k))))
  term_sigma2 = sum(tails_LPTL * (lambda_pri*n*abs(beta_k)/sigma_k^3))
  term_sigma3 = 1/sigma_k*(-n + (sigma_k^(-2))*sum((y - C_k %*% beta_k)^2))

  gradient_sigma <- term_sigma1 + term_sigma2 + term_sigma3

  return(c(gradient_sigma, gradient_beta))

}

# Gradient of the log posterior density for LPTL+LPTN model
gradientLPTLLPTN <- function(parameters_k, y, C_k, lambda_pri, tau_LPTN, psi_LPTN,
    tau_LPTL, lambda_LPTL){

  n = length(y)

  sigma_k <- parameters_k[1]
  beta_k <- as.matrix(parameters_k[-1])

  d = length(beta_k)
  b_k <- 2*sigma_k^2/(lambda_pri*n)

  z <- abs((y - C_k %*% beta_k)) / sigma_k
  t_k <- abs(beta_k) / b_k

  # z on the tail or not for LPTL and LPTN
  tails_LPTN <- as.numeric(z <= tau_LPTN)
  tails_LPTL <- as.numeric(t_k <= tau_LPTL)

  z_floor <- apply(cbind(z), 1, max, 1.001)

  # Gradient with respect to beta
  term_1 <- rep(NA, length(beta_k))
  term_2 <- rep(NA, length(beta_k))

  for(j in 1: (length(beta_k)))
  {
    term_1[j] <- (1 - tails_LPTL[j])*(-1/beta_k[j] - (lambda_LPTL+1)/(beta_k[j]*
       log(abs(beta_k[j])/b_k)))
    term_2[j] <-  tails_LPTL[j]*(-beta_k[j]/(abs(beta_k[j])*b_k))
  }
  term_3 <- t(C_k) %*% (tails_LPTN*exp(-2*log(sigma_k))*(y - C_k %*% beta_k))
  term_4 <- t(C_k) %*% ((1 - tails_LPTN) / (y - C_k %*% beta_k))
  term_5 <- psi_LPTN*t(C_k) %*% ((1 - tails_LPTN)/((y - C_k %*% beta_k)*log(z)))

  gradient_beta <- as.matrix(term_1)+as.matrix(term_2)+term_3+term_4+term_5

  # Gradient with respect to sigma
  term_sigma1 = -(2*d+1)/sigma_k + 2/(sigma_k)*sum((1-tails_LPTL)*(1 + (lambda_
      LPTL+1)/(log(abs(beta_k)/b_k))))
  term_sigma2 = sum(tails_LPTL * (lambda_pri*n*abs(beta_k)/sigma_k^3))
  term_sigma3 <- -n/sigma_k + sigma_k^(-3) * sum((y - C_k %*% beta_k)^2 * tails_
      LPTN)
  term_sigma4 <- sigma_k^(-1)*sum((1 - tails_LPTN) * (1 + psi_LPTN/log(z)))

  gradient_sigma <- term_sigma1 + term_sigma2 + term_sigma3 + term_sigma4

  return(c(gradient_sigma, gradient_beta))
}

############## Functions for the Cross Validation algorithm ################

#------------- Cross-validation for subset selection ---------------------
# Tune best lambda on each training set
tune.subset_model = function(train_nba, X_std_train_nba, nfolds_inner)
{
  n = nrow(train_nba)
  set.seed(2019)
  folds_cv = split(sample(n), seq_len(nfolds_inner)) # create inner folds

  # Matrix to store the MSE value in cross validation
  cv_MSE = matrix(NA, nfolds_inner, p-1, dimnames=list(NULL, paste(1:(p-1))))

  # Inner cross-validation
  for(i in seq(nfolds_inner)){
    # Separate training data set
    dtrain = train_nba[-folds_cv[[i]],]
    dvalid = train_nba[folds_cv[[i]],]

    X_std_dtrain = cbind(1, dtrain[,2:(dim(dtrain)[2])])
    X_std_dvalid = cbind(1, dvalid[,2:(dim(dvalid)[2])])

    # Using regsubsets function to fit models
    best.fit = regsubsets(Salary ~., data = dtrain, nvmax = (p-1))

    for (j in 1:(p-1)){

      # Estimates of coefficients for different number of variables
      coef_cv = coef(best.fit, id=j)

      # Predict on validation set
      pred = as.matrix(X_std_dvalid[,c("1",names(coef_cv)[-1])])%*% coef_cv

      # Calculate the MSE on each inner fold
      cv_MSE[i, j] = mean((dvalid$Salary - pred)^2)
    }
  }
  subset_MSE = apply(cv_MSE , 2, mean)

  save(subset_MSE, file = "subset_MSE.RData") # for plot

  # Choose the model with number of variables with minimum MSE
  return(which.min(subset_MSE))
}

# Outer cross-validation
outerloop.subset = function(i)
{
  fold = folds[[i]]

  # Separate the whole dataset to training set and test set
  train_nba = data_standard[-fold, ]
  test_nba  = data_standard[fold, ]

  X_std_train_nba = cbind(1, train_nba[,2:(dim(train_nba)[2])])
  X_std_test_nba = cbind(1, test_nba[,2:(dim(test_nba)[2])])

  # Tune best lambda on each outer fold
  n_var.best = tune.subset_model(train_nba = train_nba, X_std_train_nba = X_std_
      train_nba, nfolds_inner = nfolds_inner)

  # Perform best subset selection on the training set
  subset_model = regsubsets (Salary ~. , data = train_nba, nvmax = n_var.best)

  # Obtain the estimates from best model
  coef_model = coef(subset_model, n_var.best)

  # Predict on test set
  pred = as.matrix(X_std_test_nba[,c("1",names(coef_model)[-1])]) %*% coef_model

  # Calculate the MSE
  mse_subset = mean((test_nba[,1] - pred)^2)

  n_var = n_var.best # best number of variables for each outer loop

  return(c(mse_subset = mse_subset, n_var = n_var))
}

#--------------------- Nested cross validation for ridge --------------------
# Inner cross-validation and measure on each fold
cval_ridge.x = function(lambda, k)
{
  n = nrow(train_nba)
  set.seed(2019)
  folds_cv = split(sample(n), seq_len(k)) # create inner folds

  # Inner cross-validation
  cval.fold_ridge.x = function(fold, data_cv, lambda)
  {
    # Separate training data set
    dtrain = data_cv[-fold, ]
    dvalid  = data_cv[fold, ]

    X_std_train = cbind(1, dtrain[,2:dim(dtrain)[2]])
    X_std_valid = cbind(1, dvalid[,2:dim(dvalid)[2]])

    # Optimx with BFGS optimiser
    Ridge.fold.x = optimx(parameters_k, logpiRidge, gr = gradientRidge, method = "
       BFGS",
                          control = list(maximize = TRUE, trace = 0, reltol =
                             10^(-12)),
                          y = dtrain[,1], C_k = as.matrix(X_std_train), lambda_pri
                              = lambda)

    # Predict on validation set
    pred_cv = as.matrix(X_std_valid) %*% unlist(as.vector(Ridge.fold.x)[2:(dim(
       train_nba)[2]+1)])

    mse = mean((dvalid[,1] - pred_cv)^2) # Mean Squared Error

    return(mse)
  }

  mse_cv = lapply(folds_cv,cval.fold_ridge.x,data_cv = train_nba,lambda = lambda)

  mean_mse = mean(unlist(mse_cv)) # average of MSE for 10 inner folds

  return(c(mean_mse = mean_mse, lambda_cur = lambda))
}

# Tune best lambda on each training set
tune.Ridge_model.para.x = function(train_nba, X_std_train_nba, parameters_k,
    cluster, lambda, nfolds_inner)
{
  # Eport data, functions,library to nodes at the beginning to reduce overhead
  clusterExport(cluster, "parameters_k")
  clusterExport(cluster, "train_nba", envir = environment())
  clusterEvalQ(cluster, library(rmutil))
  clusterEvalQ(cluster, library(optimx))
  clusterExport(cluster, c("logpiRidge", "gradientRidge"))

  # Parallel on 100 models (100 lambdas)
  result_ridge_cv.x = parSapply(cluster, lambda, cval_ridge.x , k = nfolds_inner)

  save(result_ridge_cv.x, file= "result_ridge_cv.x.RData") # for plot

  # Select best lambda based on the mean MSE in inner cross-validation
  index = which.min(result_ridge_cv.x['mean_mse',])
  lambda_best = result_ridge_cv.x['lambda_cur',index]

  # Refit using the best lambda on the training set (including validation set)
  Ridge_model_cv.x = optimx(parameters_k, logpiRidge, gr = gradientRidge, method =
      "BFGS", control = list(maximize = TRUE, trace = 0, reltol = 10^(-12)), y =
     train_nba[,1], C_k = as.matrix(X_std_train_nba), lambda_pri = lambda_best)

  return(c(lambda_best, Ridge_model_cv.x = unlist(as.vector(Ridge_model_cv.x)[1:(
     dim(train_nba)[2]+1)])))
}

# Outer cross-validation
outerloop.ridge.x = function(i)
```

```
{
  fold = folds[[i]]

  # Separate the whole dataset to training set and test set
  train_nba = data_standard[-fold, ]
  test_nba  = data_standard[fold, ]

  X_std_train_nba = cbind(1, train_nba[,2:dim(train_nba)[2]])
  X_std_test_nba = cbind(1, test_nba[,2:dim(test_nba)[2]])

  # Tune best lambda on each outer fold
  Ridge_model.x = tune.Ridge_model.para.x(train_nba = train_nba, X_std_train_nba =
       X_std_train_nba, parameters_k = parameters_k, cluster = c1, nfolds_inner =
     nfolds_inner, lambda = lambda_path_ridge)
  # Predict on test set
  pred = as.matrix(X_std_test_nba) %*% Ridge_model.x[c(-1,-2)]

  mse_ridge.x = mean((test_nba[,1] - pred)^2) # Mean-squared error

  lambmda_best_ridge.x = Ridge_model.x[1] # best lambda for each outer fold

  return(c(mse_ridge.x = mse_ridge.x, lambmda_best_ridge.x))
}

#--------------------- Nested cross validation for lasso --------------------
# Inner cross-validation and measure on each fold
cval_lasso.x = function(lambda, k)
{
  n_train = nrow(train_nba)
  set.seed(2019)
  folds_inner = split(sample(n_train), seq_len(k)) # create inner folds

  # Inner cross-validation
  cval.fold_lasso.x = function(fold, data_cv, lambda)
  {
    # Separate training data set
    dtrain = data_cv[-fold, ]
    dvalid  = data_cv[fold, ]

    X_std_train = cbind(1, dtrain[,2:(dim(data_cv)[2])])
    X_std_valid = cbind(1, dvalid[,2:(dim(data_cv)[2])])

    # Optimx with BFGS optimiser
    Lasso.fold.x = optimx(parameters_k, logpiLasso, gr = gradientLasso, method = "
       BFGS", control = list(maximize = TRUE, trace = 0, reltol = 10^(-12)), y =
       dtrain[,1], C_k = as.matrix(X_std_train), lambda_pri = lambda)

    # Predict on validation set
    pred_cv = as.matrix(X_std_valid) %*% unlist(as.vector(Lasso.fold.x)[2:(dim(
       train_nba)[2]+1)])

    mse = mean((dvalid[,1] - pred_cv)^2)  # Mean-squared error

    return(mse)
  }

  mse_cv = lapply(folds_inner,cval.fold_lasso.x,data_cv=train_nba,lambda = lambda)
  mean_mse = mean(unlist(mse_cv)) # average of MSE for 10 inner folds

  return(c(mean_mse = mean_mse, lambda_cur = lambda))
}

# Tune best lambda on each training set
tune.Lasso_model.para.x = function(train_nba, X_std_train_nba, parameters_k,
    cluster, lambda, nfolds_inner)
{
  # Eport data, functions,library to nodes at the beginning to reduce overhead
  clusterExport(cluster, "train_nba", envir = environment())
  clusterExport(cluster, "parameters_k")
  clusterEvalQ(cluster, library(rmutil))
  clusterEvalQ(cluster, library(optimx))
  clusterExport(cluster, c("gradientLasso","logpiLasso"))

  # Parallel on 100 models (100 lambdas)
  result_lasso_cv.x = parSapply(cluster, lambda, cval_lasso.x, k = nfolds_inner)

  save(result_lasso_cv.x, file = "result_lasso_cv.x.RData") # for plot

  # Select best lambda based on the mean MSE in inner cross-validation
  index = which.min(result_lasso_cv.x['mean_mse',])
  lambda_best = result_lasso_cv.x['lambda_cur',index]

  # Refit using the best lambda on the training set (including validation set)
  Lasso_model_cv.x = optimx(parameters_k, logpiLasso, gr = gradientLasso, method =
      "BFGS",control = list(maximize = TRUE, trace = 0, reltol = 10^(-12)), y =
     train_nba[,1], C_k = as.matrix(X_std_train_nba), lambda_pri = lambda_best)

  return(c(lambda_best = lambda_best, Lasso_model_cv.x = unlist(as.vector(Lasso_
     model_cv.x)[1:(dim(train_nba)[2]+1)])))
}

# Outer cross-validation
outerloop.lasso.x = function(i)
{
  fold = folds_outer[[i]]

  # Separate the whole dataset to training set and test set
  train_nba = as.matrix(data_standard[-fold, ])
  test_nba  = as.matrix(data_standard[fold, ])

  X_std_train_nba = cbind(1, train_nba[,2:(dim(train_nba)[2])])
  X_std_test_nba = cbind(1, test_nba[,2:(dim(test_nba)[2])])

  # Tune best lambda on each outer fold
  Lasso_model.x = tune.Lasso_model.para.x(train_nba = train_nba,X_std_train_nba =
     X_std_train_nba,parameters_k = parameters_k, cluster = c1, nfolds_inner =
     nfolds_inner, lambda = lambda_path_lasso)
  # Predict on test set
  pred = as.matrix(X_std_test_nba) %*% Lasso_model.x[c(-1, -2)]

  mse_lasso.x = mean((test_nba[,1] - pred)^2)  # Mean-squared error

  lambda_best.x = Lasso_model.x[1] # best lambda for each outer fold
  return(c(mse_lasso.x = mse_lasso.x, lambda_best.x = lambda_best.x))
}

#------------------ Nested cross validation for Elastic Net ------------------
# Inner cross-validation and measure on each fold
cval_elasti.net.x = function(lambda, alpha, k)
{
  n = nrow(train_nba)
  set.seed(2019)
  folds_cv = split(sample(n), seq_len(k)) # create inner folds

  # Inner cross-validation
  cval.fold_elastic.net.x = function(fold, data_cv, lambda, alpha)
  {
    # Separate training data set
    dtrain = data_cv[-fold, ]
    dvalid  = data_cv[fold, ]

    X_std_train = cbind(1, dtrain[,2:dim(dtrain)[2]])
    X_std_valid = cbind(1, dvalid[,2:dim(dvalid)[2]])

    # Optimx with BFGS optimiser
    Elastic.net.fold.x = optimx(parameters_k, logpiEN, gr = gradientEN, method = "
       BFGS",control = list(maximize = TRUE, trace = 0,reltol = 10^(-12)),y =
       dtrain[,1],C_k = as.matrix(X_std_train),lambda_pri = lambda, alpha = alpha)

    # Predict on validation set
    pred_cv = as.matrix(X_std_valid) %*% unlist(as.vector(Elastic.net.fold.x)[2:(
       dim(train_nba)[2]+1)])

    mse = mean((dvalid[,1] - pred_cv)^2) # Mean Squared Error
    return(mse)
  }

  mse_cv = sapply(folds_cv, cval.fold_elastic.net.x, data_cv = train_nba,
                  lambda = lambda, alpha = alpha)

  mean_mse = mean(mse_cv) # average of MSE for 10 inner folds

  return(mean_mse)
}

# Tune best lambda on each training set
tune.Elastic.net_model.para.x = function(train_nba, X_std_train_nba, parameters_k,
     cluster, lambda_path, alpha_path, nfolds_inner)
{
  # Eport data, functions,library to nodes at the beginning to reduce overhead
  clusterExport(cluster, "parameters_k")
  clusterExport(cluster, "train_nba", envir = environment())
  clusterEvalQ(cluster, library(rmutil))
  clusterEvalQ(cluster, library(optimx))
  clusterExport(cluster, c("logpiEN", "gradientEN","d_elastic.net"))

  lambda_path  =lambda_path_elastic.net
  result_elastic.net_cv.x=matrix(NA,ncol=length(lambda_path),nrow=length(alpha_
     path))
  # For each alpha, parallel on lambda
  for(i in 1:length(alpha_path))
  {
    # Parallel on 100 models (100 lambdas)
    result_elastic.net_cv.x[i, ]=parSapply(cluster,lambda_path,cval_elasti.net.x,
        alpha = alpha_path[i],  k = nfolds_inner)
  }
  save(result_elastic.net_cv.x, file= "result_elastic.net_cv.x.RData") # for plot

  # Select best lambda and alpha based on the mean MSE in inner cross-validation
  index_alpha=which(result_elastic.net_cv.x==min(result_elastic.net_cv.x), arr.ind
      = TRUE)[1]
  index_lambda = which(result_elastic.net_cv.x == min(result_elastic.net_cv.x),
     arr.ind = TRUE)[2]

  alpha_best = alpha_path[index_alpha]
  lambda_best = lambda_path[index_lambda]

  # Refit using the best lambda on the training set (including validation set)
  Elastic.net_model_cv.x = optimx(parameters_k, logpiEN, gr = gradientEN, method =
      "BFGS",control = list(maximize = TRUE, trace = 0, reltol = 10^(-12)), y =
     train_nba[,1], C_k = as.matrix(X_std_train_nba),lambda_pri = lambda_best,
     alpha = alpha_best)

  return(c(lambda_best = lambda_best, alpha_best = alpha_best, Elastic.net_model_
     cv.x =  unlist(as.vector(Elastic.net_model_cv.x)[1:(dim(train_nba)[2]+1)])))
}

# Outer cross-validation
outerloop.elastic.net.x = function(i)
{
  fold = folds[[i]]

  # Separate the whole dataset to training set and test set
  train_nba = data_standard[-fold, ]
  test_nba  = data_standard[fold, ]

  X_std_train_nba = cbind(1, train_nba[,2:dim(train_nba)[2]])
  X_std_test_nba = cbind(1, test_nba[,2:dim(test_nba)[2]])

  # Tune best lambda on each outer fold
  Elastic.net_model.x = tune.Elastic.net_model.para.x(train_nba = train_nba, X_std
     _train_nba = X_std_train_nba, parameters_k =  parameters_k, cluster = c1,
     nfolds_inner = nfolds_inner,lambda_path = lambda_path_elastic.net, alpha_path
      = alpha_path)
  # Predict on test set
  pred = as.matrix(X_std_test_nba) %*% Elastic.net_model.x[c(-1,-2,-3)]

  mse_elastic.net.x = mean((test_nba[,1] - pred)^2) # Mean-squared error

  return(c(mse_elastic.net.x = mse_elastic.net.x, lambda_best_elastic.net.x =
     Elastic.net_model.x[1],alpha_best_elastic.net.x = Elastic.net_model.x[2]))
}

#------------------ Nested cross validation for LPTL model ------------------
# Inner cross-validation and measure on each fold
cval_Normal_LPTL.x = function(lambda, k)
```

```r
{
  n = nrow(train_nba)
  set.seed(2019)
  folds_cv = split(sample(n), seq_len(k))# create inner folds

  # Inner cross-validation
  cval.fold_Normal_LPTL.x = function(fold, data_cv, lambda, tau_LPTL, lambda_LPTL)
  {
    # Separate training data set
    dtrain = data_cv[-fold, ]
    dvalid  = data_cv[fold, ]

    X_std_train = cbind(1, dtrain[,2:dim(dtrain)[2]])
    X_std_valid = cbind(1, dvalid[,2:dim(dtrain)[2]])

    # Optimx with BFGS optimiser
    Normal_LPTL.fold = optimx(parameters_k, logpiLPTL, gr = gradientLPTL, method =
         "BFGS",control = list(maximize = TRUE, trace = 0, reltol = 10^(-12)), y =
        dtrain[,1], C_k = as.matrix(X_std_train), lambda_pri = lambda, tau_LPTL =
        tau_LPTL, lambda_LPTL = lambda_LPTL)

    # Predict on validation set
    pred_cv = as.matrix(X_std_valid) %*% unlist(as.vector(Normal_LPTL.fold)[2:(dim
        (dtrain)[2]+1)])

    mse = mean((dvalid[,1] - pred_cv)^2) # Mean Squared Error

    return(mse)
  }

  mse_cv = sapply(folds_cv, cval.fold_Normal_LPTL.x, data_cv = train_nba, lambda =
      lambda,tau_LPTL = tau_LPTL, lambda_LPTL = lambda_LPTL)

  mean_mse = mean(mse_cv) # average of MSE for 10 inner folds

  return(c(mean_mse = mean_mse, lambda_cur = lambda))
}

# Tune best lambda on each training set
tune.Normal_LPTL_model.para.x = function(train_nba, X_std_train_nba, parameters_k,
     cluster, lambda, nfolds_inner, tau_LPTL, lambda_LPTL)
{
  # Eport data, functions,library to nodes at the beginning to reduce overhead
  clusterExport(cluster, "train_nba", envir = environment())
  clusterExport(cluster, c("parameters_k", "tau_LPTL", "lambda_LPTL"))
  clusterEvalQ(cluster, library(rmutil))
  clusterEvalQ(cluster, library(optimx))
  clusterEvalQ(cluster, library(caret))
  clusterExport(cluster, c("gradientLPTL","dLPTL","logpiLPTL"))

  # Parallel on 100 models (100 lambdas)
  result.Normal_LPTL.x.cv = parSapply(cluster, lambda_path_LPTL, cval_Normal_LPTL.
      x, k = nfolds_inner)

  save(result.Normal_LPTL.x.cv, file= "result.Normal_LPTL.x.cv.RData") # for plot

  # Select best lambda based on the mean MSE in inner cross-validation
  index = which.min(result.Normal_LPTL.x.cv['mean_mse',])
  lambda_best = result.Normal_LPTL.x.cv['lambda_cur',index]

  # Refit using the best lambda on the training set (including validation set)
  result_normal_LPTL.x.cv =optimx(parameters_k, logpiLPTL, gr = gradientLPTL,
     method = "BFGS",control = list(maximize = TRUE, trace = 0, reltol = 10^(-12))
     , y = train_nba[,1], C_k = as.matrix(X_std_train_nba), lambda_pri = lambda_
     best, tau_LPTL = tau_LPTL, lambda_LPTL = lambda_LPTL)

  return(c(lambda_best = lambda_best, result_normal_LPTL.x.cv = unlist(as.vector(
     result_normal_LPTL.x.cv)[1:(dim(train_nba)[2]+1)])))
}

# Outer cross-validation
outerloop.nomal_LPTL.x = function(i)
{
  fold = folds[[i]]
  train_nba = data_standard[-fold, ]
  test_nba  = data_standard[fold, ]

  # Separate the whole dataset to training set and test set
  X_std_train_nba = cbind(1, train_nba[,2:dim(train_nba)[2]])
  X_std_test_nba = cbind(1, test_nba[,2:dim(train_nba)[2]])

  # Tune best lambda on each outer fold
  Normal_LPTL_model.x = tune.Normal_LPTL_model.para.x(train_nba = train_nba, X_std
     _train_nba = X_std_train_nba, parameters_k = parameters_k, cluster = c1,
     nfolds_inner = nfolds_inner, lambda = lambda_path_LPTL, tau_LPTL = tau_LPTL,
     lambda_LPTL = lambda_LPTL)
  # Predict on test set
  pred = as.matrix(X_std_test_nba) %*% Normal_LPTL_model.x[-c(1,2)]

  mse_normal_LPTL.x = mean((test_nba[,1] - pred)^2) # Mean-squared error

  lambda_best = Normal_LPTL_model.x[1] # best lambda for each outer fold

  return(c(mse_normal_LPTL.x = mse_normal_LPTL.x, lambda_best = lambda_best))
}

#----------------- Nested cross validation for LPTL+LPTN model -----------------
# Inner cross-validation and measure on each fold
cval_LPTN_LPTL.x = function(lambda, k){
  n = nrow(train_nba)
  set.seed(2019)
  folds_cv = split(sample(n), seq_len(k)) # create inner folds

  # Inner cross-validation
  cval.fold_Normal_LPTL.x = function(fold, data_cv, lambda, tau_LPTN, lambda_LPTN,
      tau_LPTL, lambda_LPTL){

    # Separate training data set
    dtrain = data_cv[-fold, ]
    dvalid  = data_cv[fold, ]

    X_std_train = cbind(1, dtrain[,2:dim(dtrain)[2]])
    X_std_valid = cbind(1, dvalid[,2:dim(dtrain)[2]])
    # Optimx with BFGS optimiser
    LPTN_LPTL.fold = optimx(par = parameters_k, fn = logpiLPTLLPTN, gr =
       gradientLPTLLPTN, method = "BFGS",control = list(maximize = TRUE, trace =
       0, reltol = 10^(-12)), y = dtrain[,1], C_k = as.matrix(X_std_train), lambda
       _pri = lambda, tau_LPTN  = tau_LPTN, psi_LPTN  = lambda_LPTN +1,tau_LPTL =
       tau_LPTL, lambda_LPTL = lambda_LPTL)

    # Predict on validation set
    pred_cv = as.matrix(X_std_valid) %*% unlist(as.vector(LPTN_LPTL.fold)[2:(dim(
       dtrain)[2]+1)])

    mse = mean((dvalid[,1] - pred_cv)^2) # Mean Squared Error
    return(mse)
  }

  mse_cv = sapply(folds_cv, cval.fold_Normal_LPTL.x, data_cv = train_nba,
                  lambda = lambda, tau_LPTN = tau_LPTN, lambda_LPTN = lambda_LPTN,
                  tau_LPTL = tau_LPTL, lambda_LPTL = lambda_LPTL)

  mean_mse = mean(mse_cv) # average of MSE for 10 inner folds

  return(c(mean_mse = mean_mse, lambda_cur = lambda))
}

# Tune best lambda on each training set
tune.LPTN_LPTL_model.para.x = function(train_nba, X_std_train_nba, parameters_k,
    cluster, nfolds_inner, lambda, tau_LPTN, lambda_LPTN,  tau_LPTL, lambda_LPTL){

  # Eport data, functions,library to nodes at the beginning to reduce overhead
  clusterExport(cluster, "train_nba", envir = environment())
  clusterExport(cluster, c("parameters_k", "tau_LPTL", "lambda_LPTL", "tau_LPTN",
     "lambda_LPTN"))
  clusterEvalQ(cluster, library(rmutil))
  clusterEvalQ(cluster, library(optimx))
  # Export function to the clusters
  clusterExport(cluster, c("gradientLPTLLPTN","dLPTL","logpiLPTLLPTN"))

  # Parallel on 100 models (100 lambdas)
  result.LPTN_LPTL.x.cv = parSapply(cluster, lambda_path_LPTL, cval_LPTN_LPTL.x, k
      = nfolds_inner)
  save(result.LPTN_LPTL.x.cv, file = "result.LPTN_LPTL.x.cv.RData")# for plot

  # Select best lambda based on the mean MSE in inner cross-validation
  index = which.min(result.LPTN_LPTL.x.cv['mean_mse',])
  lambda_best = result.LPTN_LPTL.x.cv['lambda_cur',index]

  # Refit using the best lambda on the training set (including validation set)
  result_LPTN_LPTL.x.cv =optimx(parameters_k, logpiLPTLLPTN, gr = gradientLPTLLPTN
     , method = "BFGS",control = list(maximize = TRUE, trace = 0, reltol =
     10^(-12)), y = train_nba[,1], C_k = as.matrix(X_std_train_nba), lambda_pri =
     lambda_best, tau_LPTN  = tau_LPTN, psi_LPTN  = lambda_LPTN +1, tau_LPTL = tau
     _LPTL, lambda_LPTL = lambda_LPTL)

  return(c(lambda_best = lambda_best, result_LPTN_LPTL.x.cv = unlist(as.vector(
     result_LPTN_LPTL.x.cv)[1:(dim(train_nba)[2]+1)])))
}
```

```
# Outer cross-validation
outerloop.LPTN_LPTL.x = function(i)
{
  fold = folds[[i]]

  # Separate the whole dataset to training set and test set
  train_nba = data_standard[-fold, ]
  test_nba  = data_standard[fold, ]

  X_std_train_nba = cbind(1, train_nba[,2:dim(train_nba)[2]])
  X_std_test_nba = cbind(1, test_nba[,2:dim(train_nba)[2]])

  # Tune best lambda on each outer fold
  LPTN_LPTL_model.x = tune.LPTN_LPTL_model.para.x(train_nba = train_nba,X_std_
     train_nba = X_std_train_nba,parameters_k = parameters_k, cluster = c1, nfolds
     _inner = nfolds_inner, lambda = lambda_path_LPTL,tau_LPTN = tau_LPTN,lambda_
     LPTN = lambda_LPTN, tau_LPTL = tau_LPTL, lambda_LPTL = lambda_LPTL)
  # Predict on test set
  pred = as.matrix(X_std_test_nba) %*% LPTN_LPTL_model.x[-c(1,2)]

  mse_LPTN_LPTL.x = mean((test_nba[,1] - pred)^2) # Mean-squared error

  lambda_best = LPTN_LPTL_model.x[1] # best lambda for each outer fold

  return(c(mse_LPTN_LPTL.x = mse_LPTN_LPTL.x, lambda_best = lambda_best))
}

##################### Figure 1.1 #######################
# Initialization
beta = seq(-2, 3, 0.01)
n=4; y=1; s=0.5

# Prior density
prior = 1/sqrt(2*s)*exp(-abs(beta)/s)

# Likelihood density
likelihood = sqrt(n/(2*pi))*exp(-n*(y-beta)^2/2)

# Posterior density
posterior = likelihood*prior

pdf("11.pdf",width=8,height=8)
par(mar=c(5,5,2,2)+0.1)

plot(beta,likelihood, type = "l",ylim = c(0,1), lwd = 4,col="mediumblue",
     lty=5, cex.lab = 1.5, cex = 1.5, cex.axis = 1.5, ylab= "density")

lines(beta,prior, col="darkorange",lwd=4, lty=4)
lines(beta, posterior, lwd = 4, col="red",lty=1)
legend("topright", legend = c("Likelihood", "Prior", "Posterior"),  lwd = 3,
       col = c("blue", "darkorange", "red"), lty = c(5, 4, 1), cex = 1.5)
dev.off()

##################### Figure 3.1 #######################
library(rmutil) # Laplace distribution

pdf("LPTL1.pdf", width = 10, height = 10)
```

```
b = 1 # scale of the laplace
z <- seq(-6, 6, .01)
par(mar = c(4, 5, 1, 3))

# parameters in LPTL
rho_LPTL <- 0.80
tau_LPTL <- qlaplace((1 + rho_LPTL) / 2)
lambda_LPTL <- 2 * tau_LPTL * log(tau_LPTL) * dlaplace(tau_LPTL) /
                 (1 - (plaplace(tau_LPTL) - plaplace(-tau_LPTL)))

plot(z, dLPTL(y = z, b=b, tau_LPTL = tau_LPTL, psi_LPTL = lambda_LPTL + 1),type =
    "l", cex.lab = 1.5, cex.axis = 1.5, cex = 1.5, lwd = 3, xlab = expression(x),
    ylab = expression(f(x)), col = "darkorange", lty = 1)

rho_LPTL <- 0.90
tau_LPTL <- qlaplace((1 + rho_LPTL) / 2) # 2.3
lambda_LPTL <- 2 * tau_LPTL * log(tau_LPTL) * dlaplace(tau_LPTL) /
                 (1 - (plaplace(tau_LPTL) - plaplace(-tau_LPTL)))
lines(z,dLPTL(y = z, tau_LPTL=tau_LPTL, psi_LPTL = lambda_LPTL+1, b=b),type = "l",
     lwd = 3, lty = 5, col = "blue")

rho_LPTL <- 0.95
tau_LPTL <- qlaplace((1 + rho_LPTL) / 2)
lambda_LPTL <- 2 * tau_LPTL * log(tau_LPTL) * dlaplace(tau_LPTL) /
                 (1 - (plaplace(tau_LPTL) - plaplace(-tau_LPTL)))
lines(z, dLPTL(y = z, tau_LPTL = tau_LPTL, psi_LPTL = lambda_LPTL + 1, b=b),
      type = "l", lwd = 3, lty = 4, col = "darkgreen")

legend("topleft", legend = c("LPTL(0.80)", "LPTL(0.90)", "LPTL(0.95)"), lwd = 3,
       col = c("darkorange", "blue", "darkgreen"), lty = c(1, 5, 4), cex = 1.5)
dev.off()

##################### Figure 3.2 ########################
pdf("LPTL2.pdf", width = 10, height = 10)
z2 = seq(2.5, 15, .01)

plot(z2, dLPTL(y = z2, tau_LPTL = tau_LPTL, psi_LPTL = lambda_LPTL + 1, b=b),
     type = "l", lwd = 3, lty = 5, col = "blue", xlab = "x", cex.lab = 1.5,
     cex.axis = 1.5, cex = 1.5, ylab = "f(x)")

lines(z2, dnorm(z2),  type = "l", col = "darkorange",lwd = 3, lty = 1)

lines(z2, dlaplace(z2, s =b), type = "l", cex.lab = 1.5, cex.axis = 1.5,
      cex = 1.5, lwd = 3, col = "darkgreen", lty = 4)

legend("topright", legend = c("LPTL(0.90)", "Normal", "Laplace"),
       col = c("blue", "darkorange", "darkgreen"),
       lty = c(5, 1, 4), cex = 1.5, lwd = 3)
dev.off()

##################### Figure 3.3 ########################
# Parameters of LPTL in this plot
rho_LPTL <- 0.95
tau_LPTL <- qlaplace((1 + rho_LPTL) / 2)
lambda_LPTL <- 2 * tau_LPTL * log(tau_LPTL) * dlaplace(tau_LPTL) /
  (1 - (plaplace(tau_LPTL) - plaplace(-tau_LPTL)))
```

```
# Initialization of the simple case
beta<-seq(0.15,3,length=50)
sigma<-seq(0.2, 1, length=50)
lambda=50; y=1; x=0.8; n=1

# calculating the density values
z <- outer(beta,sigma,simple_obj)
z_neg <-outer(beta,sigma, simple_obj_neg)

# 3-D plot of negative objective funtion, lambda=50
pdf("3d.pdf",width = 10,height = 8)
persp(beta, sigma, z_neg, main="Two dimensional Negative Objective Function",
      col="lightblue", theta=30, phi=20, r=50, d=0.1, expand=0.5,
      ltheta=90, lphi=180, shade=0.75, ticktype="detailed",nticks=5,
      zlab = "negative of f", cex.lab = 1.5, cex.main=2,
      cex.axis = 1.5, cex = 1.5)
dev.off()

##################### Figure 3.4 ########################
# fix sigma
sigma.fix = 0.5

pdf("22.pdf",width=10,height=5)
par(mfrow=c(1,2))
plot(beta, part_obj(beta,sigma.fix),type = "l",lwd=6, main="Part 2 ",
     cex.lab = 1.5, cex.main=2, cex.axis = 1.5, cex = 1.5, ylab = "f")

plot(beta, simple_obj(beta,sigma.fix),type = "l",lwd=6, main  = "RSS + Part2",
     cex.lab = 1.5, cex.main=2, cex.axis = 1.5, cex = 1.5, ylab="f")
dev.off()

##################### Figure 3.5 ########################
library(ggpubr)

# Generate two-dimension dataset
n <- 20 # sample size
p <- 2 # number of covariates (including the intercept)

# Sampling
set.seed(2019)
x_2 <- sample(1:n)
X <- cbind(rep(1,n), x_2)

# Standardization of design matrix
# NOte: we do not use "sd" function since in R the denominator is (n-1)
X_standard <- t((t(X) - c(0, mean(x_2)))/c(1, sqrt(sum((x_2-mean(x_2))^2) /(n)) ))

# Parameters of LPTL in this plot
rho_LPTL <- 0.95
tau_LPTL <- qlaplace((1 + rho_LPTL) / 2)
lambda_LPTL <- 2 * tau_LPTL * log(tau_LPTL) * dlaplace(tau_LPTL) /
  (1 - (plaplace(tau_LPTL) - plaplace(-tau_LPTL)))

### Left Plot: lambda = 1 ###
# Initialization
beta_new = seq(0.001, 1, 0.005)
lambda_pri <- 1 # lambda of Laplace
```

```
estimate_ols <- rep(NA, length(beta_new))
estimate_LPTL <- rep(NA, length(beta_new))
estimate_lasso <- rep(NA, length(beta_new))

for(i in 1:length(beta_new))
{
  # Create new observations using new beta
  set.seed(2019)
  y_new <- round(X_standard[,2]*beta_new[i] + rnorm(n, mean = 0, sd = 0.1), 2)
  ols_new <- lm(y_new ~ X_standard[,-1])

  MLE_Normal <- c(sqrt((n - p) / n) * sigma(ols_new), ols_new$coefficients)
  estimate_ols[i] = MLE_Normal[3]

  MAP_lasso <- find_MAP_Lasso(nb_start = 100,
                              initial = as.vector(round(MLE_Normal, 2)),
                              sd_initial =abs(round(MLE_Normal/10,2)), y = y_new,
                              X = X_standard, lambda_pri = lambda_pri)

  estimate_lasso[i] <- MAP_lasso[3]

  MAP_LPTL <- find_MAP_LPTL(nb_start = 100,
                            initial = as.vector(round(MLE_Normal, 2)),
                            sd_initial =abs(round(MLE_Normal/10,2)), y = y_new,
                            X = X_standard, lambda_pri = lambda_pri,
                            tau_LPTL = tau_LPTL, psi_LPTL = lambda_LPTL + 1)

  estimate_LPTL[i] <- MAP_LPTL[3]
}

# Construct a data frame to store the estimates of lasso and LPTL
dt.lasso_LPTL = data.frame(estimate_ols = estimate_ols, estimate_lasso = estimate_
    lasso,estimate_LPTL = estimate_LPTL)

### Right Plot: lambda = 2 ###
beta_new2 = seq(0.001, 2, 0.005)
lambda_pri = 2
estimate_ols2 <- rep(NA, length(beta_new))
estimate_LPTL2 <- rep(NA, length(beta_new))
estimate_lasso2 <- rep(NA, length(beta_new))

for(i in 1:length(beta_new2))
{
  # Create new observations using new beta
  set.seed(2019)
  y_new <- round(X_standard[,2]*beta_new2[i] + rnorm(n, mean = 0, sd = 0.1), 2)
  ols_new <- lm(y_new ~ X_standard[,-1])

  MLE_Normal <- c(sqrt((n - p) / n) * sigma(ols_new), ols_new$coefficients)
  estimate_ols2[i] = MLE_Normal[3]

  MAP_lasso <- find_MAP_Lasso(nb_start = 10,
                              initial = as.vector(round(MLE_Normal, 2)),
                              sd_initial =abs(round(MLE_Normal/10,2)), y = y_new,
                              X = X_standard, lambda_pri = lambda_pri)

  estimate_lasso2[i] <- MAP_lasso[3]
  MAP_LPTL <- find_MAP_LPTL(nb_start = 10,
                            initial = as.vector(round(MLE_Normal, 2)),
                            sd_initial =abs(round(MLE_Normal/10,2)), y = y_new,
                            X = X_standard, lambda_pri = lambda_pri,
                            tau_LPTL = tau_LPTL, psi_LPTL = lambda_LPTL + 1)

  estimate_LPTL2[i] <- MAP_LPTL[3]
}

dt.lasso_LPTL2 = data.frame(estimate_ols2 = estimate_ols2, estimate_lasso2 =
    estimate_lasso2,estimate_LPTL2 = estimate_LPTL2)

cairo_pdf("t1.pdf", width = 10, height = 5)

pc1 <- ggplot(data = dt.lasso_LPTL)+geom_point(aes(x = estimate_ols, y = estimate_
    lasso, color = "Lasso", shape = "Lasso"),size = 2)+geom_point(aes(x = estimate_
    ols, y = estimate_LPTL, color = "LPTL",shape="LPTL"), size = 2,alpha=0.6)+scale
    _color_manual(name="Model",values = c("Lasso" = 'red', "LPTL" = 'black'))+scale
    _shape_manual(name="Model",values = c("Lasso" = 1, "LPTL" = 8))+ylab("Estimates
     of beta")+ xlab("OLS Estimate")+ggtitle("\u03BB = 1")+theme_bw()+theme(axis.
    title.x = element_text(size = 16), axis.title.y = element_text(size = 16),axis.
    text = element_text(size = 14),plot.title = element_text(size = 20,hjust = 0.5)
    ,legend.text = element_text(size = 16), legend.title = element_text(size =15))

pc2 <- ggplot(data = dt.lasso_LPTL2)+geom_point(aes(x = estimate_ols2, y =
    estimate_lasso2,color = "Lasso", shape = "Lasso"),size = 3)+geom_point(aes(x =
    estimate_ols2, y = estimate_LPTL2, color = "LPTL",shape="LPTL"), size = 2,alpha
    =0.5)+scale_color_manual(name="Model",values = c("Lasso" = 'red', "LPTL" = '
    black'))+scale_shape_manual(name="Model",values = c("Lasso" = 1, "LPTL" = 8))+
    ylab("Estimates of beta")+ xlab("OLS Estimate")+ ggtitle("\u03BB = 2")+theme_bw
    ()+theme(axis.title.x = element_text(size = 16), axis.title.y = element_text(
    size = 16),axis.text = element_text(size = 14),plot.title = element_text(size =
     20,hjust = 0.5),legend.text = element_text(size = 16), legend.title = element_
    text(size =17))

ggarrange(pc1, pc2, ncol = 2, common.legend = TRUE, legend = "bottom")
dev.off()



######################## Chapter 4: Real Data Analysis ########################

#---------------------------Data Cleaning--------------------------------
# Load the data sets
salary <- read.csv("salary.csv", header = TRUE)
perform <- read.csv("perform2.csv", header = TRUE)

# Check duplicated rows by "Player" for the two data sets
n_dup <- data.frame(table(salary$Player))
salary[salary$Player %in% n_dup$Var1[n_dup$Freq > 1],]

n_dup2 <- data.frame(table(perform$Player))
perform[perform$Player %in% n_dup2$Var1[n_dup2$Freq > 1],]

# Remove duplicated rows by "Player"
salary <- salary[!duplicated(salary$Player), ]
perform <- perform[!duplicated(perform$Player), ]
```

```
# Merge Data
merge_data <- merge(perform ,salary, by="Player", sort = FALSE)

# Check duplicate for the merged data
n_dup3 <- data.frame(table(merge_data$Player))
merge_data[merge_data$Player %in% n_dup3$Var1[n_dup3$Freq > 1],]
# No duplicate rows anymore

# Delete columns of index, plyer's name and redundant "team" variable
data <- merge_data[,-c(1,2,5)]

# Switch variables position
Salary <- data$Salary
team <- data$Tm.y

data <- cbind(Salary, team, data)
data <- data[,-c(27,28)]

# Check missing values and drop them
anyNA(data)
sum(is.na(data)==TRUE) # 4 missing values

# Delete the rwos that consists missing values
data  = data[complete.cases(data),]
anyNA(data) # FLASE

#------------------Section 4.2: Exploratory Data Analysis----------------------
library(ggplot2)
library(beeswarm)
library(gplots)
library(ggExtra) # ggMarginal
library(corrplot) # Correlation Plot
library(dplyr)

# look through the new dataset
head(data)
str(data)
summary(data)
dim(data)

################ Figure 4.1:  Correlation plot ###################
Team <- as.numeric(data$team)
Position <- as.numeric(data$Pos)
data_cor <- cbind(data[,-c(2,3)], Team, Position)
corr <- cor(data_cor) # Calculate the correlation matrix

pdf("cor.pdf",width = 10, height = 10)
palette = colorRampPalette(c("darkblue", "white", "darkred")) (26)
corrplot(corr,method="color",col=palette,tl.col="black",tl.cex=1.3,cl.cex=1.3)
dev.off()

######### Figure 4.2: Salary distribuiton by Positions ###########

# Reduce 11 levels to 5 by selecting the main position of the player
levels(data$Pos)[levels(data$Pos)=="C-PF"] <- "C"
levels(data$Pos)[levels(data$Pos)=="PF-C"] <- "PF"
levels(data$Pos)[levels(data$Pos)=="PF-SF"] <- "PF"
levels(data$Pos)[levels(data$Pos)=="SF-SG"] <- "SF"
levels(data$Pos)[levels(data$Pos)=="SG-PF"] <- "SG"
levels(data$Pos)[levels(data$Pos)=="SG-SF"] <- "SG"

pdf('bee.pdf', width = 10, height = 10)
par(mar=c(5,5.2,3,1.65)+0.1)

palette2 = colorRampPalette(c("lightblue", "darkblue")) (5)
beeswarm(Salary ~ Pos, data = data,col = palette2,pch = 16,cex =1.3, yaxt = "n",
    ylab = "", xlab = "Position",cex.lab = 1.6, cex.axis = 1.5, spacing = 0.9)

ylimm=c("$0","$5M","$10M","$15M","$20M","$25M","$30M", "$35M")
axis(2,at = seq(0,35000000,length.out = 8),label =ylimm,las =1,cex.axis=1.3)

title(ylab="Salary", cex.lab=1.6,  line=3.7)
abline(h=seq(0, 35000000, length.out = 8), lty=2, col="darkgrey")
dev.off()

### Table 4.3: Average Salary and number of players of each position ###
summary(data$Pos)
# C   PF  PG  SF  SG
# 91  97  99  73 120

mean(data$Salary[data$Pos=="PG"])# 7200504
mean(data$Salary[data$Pos=="SF"])# 7418001
mean(data$Salary[data$Pos=="C"]) # 8809944
mean(data$Salary[data$Pos=="PF"])# 6942628
mean(data$Salary[data$Pos=="SG"])# 5907966

############### Figure 4.3: Each team's total salary #################

# Construct a data frame to store each team's salary
salary1 <- rep(NA, 30)
team_salary = data.frame("team" = levels(data$team), "salary" = salary1)

# Add players' salary up in the same team
for(i in seq_along(levels(data$team)))
{
  team_salary$salary[i] <- sum(data$Salary[data$team== team_salary$team[i]])
}

# Sort team accordding to the amount of salary
team_salary_sort <- arrange(team_salary, salary)
team_salary_sort$team<-factor(team_salary_sort$team,levels=team_salary_sort$team)

pdf('tm.pdf', width = 10, height = 10)
par(mar=c(5,4.8,3,1.65)+0.1)

palette3 = colorRampPalette(c("lightblue", "darkblue")) (30)
xlimm=c("$0","$50M","$100M","$150M")

ggplot(data=team_salary_sort,aes(x=team,y =salary))+geom_bar(stat="identity",fill
    = palette3)+coord_flip()+scale_y_continuous(breaks = seq(0, 150000000,length.
    out = 4),labels = xlimm)+geom_hline(aes(yintercept=123700000), linetype = "
    dashed",color ="red", size =1.2)+geom_hline(aes(yintercept=101870000),linetype
    ="dashed", color = "orange", size =1.2)+annotate("text", x=1.6, y
    =123700000+11000000, label="Luxury Tax", size =6)+annotate("text", x=1.6,y
    =101870000-10000000, label="Salary Cap", size=6)+geom_text(aes(x = 0,y
    =123700000+10000000, label ="$123.7M", vjust = -0.5),col="red",show.legend =
    FALSE,size=6)+geom_text(aes(x = 0,y =101870000-10000000,label="$101.9M",vjust =
     -0.5),col="orange",show.legend=FALSE,size=6)+xlab("Team")+ylab("Salary")+theme
    (axis.title.x=element_text(size = 18),axis.title.y =element_text(size=18),axis.
    text = element_text(size=15),plot.margin=unit(rep(0.7,4),'lines'))
dev.off()

############### Figure 4.4: Salary against PER #################
pdf("per.pdf", width = 10, height = 10)
xlimm2=c("$0","$10M","$20M","$30M")
palette4 = c("#3EBB00FF","darkorange","red","skyblue","blue")
scatterplot <- ggplot(data, aes(x = data$PER, y = data$Salary))+geom_point(aes(
    color = data$Pos), size = 2)+scale_color_manual("Position",values = palette4)+
    xlab("PER")+ylab("Salary")+scale_y_continuous(breaks = seq(0, 30000000,length.
    out = 4), labels = xlimm2)+geom_hline(aes(yintercept=30000000), linetype = "
    dashed", color = "purple", size =1)+geom_vline(aes(xintercept=20), linetype = "
    dashed", color = "#FF9FFFFF", size =1)+theme(axis.title.x = element_text(size =
     15), axis.title.y = element_text(size = 15),axis.text = element_text(size =
    14), plot.title = element_text(size = 20,hjust = 0.5),legend.text = element_
    text(size = 14), legend.title = element_text(size =15))ggMarginal(scatterplot,
    type = "histogram", fill ="#6D9EC1", color= "#BFD5E3")
dev.off()

############### Figure 4.5: Salary against Minutes played #################
xlimm2=c("$0","$10M","$20M","$30M")
palette4 = c("#3EBB00FF","darkorange","red","skyblue","blue")

pdf("mp.pdf",width = 10,height = 10)
scatterplot <- ggplot(data, aes(x = data$MP, y = data$Salary))+geom_point(aes(
    color = data$Pos), size = 2)+scale_color_manual("Position",values = palette4)+
    xlab("Minutes Played")+ylab("Salary")+scale_y_continuous(breaks = seq(0,
    30000000,length.out = 4), labels = xlimm2)+geom_hline(aes(yintercept=30000000),
     linetype = "dashed", color = "purple", size =1)+geom_vline(aes(xintercept
    =2000), linetype = "dashed", color = "#FF9FFFFF", size =1)+theme(axis.title.x =
     element_text(size = 15), axis.title.y = element_text(size = 15),axis.text =
    element_text(size = 14), plot.title = element_text(size = 20,hjust = 0.5),
    legend.text = element_text(size = 14), legend.title = element_text(size =15))
ggMarginal(scatterplot, type = "histogram", fill = "#6D9EC1", color= "#BFD5E3")
dev.off()

#--------------------------Data Preprocessing-------------------------------

### Construct Dummy variables ###

# One dummy variable about salary
# Set "1" if the team's total salary is larger than the "Salary Cap"

# Two dummy variables about positions
# One set value "1" for position "C", the other set "1" for position "SG"
# If both of them are 0, then its postion is "PG", "PF" or "SF"

Team.d <- rep(0, dim(data)[1])
pos.C <- rep(0, dim(data)[1])
pos.SG <- rep(0, dim(data)[1])
data <- cbind(data, Team.d, pos.C, pos.SG)
```

```r
data$Team.d[team_salary$salary[data$team] > 101869000] = 1
data$pos.C[data$Pos =="C"] = 1
data$pos.SG[data$Pos=="SG"] = 1

### Drop useless variables and correlated variables (X3PAr, TRB, WS, BPM, VORP)
drops <- c("team","Pos","X3PAr","TRB.","WS","BPM","VORP")
data <- data[ , !(names(data) %in% drops)]

# Store dimention of dataset
n = length(data[,1]);n
p = length(data[1,-1])+1;p

# Standardised the whole dataset(including response and design matrix)
data_centered = apply(data, 2, function(x) x-mean(x))
data_standard = apply(data_centered, 2, function(x) x/sqrt(sum(x^2)/n))
dim(data_standard)

# Add first column in the desine matrix
X_std = cbind(1, data_standard[,2:dim(data_standard)[2]])

# AS a data frame
data_standard = as.data.frame(data_standard)

################### Linear Regression: OLS estimates ####################
# Using OLS estimate as the initial values for our algorithms

# Fit Linear Regression
set.seed(2019)
ols_nba = lm(data_standard$Salary ~. , data = data_standard)
summary(ols_nba)
# R square: 0.5479

# OLS estimates
MLE_nba <- c(sqrt((n - p) / n) * sigma(ols_nba), ols_nba$coefficients)
round(as.vector(MLE_nba[-1]), 3)
#  0.000  0.327 -0.487  0.797  0.516 -0.287  0.082 -0.136  0.045 -0.133 -0.050
# -0.055  0.136 -0.007  0.002  0.007  0.018 -0.038  0.019  0.064  0.016 -0.009

# Set initial values for optimx function
# The estimate of intercept of OLS is so small that cause error in optimx
# We reset it to a samll value nearly 0 for each method
sigma_k =  as.vector(MLE_nba)[1]
betahat_k = as.vector(MLE_nba)[-1]
betahat_k[1] = 0.0001
parameters_k <- c(sigma_k, betahat_k)

############# Performance Evaluation for competitors and our models #############
library(rmutil)
library(leaps)
library(caret)
library(parallel)
library(optimx)

### Setting for (nested) cross-validation ###
nfolds_outer = 5 # Number of outer loops
nfolds_inner = 10 # Number of inner loops
nfolds = 10 # Number of folds for corss-vaidating on the whole dataset
```

```
# Create outer folds
set.seed(2019)
folds = createFolds(data_standard$Salary, k = nfolds_outer)

#--------------------------Best Subset Selection-------------------------------
result_subset <- sapply(seq_along(folds), outerloop.subset)
result_subset
# MSE               0.525145 0.4182759  0.5185936 0.5711661 0.5545292
# Number of var.:  12.000000 8.0000000 10.0000000 9.0000000 9.0000000
mean(result_subset["mse_subset",])
# 0.517542

# Fit on the whole dataset (as the traininig data) to get the best fitted model
data_std_train_nba = cbind(1, data_standard[,2:dim(data_standard)[2]])
set.seed(2019)
subset_best_nvar = tune.subset_model(train_nba = data_standard,
                                     X_std_train_nba = data_std_train_nba,
                                     nfolds_inner = nfolds_inner)

subset_best = regsubsets (Salary ~. ,data=data_standard,nvmax=subset_best_nvar)

# Best model of subset selection
coef_subset = coef(subset_best, subset_best_nvar)
round(coef_subset,3)
# (Intercept)  Age      G      MP     PER     TS.     FTr     TOV.  Team.d
#   0.000     0.336  -0.492  0.831  0.301  -0.166   0.094   0.060   0.064

# Number of variables for best mode
subset_best_nvar # 8

#-----------------------Ridge Regression (optimx)---------------------------
# lambda sequence (path) for ridge
alpha = 1; K = 100
lambda_max = max(abs(colSums(as.matrix(data_standard[,2:p]) *
                             as.vector(data_standard[,1]))))/(alpha*n)
lambda_min = 0.0005*lambda_max
lambda_path_ridge<- round(exp(seq(log(lambda_max), log(lambda_min),
                                  length.out = K)), digits = 10)

c1 = makeCluster(5) # Create nodes
mse_ridge.x = rep(NA, nfolds_outer)

result_ridge.x <- sapply(seq_along(folds), outerloop.ridge.x)
result_ridge.x
# mse_ridge 0.518217709 0.409694669 0.515087328 0.551591420 0.547075774
# lambda    0.004766676 0.005147062 0.005557803 0.008158716 0.006997362

mean(result_ridge.x['mse_ridge.x',])
# 0.5083334

stopCluster(c1)

## Fit on the whole dataset (as the traininig data) to get the best fitted model
c1 = makeCluster(5)

data_std_train_nba = cbind(1, data_standard[,2:dim(data_standard)[2]])
```

```
Ridge_best_model.x = tune.Ridge_model.para.x(train_nba = data_standard,nfolds_
    inner = nfolds,parameters_k = parameters_k,X_std_train_nba = data_std_train_nba
    ,cluster = c1, lambda = lambda_path_ridge)
stopCluster(c1)

coefs.best.ridge.x = as.vector(Ridge_best_model.x[c(-1,-2)])
round(coefs.best.ridge.x, 3)
#  0.000  0.327 -0.473  0.784  0.499 -0.276  0.082 -0.130  0.047 -0.124 -0.047
    -0.053  0.130 -0.002  0.002  0.008  0.017 -0.037  0.019  0.064  0.015 -0.008
ridge.best.lambda.x = Ridge_best_model.x[1]

#-------------------------Lasso regression (optimx)---------------------------
# lambda sequence (path) for lasso
alpha = 1; K = 100
lambda_max = max(abs(colSums(as.matrix(data_standard[,2:p]) *
                              as.vector(data_standard[,1]))))/(alpha*n)
lambda_min = 0.001*lambda_max
lambda_path_lasso<- round(exp(seq(log(lambda_max), log(lambda_min),
                                 length.out = K)), digits = 10)

c1 = makeCluster(5)  # Create nodes
mse_lasso.x = rep(NA, nfolds_outer)

result_lasso.x <- sapply(seq_along(folds_outer), outerloop.lasso.x)
result_lasso.x
# mse_lasso   0.506471809 0.40482374 0.5210219 0.55539124 0.55925404
# lambda      0.009007574 0.02080867 0.0146801 0.01687855 0.02392491

mean(result_lasso.x['mse_lasso.x',])  # 0.5093925
stopCluster(c1)

# Fit on the whole dataset to get the best fitted model
c1 = makeCluster(5)
data_std_train_nba = cbind(1, data_standard[,2:dim(data_standard)[2]])

Lasso_best_model.x = tune.Lasso_model.para.x(train_nba = data_standard, cluster =
    c1,X_std_train_nba = data_std_train_nba,parameters_k = parameters_k, nfolds_
    inner = nfolds_inner,lambda = lambda_path_lasso)
stopCluster(c1)

coefs.lasso.x = as.vector(Lasso_best_model.x[c(-1,-2)])
round(coefs.lasso.x,3)
# 0.000  0.321 -0.380  0.712  0.248 -0.111  0.075 -0.020  0.060  0.000  0.000
# -0.012 0.040  0.051  0.000  0.000  0.000 -0.006  0.009  0.054  0.000  0.000
lasso.best.lambda.x = Lasso_best_model.x[1]

#-------------------------Elastic Net (optimx)---------------------------
# lambda and alpha sequence (path) for elastic net
alpha_path = seq(0.1, 0.9, by = 0.1)
alpha = 1; K = 100
lambda_max = max(abs(colSums(as.matrix(data_standard[,2:p]) *
                            as.vector(data_standard[,1]))))/(alpha*n)
lambda_min = 0.001*lambda_max

lambda_path_elastic.net <- round(exp(seq(log(lambda_max), log(lambda_min),
                                length.out = K)), digits = 10)
```

```
c1 = makeCluster(5)  # create nodes

result_elastic.net.x <- sapply(seq_along(folds), outerloop.elastic.net.x)
result_elastic.net.x

# mse_elastic.net   0.506710558 0.40449671 0.52098331 0.55516983 0.56077061
# lambda            0.009658527 0.02231245 0.01574099 0.01809832 0.02565389
# alpha             0.100000000 0.10000000 0.10000000 0.10000000 0.10000000

mean(result_elastic.net.x["mse_elastic.net.x",]) # 0.5096262
stopCluster(c1)

# Fit on the whole dataset to get the best fitted model
c1 = makeCluster(5)

data_std_train_nba = cbind(1, data_standard[,2:dim(data_standard)[2]])

Elastic.net_best_model.x = tune.Elastic.net_model.para.x(train_nba = data_standard
    , cluster = c1,X_std_train_nba = data_std_train_nba,nfolds_inner = nfolds_inner
    ,parameters_k = parameters_k,lambda = lambda_path_elastic.net,alpha = alpha_
    path)
stopCluster(c1)

coefs.best.elastic.net.x = as.vector(Elastic.net_best_model.x[c(-1,-2,-3)])
round(coefs.best.elastic.net.x, 3)
#  0.000  0.325 -0.437  0.756  0.387 -0.202  0.081 -0.076  0.058 -0.058 -0.029
# -0.033  0.087  0.016  0.000  0.005  0.000 -0.017  0.019  0.060  0.000  0.000
elastic.net.best.lambda.x = Elastic.net_best_model.x[1]
alpha.x = Elastic.net_best_model.x[2]

#----------------------------- LPTL model (optimx) ------------------------------
# lambda sequence (path) for LPTL
lambda_path_LPTL <- seq(1, 40, length.out = 100)

# best parameters of LPTL distribution
rho_LPTL <- 0.97
tau_LPTL <- qlaplace((1 + rho_LPTL) / 2)
lambda_LPTL <- 2 * tau_LPTL * log(tau_LPTL) * dlaplace(tau_LPTL) / (1 - (plaplace(
    tau_LPTL) - plaplace(-tau_LPTL)))

c1 = makeCluster(5) # create nodes
mse_normal_LPTL.x = rep(NA, nfolds_outer)

result_normal_LPTL.x <- sapply(seq_along(folds), outerloop.nomal_LPTL.x)
result_normal_LPTL.x

mean(result_normal_LPTL.x['mse_normal_LPTL.x',])
# best: rho 0.97
# mse_normal_LPTL.x   0.4616803  0.3810919  0.4729345  0.5273531  0.5004357
# lambda_best         34.0909091 22.2727273 37.2424242 34.0909091 21.0909091
# mean: 0.4686991

# rho 0.98
# mse_normal_LPTL.x   0.4572499  0.3771393  0.4761521  0.5373029  0.5545153
# lambda_best         38.0303030 38.8181818 34.8787879 20.3030303 28.9696970
# 0.4804719
# rho 0.96
# mse_normal_LPTL.x   0.4629084  0.3792438  0.4726643  0.5391966  0.5002248
# lambda_best         19.9090909 26.2121212 30.9393939 29.7575758 16.3636364
# 0.4708476

# rho 0.95
# mse_normal_LPTL.x   0.4678269 0.387900   0.502912   0.5314451   0.5003999
# lambda_best         19.1212121 5.333333  17.151515  26.6060606  24.6363636
# 0.4780968

# rho 0.94
# mse_normal_LPTL.x   0.4721757 0.3879777  0.4983558  0.5341039  0.5122942
# lambda_best         15.1818182 5.3333333 19.1212121 17.9393939 23.0606061
# 0.4809815

# rho 0.93
# mse_normal_LPTL.x   0.4632727 0.3961622  0.4877376  0.5315637  0.5363541
# lambda_best         10.4545455 9.2727273 12.8181818 31.3333333 38.0303030
# 0.483018
stopCluster(c1)

# Fit on the whole dataset to get the best fitted model
c1 = makeCluster(5)
data_std_train_nba = cbind(1, data_standard[,2:dim(data_standard)[2]])

Normal_LTPL_best_model.x = tune.Normal_LPTL_model.para.x(train_nba = data_standard
    , X_std_train_nba = data_std_train_nba, parameters_k = parameters_k,cluster =
    c1, nfolds_inner = nfolds_inner,tau_LPTL = tau_LPTL,lambda_LPTL = lambda_LPTL,
    lambda = lambda_path_LPTL)
stopCluster(c1)

coefs.Normal_LPTL.x = as.vector(Normal_LTPL_best_model.x[c(-1,-2)])
round(coefs.Normal_LPTL.x,3)
#  0.000  0.327 -0.486  0.796  0.516 -0.286  0.080 -0.134  0.040 -0.131 -0.046
# -0.050  0.135  0.000  0.000  0.000  0.000 -0.030 -0.002  0.060  0.000  0.000

best.lambda_Normal_LPTL = Normal_LTPL_best_model.x[1] # 24.63636

#----------------------------LPTL + LPTN Model (optimx)------------------------
# lambda sequence (path) of LPTL+LPTN is the same as LPTL
lambda_path_LPTL <- seq(1, 40, length.out = 100)

# best LPTN parameters
rho_LPTN <- 0.95
tau_LPTN <- qnorm((1 + rho_LPTN) / 2)
lambda_LPTN <- 2 * tau_LPTN * log(tau_LPTN) * dnorm(tau_LPTN) /
                    (1 - (pnorm(tau_LPTN) - pnorm(-tau_LPTN)))

# best LPTL parameters
rho_LPTL <- 0.97
tau_LPTL <- qlaplace((1 + rho_LPTL) / 2)
lambda_LPTL <- 2 * tau_LPTL * log(tau_LPTL) * dlaplace(tau_LPTL) /
                  (1 - (plaplace(tau_LPTL) - plaplace(-tau_LPTL)))

# Make the initial value of beta_1 0.0005, otherwise it's too small
sigma_k =  as.vector(MLE_nba)[1]
betahat_k = as.vector(MLE_nba)[-1]
```

```
betahat_k[1] = 0.0005
parameters_k <- c(sigma_k, betahat_k)

c1 = makeCluster(5)
mse_LPTN_LPTL.x = rep(NA, nfolds_outer)

result_LPTN_LPTL.x <- sapply(seq_along(folds), outerloop.LPTN_LPTL.x)
result_LPTN_LPTL.x

mean(result_LPTN_LPTL.x['mse_LPTN_LPTL.x',])
# best (0.97, 0.95) lambda 1 to 40
# mse_LPTN_LPTL.x   0.4647805  0.3799267  0.4769368  0.5272415  0.4946309
# lambda_best       39.6060606 38.0303030 35.6666667 29.7575758 36.0606061
# 0.4687033

# (0.97, 0.98 LPTN)
# mse_LPTN_LPTL.x   0.463062  0.3770966  0.4762955  0.581894  0.494822
# lambda_best       39.606061 31.7272727 32.1212121 32.909091 35.666667
# 0.478634

# (0.97, 0.97 LPTN)
# mse_LPTN_LPTL.x   0.4652953  0.3908232  0.4754009  0.5401288  0.496362
# lambda_best       39.2121212 37.2424242 34.4848485 34.4848485 18.727273
# 0.473602

# 0.97 0.96
# mse_LPTN_LPTL.x   0.4772344  0.3731962  0.5022402  0.5247989  0.4943899
# lambda_best       34.4848485 40.0000000 13.2121212 19.1212121 39.2121212
# 0.4743719

# 0.97 0.94
# mse_LPTN_LPTL.x   0.472948  0.3724078  0.4904857  0.5271235  0.5109561
# lambda_best       38.424242 38.4242424 38.4242424 36.4545455 39.6060606
# 0.4747842

# 0.97, 0.93
# mse_LPTN_LPTL.x   0.4911841  0.3896591  0.4762046  0.5269822  0.4945186
# lambda_best       34.0909091 23.8484848 23.0606061 26.2121212 30.5454545
# 0.4757097
stopCluster(c1)

## Fit on the whole dataset (as the traininig data) to get the best fitted model
c1 = makeCluster(5)
data_std_train_nba = cbind(1, data_standard[,2:dim(data_standard)[2]])

LPTN_LTPL_best_model.x = tune.LPTN_LPTL_model.para.x(train_nba = data_standard, X_
   std_train_nba = data_std_train_nba,parameters_k = parameters_k, cluster = c1,
   nfolds_inner = nfolds_inner,tau_LPTL = tau_LPTL,lambda_LPTL = lambda_LPTL,
   lambda = lambda_path_LPTL,tau_LPTN = tau_LPTN,lambda_LPTN = lambda_LPTN)
stopCluster(c1)

coefs.LPTN_LPTL.x = as.vector(LPTN_LTPL_best_model.x[c(-1,-2)])
round(coefs.LPTN_LPTL.x,3)
#  0.000  0.326 -0.486  0.796  0.515 -0.287  0.081 -0.132  0.040 -0.132 -0.046
# -0.048  0.135  0.018  0.000  0.000  0.000 -0.031  0.000  0.059  0.000  0.000

best.lambda_LPTN_LPTL = LPTN_LTPL_best_model.x[1] # 25.42424
```

```
sigma_LPTL_LPTN = LPTN_LTPL_best_model.x[2] # 0.6825473

#---------------Figures in Section 4.6: Results of Analysis-----------------------
library(ggplot2)
library(ggvis)
library(grid)
library(ggpubr) # For ggarange

##################### Figure 4.6 #######################
# Plot of performance(MSE) of methods on each fold
Fold  = c("Fold 1", "Fold 2", "Fold 3", "Fold 4", "Fold 5")

perform <- data.frame(Fold = c(1,2,3,4,5), subset.p = result_subset[1,],
                      ridge.p = result_ridge.x[1,],
                      lasso.p = result_lasso.x[1,],
                      elastic.net.p = result_elastic.net.x[1,],
                      lptl.p = result_normal_LPTL.x[1,],
                      lptl.lptn.p = result_LPTN_LPTL.x[1,])

pdf("perform.pdf",width = 10, height = 6)
ggplot(data = perform)+geom_line(aes(x = Fold, y = subset.p, color = "Subset
   Selection"),size =1)+geom_point(aes(x = Fold, y = subset.p, shape = "Subset
   Selection", fill = "Subset Selection"), size = 3)+geom_line(aes(x = Fold, y =
   ridge.p, color = "Ridge"),size =1)+geom_point(aes(x = Fold, y = ridge.p, shape
   = "Ridge", fill = "Ridge"), size = 3)+geom_line(aes(x= Fold, y = lasso.p, color
    = "Lasso"))+geom_point(aes(x= Fold, y = lasso.p, shape = "Lasso", fill = "
   Lasso"), size = 3) +geom_line(aes(x = Fold, y = elastic.net.p, color = "Elastic
    Net"),size =1)+geom_point(aes(x = Fold, y = elastic.net.p, shape = "Elastic
   Net", fill = "Elastic Net"), size = 3)+geom_line(aes(x = Fold, y = lptl.p,
   color = "LPTL"),size =1)+geom_point(aes(x = Fold, y = lptl.p, shape = "LPTL",
   fill = "LPTL"), size = 3)+geom_line(aes(x = Fold, y = lptl.lptn.p, color = "
   LPTL+LPTN"),size =1)+geom_point(aes(x = Fold, y = lptl.lptn.p, shape = "LPTL+
   LPTN", fill = "LPTL+LPTN"), size = 3)+ ylab("MSE")+xlim( c("Fold 1", "Fold 2",
   "Fold 3", "Fold 4", "Fold 5"))+scale_shape_manual(name = "Method", values = c('
   Subset Selection'= 22,'Ridge' = 25,'Lasso' = 24,  "Elastic Net"=1, "LPTL" = 21,
    'LPTL+LPTN' = 23))+scale_color_manual(name = "Method", values = c('Subset
   Selection'= 'green','Ridge' = 'orange','Lasso' = 'purple', "Elastic Net" = "
   blue",'LPTL' = 'darkturquoise','LPTL+LPTN' = 'pink'))+scale_fill_manual(name =
   "Method", values = c('Subset Selection' = 'darkgreen','Ridge' = 'orange','Lasso
   ' = 'red', "Elastic Net" = "white",'LPTL' = 'darkblue','LPTL+LPTN' = 'yellow'))
   +theme(axis.title.x = element_text(size = 16), axis.title.y = element_text(size
    = 16),axis.text = element_text(size = 14),legend.position = c(0.886,0.226),
   legend.text = element_text(size = 14), legend.title = element_text(size =16),
   legend.background = element_rect(fill="lightskyblue1",size=0.5, linetype="solid
   ", colour ="darkblue"),legend.key.size = unit(1.6,"line"))
dev.off()

##################### Figure 4.7 #######################
# store the estimates of subset selection
coef_subset_plot = c(0, 0.336, -0.492,0.831,0.301, -0.166,
                     0.094,0,0,0,0,0,0.060,0,0,0,0,0,0,0.064,0,0)

# Construct a data frame storing all the best estimates of methods
estimates = data.frame(variables = seq(1,22,by=1),
                       subset = coef_subset_plot,
                       ridge = coefs.best.ridge.x,
                       lasso =coefs.lasso.x,
                        els = coefs.best.elastic.net.x,
                        lptl.lptn = coefs.LPTN_LPTL.x,
                        lptl = coefs.Normal_LPTL.x,
                        ols = as.vector(MLE_nba[-1]))

pdf("estimates.pdf", width = 10, height = 10)

# Top plot: comparison between best lasso estiamtes and OLS estimates
e1 <- ggplot(data = estimates)+ geom_hline(aes(yintercept=0), linetype="dashed",
   color = "black", size =1.2,alpha=0.5)+geom_line(aes(x = variables, y = ols,
   color = "OLS"),size = 1)+geom_point(aes(x = variables, y = ols, shape = "OLS",
   fill = "OLS"), size=3)+geom_line(aes(x = variables, y = lasso, color = "Lasso")
   ,size=1)+geom_point(aes(x = variables, y = lasso, shape = "Lasso", fill ="Lasso
   "), size=3)+ylab("Estimates")+xlab("Parameters")+scale_y_continuous(breaks=seq
   (-0.5, 0.8, by=0.25))+scale_shape_manual(name = "Method",values = c('Lasso' =
   24, "OLS" = 21))+scale_color_manual(name = "Method",values = c('Lasso' = '
   darkorange', 'OLS' = 'green3'))+scale_fill_manual(name = "Method",values = c('
   Lasso' = 'yellow','OLS' = 'darkgreen'))+theme_bw()+theme(axis.title.x = element
   _text(size = 15), axis.title.y = element_text(size = 15),axis.text = element_
   text(size = 13),legend.position = c(0.94,0.87),legend.text = element_text(size
   = 14),legend.background = element_rect(fill="lightskyblue1",size=0.5, linetype=
   "solid", colour ="darkblue"),legend.title = element_text(size =16),legend.key.
   size = unit(1.6,"line"))

# Bottom plot: comparison between best LPTL estimates and OLS estimates
e2 <- ggplot(data = estimates)+geom_hline(aes(yintercept=0), linetype="dashed",
   color = "black", size =1.2,alpha=0.5)+geom_line(aes(x = variables, y = ols,
   color = "OLS"),size=1)+geom_point(aes(x = variables, y = ols, shape = "OLS",
   fill = "OLS"), size=3)+geom_line(aes(x = variables, y = lptl, color = "LPTL"),
   size=1)+geom_point(aes(x = variables, y = lptl, shape = "LPTL", fill = "LPTL"),
    size=3)+scale_shape_manual(name = "Method",values = c("OLS" = 21, 'LPTL' = 25)
   )+scale_color_manual(name = "Method",values = c('OLS' = 'green3','LPTL' = '
   skyblue'))+scale_fill_manual(name = "Method",values = c('OLS' = 'darkgreen','
   LPTL' = 'mediumblue'))+ylab("Estimates")+xlab("Parameters")+scale_y_continuous(
   breaks=seq(-0.5, 0.8, by=0.25))+theme_bw()+theme(axis.title.x = element_text(
   size = 15), axis.title.y = element_text(size = 15),axis.text = element_text(
   size = 13),legend.position = c(0.94,0.87),legend.text = element_text(size = 14)
   , legend.background = element_rect(fill="lightskyblue1",size=0.5, linetype="
   solid", colour ="darkblue"),legend.title = element_text(size =16),legend.key.
   size = unit(1.6,"line"))

ggarrange(e1,e2, nrow = 2)
dev.off()

##################### Figure 4.8 ########################
# Construct a dataframe for the result (Lasso)
lasso_mse <- data.frame(MSE =  result_lasso_cv.x['mean_mse',],
                        Lambda = log(result_lasso_cv.x['lambda_cur',]))
min.lambda.lasso =  log(lasso.best.lambda.x)

# Construct a dataframe for the result (LPTL)
LPTL_Normal_mse <- data.frame(MSE = result.Normal_LPTL.x.cv['mean_mse',],
                              Lambda =log(result.Normal_LPTL.x.cv['lambda_cur',]))
min.lambda.lptl =  log(best.lambda_Normal_LPTL)

pdf("effect.pdf",width = 10, height = 5)
# Left plot of  MSE of Best Lasso model against log(Lambda)
```

```
a1 <- ggplot(data = lasso_mse, aes(x = Lambda, y = MSE))+geom_line(size =1, color
    = "skyblue")+geom_point(color = "darkblue",size = 1.8)+xlab("Log(Lambda)")+
    ggtitle("Lasso")+geom_vline(aes(xintercept=min.lambda.lasso), linetype = "
    dashed", color = "darkorange", size =1)+theme_bw()+theme(axis.title.x = element
    _text(size = 15), axis.title.y = element_text(size = 15),axis.text = element_
    text(size = 13),plot.title = element_text(size = 30,hjust = 0.5),plot.margin=
    unit(rep(0.8,4),'lines'))+annotation_custom(textGrob("-3.8", gp = gpar(col = "
    red", fontsize= 14)), xmin=min.lambda.lasso, xmax=min.lambda.lasso,ymin=0.492,
    ymax=0.492)+annotation_custom(segmentsGrob(gp = gpar(col = "red", lwd = 2)),
    xmin=min.lambda.lasso, xmax=min.lambda.lasso,ymin=0.492, ymax=0.492)+coord_
    cartesian(ylim=c(0.5,0.8), clip="off")

# Right plot of MSE of Best LPTL model against log(Lambda)
a2 <- ggplot(data = LPTL_Normal_mse, aes(x = Lambda, y = MSE))+geom_line(size =1,
    color = "skyblue")+geom_point(color = "darkblue",size = 2)+xlab("Log(Lambda)")+
    ggtitle("LPTL")+geom_vline(aes(xintercept=min.lambda.lptl), linetype = "dashed"
    , color = "darkorange", size =1)+theme_bw()+theme(axis.title.x = element_text(
    size = 15), axis.title.y = element_text(size = 15),axis.text = element_text(
    size = 13),plot.title = element_text(size = 30,hjust = 0.5),plot.margin=unit(
    rep(0.8,4),'lines'))+annotation_custom(textGrob("3.2", gp = gpar(col = "red",
    fontsize= 14)), xmin=min.lambda.lptl, xmax=min.lambda.lptl,ymin=0.44, ymax
    =0.44) +annotation_custom(segmentsGrob(gp = gpar(col = "red", lwd = 2)), xmin=
    min.lambda.lptl, xmax=min.lambda.lptl,ymin=0.44, ymax=0.44)+coord_cartesian(
    ylim=c(0.45,0.6), clip="off")

ggarrange(a1,a2, ncol = 2)
dev.off()

##################### Figure 4.9 ########################
# From the above results
mse.lptl = c(0.483018, 0.4809815, 0.4780968, 0.4708476, 0.4686991, 0.4804719)

tune.rho.lptl<-data.frame(rho.lptl=seq(0.93,0.98,by=0.01),mse.lptl=mse.lptl)

tune.rho.lptl.lptn = data.frame(rho.lptn = seq(0.93, 0.98, by=0.01),mse.lptl.lptn
    = c(0.4757097, 0.4747842,0.4678536,0.4743719,0.473602, 0.478634))

cairo_pdf("rho.pdf", width = 10, height = 5)
r1<-ggplot(data = tune.rho.lptl, aes(x =rho.lptl, y = mse.lptl))+geom_line(color =
     "darkturquoise")+geom_point(size= 3, color = "darkblue")+scale_x_continuous(
    breaks=seq(0.93, 0.98, by=0.01))+ylab("Average of MSE")+xlab("\u03C1")+ggtitle(
    "Tune \u03C1 for LPTL")+theme(axis.title.x = element_text(size = 15), axis.
    title.y = element_text(size = 15),axis.text = element_text(size = 13),plot.
    title = element_text(size = 25,hjust = 0.5))

r2<-ggplot(data = tune.rho.lptl.lptn, aes(x =rho.lptn, y = mse.lptl.lptn))+geom_
    line(color = "darkturquoise")+geom_point(size= 3, color = "darkblue")+scale_x_
    continuous(breaks=seq(0.93,0.98, by=0.01))+ylab("Average of MSE")+xlab("\u03C1"
    )+ggtitle("Tune \u03C1 for LPTN")+theme(axis.title.x = element_text(size = 15),
     axis.title.y = element_text(size = 15),axis.text = element_text(size = 13),
    plot.title = element_text(size = 25,hjust = 0.5))

ggarrange(r1,r2,ncol =2)
dev.off()

#-------------Outlier identification using LPTL+LPTN Model-----------------
# Using the estimates from LPTL+LPTN Model
```

```
Index =  seq(1,480,by=1)
se = as.vector((data_standard[,1] - X_std%*%coefs.LPTN_LPTL.x)/LPTN_LTPL_best_
    model.x[2])

# Potential outliers
outlier = which(abs(se)>2.5)
# 54 57 64 132 166 362 396 415

# Potential outliers with errors > 3
outlier2 = which(abs(se)>3)
# 57 64 166 362 415

# Look at those with errors > 3 in the original dataset
# Find that the salary (response) of all of them are quite high
data[outlier2,]

##################### Figure 4.10 ########################
pdf("outlier.pdf", width = 10, height = 10)

ggplot(data=NULL, aes(Index,se))+geom_hline(aes(yintercept=0), linetype = "dashed"
    , color = "darkorange", size =1)+geom_point(color= "blue", size = 2.5)+ scale_y
    _continuous(breaks=seq(-2.5, 3, by=1))+ylab("Standardised Errors")+geom_hline(
    aes(yintercept=2.5), linetype = "dashed", color = "purple", size =1)+geom_hline
    (aes(yintercept=-2.5), linetype = "dashed", color = "purple", size =1)+geom_
    point(aes(x = outlier, y = se[outlier]),shape = 16, size = 2.5, color = "red")+
    theme_bw()+theme(axis.title.x = element_text(size = 18), axis.title.y = element
    _text(size = 18),axis.text = element_text(size = 15),panel.background = element
    _rect(fill = 'azure1'),plot.margin=unit(rep(0.8,4),'lines'))
dev.off()
```

R code of Additional Figures in Appendix

```
##################### Figure A.1 ########################
# 3-D plot of negative objective funtion, lambda=50
pdf("3d2.pdf",width = 10,height = 8)
persp(beta, sigma, z, main="Objective Function, lambda=50",
      col="lightblue", theta=30, phi=20, r=50, d=0.1, expand=0.5,
      ltheta=90, lphi=180, shade=0.75, ticktype="detailed",nticks=5,
      zlab = " f", cex.lab = 1.5, cex.main=2, cex.axis = 1.5, cex = 1.5)
dev.off()
##################### Figure A.2 ########################
# 3-D plot of negative objective funtion, lambda=100
pdf("3d3.pdf",width = 10,height=8)
persp(beta, sigma, z_neg, main="Negative Objective Function, lambda=100",
      col="lightblue", theta=30, phi=20, r=50, d=0.1, expand=0.5,
      ltheta=90, lphi=180, shade=0.75, ticktype="detailed",nticks=5,
      zlab = " f", cex.lab = 1.5, cex.main=2, cex.axis = 1.5, cex = 1.5)
dev.off()

##################### Figure A.3 ########################
# New data correlation plot
pdf("cor2.pdf",width = 10,height=10)
corr2 <- cor(data)
palette = colorRampPalette(c("darkblue", "white", "darkred")) (22)
corrplot(corr2,method="color",col = palette,tl.col="black",tl.cex=1.3,cl.cex=1.3)
dev.off()
```

```
##################### Figure A.4 ########################
# Construct a dataframe for the result of subset selection
subset_MSE <- as.data.frame(subset_MSE)
names(subset_MSE) <- "MSE"
index <- c(1:nrow(subset_MSE))
subset_MSE<- cbind.data.frame(index,subset_MSE)

# Construct a dataframe for the result of ridge
MSE.ridge <- result_ridge_cv.x['mean_mse',]
lambda.ridge <- result_ridge_cv.x['lambda_cur',]
ridge.mse <- data.frame(MSE = MSE.ridge, Lambda = log(lambda.ridge))
min.lambda.ridge =  log(ridge.best.lambda.x)

# Left plot of MSE of Best subset selection against number of variables
pdf("app1.pdf", width = 10, height=5)
lg1 <- ggplot(data = subset_MSE, aes(x = index, y = MSE))+geom_line(size =1, color
     = "skyblue")+geom_point(color = "darkblue",size = 3)+xlab("Number of Variables
    ")+ggtitle("Subset Selection")+theme_bw()+geom_vline(aes(xintercept=subset_best
    _nvar), linetype = "dashed", color = "darkorange", size =1)+theme(axis.title.x
    = element_text(size = 15), axis.title.y = element_text(size = 15),axis.text =
    element_text(size = 13),plot.title = element_text(size = 30,hjust = 0.5))+
    annotation_custom(textGrob("8", gp = gpar(col = "red", fontsize= 15)), xmin=8,
    xmax=8,ymin=0.48, ymax=0.48)+annotation_custom(segmentsGrob(gp = gpar(col = "
    red", lwd = 3)), xmin=8, xmax=8,ymin=0.48, ymax=0.48)+coord_cartesian(ylim=c
    (0.5,0.8), clip="off")

# Right plot of MSE of Best subset selection against log(Lambda)
lg2 <- ggplot(data = ridge.mse, aes(x = Lambda, y = MSE))+geom_line(size =1, color
     = "skyblue")+geom_point(color = "darkblue",size = 2)+xlab("Log(Lambda)")+
    ggtitle("Ridge")+geom_vline(aes(xintercept=min.lambda.ridge), linetype = "
    dashed", color = "darkorange", size =1)+theme_bw()+theme(axis.title.x = element
    _text(size = 15), axis.title.y = element_text(size = 15),axis.text = element_
    text(size = 13),plot.title = element_text(size = 30,hjust = 0.5))+annotation_
    custom(textGrob("-6.2", gp = gpar(col = "red", fontsize= 14)),xmin=min.lambda.
    ridge, xmax=min.lambda.ridge,ymin=0.501, ymax=0.501) +annotation_custom(
    segmentsGrob(gp = gpar(col = "red", lwd = 2)), xmin=min.lambda.ridge, xmax=min.
    lambda.ridge,ymin=0.501, ymax=0.501)+coord_cartesian(ylim=c(0.5,0.65), clip="
    off")

ggarrange(lg1,lg2,ncol=2)
dev.off()

##################### Figure A.5 ########################
# Construct a dataframe for the result of elastic net
MSE.EN1 <- result_elastic.net_cv.x[1,]
MSE.EN2 <- result_elastic.net_cv.x[9,]
MSE.EN3 <- result_elastic.net_cv.x[5,]
EN_mse <- data.frame(MSE1 = MSE.EN1, MSE2 = MSE.EN2, MSE3 = MSE.EN3,
                     Lambda = log(lambda_path_elastic.net))
min.lambda.EN =  log(elastic.net.best.lambda.x)

# Construct a dataframe for the result of LPTL+LPTN
LPTL_LPTN_mse <- data.frame(MSE = result.LPTN_LPTL.x.cv['mean_mse',], Lambda = log
    (result.LPTN_LPTL.x.cv['lambda_cur',]))
min.lambda.lptl.lptn =  log(best.lambda_LPTN_LPTL)

pdf("app2.pdf",width=10,height=5)
```

```
# Left plot of MSE of elastic net against log(Lambda)
# where Elastic Net with 3 alpha values: 0.1,0.5,0.9
lg3 <- ggplot(data = EN_mse)+
       geom_line(aes(x = Lambda, y = MSE1, color = "0.1"))+
       geom_point(aes(x = Lambda, y = MSE1, color = "0.1"))+
       geom_line( aes(x = Lambda, y = MSE2, color = "0.9"))+
       geom_point(aes(x = Lambda, y = MSE2,color = "0.9"))+
       geom_line(aes(x = Lambda, y = MSE3, color = "0.5"))+
       geom_point(aes(x = Lambda, y = MSE3, color = "0.5"))+
       scale_color_manual(values = c("0.1" = "blue", "0.9" = "purple", "0.5" = "
          darkorange"))+labs(color = "Alpha")+xlab("Log(Lambda)")+ylab("MSE")+
          ggtitle("Elastic Net")+geom_vline(aes(xintercept=min.lambda.EN),
          linetype = "dashed", color = "blue", size =1)+theme_bw()+theme(axis.
          title.x = element_text(size = 15), axis.title.y = element_text(size =
          15),axis.text = element_text(size = 13),plot.title = element_text(size =
           30,hjust = 0.5),plot.margin=unit(rep(0.8,4),'lines'),legend.position =
          c(0.1,0.83),legend.text = element_text(size = 14), legend.title =
          element_text(size =15))+annotation_custom(textGrob("-4.6", gp = gpar(col
           = "red", fontsize= 14)),xmin=min.lambda.EN, xmax=min.lambda.EN,ymin
          =0.48, ymax=0.48) +annotation_custom(segmentsGrob(gp = gpar(col = "red",
           lwd = 2)),xmin=min.lambda.EN, xmax=min.lambda.EN,ymin=0.48, ymax=0.48)+
          coord_cartesian(ylim=c(0.5,0.8), clip="off")

# Right plot of MSE of best LPTL+LPTN model against log(Lambda)
lg4 <- ggplot(data = LPTL_LPTN_mse, aes(x = Lambda, y = MSE))+
       geom_line(size =1, color = "skyblue")+
       geom_point(color = "darkblue",size = 1.8)+
       xlab("Log(Lambda)")+ggtitle("LPTL+LPTN")+
       geom_vline(aes(xintercept=min.lambda.lptl.lptn), linetype = "dashed",color
          = "darkorange", size =1)+theme_bw()+
       theme(axis.title.x = element_text(size = 15), axis.title.y = element_text(
          size = 15),axis.text = element_text(size = 13),plot.title = element_text
          (size = 22,hjust = 0.5),plot.margin=unit(rep(0.8,4),'lines'))+annotation
          _custom(textGrob("3.2", gp = gpar(col = "red", fontsize= 14)),xmin=min.
          lambda.lptl.lptn, xmax=min.lambda.lptl.lptn,ymin=0.447, ymax=0.447) +
          annotation_custom(segmentsGrob(gp = gpar(col = "red", lwd = 2)),xmin=min
          .lambda.lptl.lptn, xmax=min.lambda.lptl.lptn, ymin=0.447, ymax=0.447)+
          coord_cartesian(ylim=c(0.45,0.6), clip="off")

ggarrange(lg3,lg4,ncol=2)
dev.off()

##################### Figure A.6 ########################
# Best subset selection estimates vs. OLS estiamtes
pdf("ap2.pdf",width=10,height = 6)
ggplot(data = estimates)+geom_hline(aes(yintercept=0),linetype="dashed",color="
   black",size=1.2,alpha=0.5)+geom_line(aes(x = variables, y = ols, color = "OLS")
   ,size = 1)+geom_point(aes(x = variables, y = ols, shape = "OLS", fill = "OLS"),
    size=3)+geom_line(aes(x = variables, y = subset, color = "Subset Selection"),
   size=1)+geom_point(aes(x = variables, y = subset, shape = "Subset Selection",
   fill ="Subset Selection"), size=3)+ylab("Estimates")+xlab("Parameters")+scale_y
   _continuous(breaks=seq(-0.5, 0.8, by=0.25))+scale_shape_manual(name = "Method",
   values = c('Subset Selection' = 24, "OLS" = 21))+scale_color_manual(name = "
   Method",values = c('Subset Selection' = 'coral3', 'OLS' = 'green3'))+scale_fill
   _manual(name = "Method",values = c('Subset Selection' = 'darkred','OLS' = '
   darkgreen'))+theme_bw()+ theme(axis.title.x = element_text(size = 15), axis.
   title.y = element_text(size = 15),axis.text = element_text(size = 13),legend.
```

```
    position = c(0.88,0.88),legend.text = element_text(size = 14),legend.background
     = element_rect(fill="lightskyblue1",size=0.5,linetype="solid", colour ="
    darkblue"),legend.title = element_text(size =16),legend.key.size =unit(1.6,"
    line"))
dev.off()

##################### Figure A.7 ########################
# Best ridge estimates vs. OLS estiamtes
pdf("ap3.pdf",width=10,height = 6)
ggplot(data = estimates)+
  geom_hline(aes(yintercept=0), linetype="dashed",color = "black", size =1.2,alpha
      =0.5)+
  geom_line(aes(x = variables, y = ols, color = "OLS"),size = 1)+
  geom_point(aes(x = variables, y = ols, shape = "OLS", fill = "OLS"), size=3)+
  geom_line(aes(x = variables, y = ridge, color = "Ridge"),size=1)+
  geom_point(aes(x=variables,y = ridge,shape = "Ridge",fill ="Ridge"), size=3)+
      ylab("Estimates")+xlab("Parameters")+scale_y_continuous(breaks=seq(-0.5, 0.8,
       by=0.25))+
  scale_shape_manual(name = "Method",values = c('Ridge' = 24, "OLS" = 21))+
  scale_color_manual(name = "Method",values = c('Ridge' = 'darkorchid', 'OLS' = '
      green3'))+
  scale_fill_manual(name = "Method",values = c('Ridge' = 'yellow','OLS' = '
      darkgreen'))+theme_bw()+
  theme(axis.title.x = element_text(size = 15), axis.title.y = element_text(size =
       15),axis.text = element_text(size = 13),legend.position = c(0.94,0.87),
      legend.text = element_text(size = 14),legend.background = element_rect(fill="
      lightskyblue1",size=0.5, linetype="solid", colour ="darkblue"),legend.title =
       element_text(size =16),legend.key.size = unit(1.6,"line"))
dev.off()

##################### Figure A.8 ########################
# Best elastic net estimates vs. OLS estiamtes
pdf("ap4.pdf",width=10,height = 6)
ggplot(data = estimates)+
  geom_hline(aes(yintercept=0), linetype="dashed",color = "black", size =1.2,alpha
      =0.5)+
  geom_line(aes(x = variables, y = ols, color = "OLS"),size = 1)+
  geom_point(aes(x = variables, y = ols, shape = "OLS", fill = "OLS"), size=3)+
  geom_line(aes(x = variables, y = els, color = "Elastic Net"),size=1)+
  geom_point(aes(x = variables, y = els, shape = "Elastic Net", fill ="Elastic Net
      "), size=3)+
  ylab("Estimates")+xlab("Parameters")+scale_y_continuous(breaks=seq(-0.5, 0.8, by
      =0.25))+
  scale_shape_manual(name = "Method",values = c('Elastic Net' = 24, "OLS" = 21))+
  scale_color_manual(name = "Method",values = c('Elastic Net' = 'dodgerblue2', '
      OLS' = 'green3'))+
  scale_fill_manual(name = "Method",values = c('Elastic Net' = 'yellow','OLS' = '
      darkgreen'))+theme_bw()+
  theme(axis.title.x = element_text(size = 15), axis.title.y = element_text(size =
       15),
        axis.text = element_text(size = 13),legend.position = c(0.9,0.89),
        legend.text = element_text(size = 14),
        legend.background = element_rect(fill="lightskyblue1",size=0.5,
                                        linetype="solid", colour ="darkblue"),
        legend.title = element_text(size =16),legend.key.size = unit(1.6,"line"))
dev.off()
```

```
##################### Figure A.9 ########################
# Best LPTL+LPTN estimates vs. OLS estiamtes
pdf("ap1.pdf",width=10,height = 6)
ggplot(data = estimates)+
  geom_hline(aes(yintercept=0), linetype="dashed",color = "black", size =1.2,alpha
     =0.5)+
  geom_line(aes(x = variables, y = ols, color = "OLS"),size=1)+
  geom_point(aes(x = variables, y = ols, shape = "OLS", fill = "OLS"), size=3)+
  geom_line(aes(x = variables, y = lptl.lptn, color = "LPTL+LPTN"),size=1)+
  geom_point(aes(x = variables, y = lptl.lptn, shape = "LPTL+LPTN", fill = "LPTL+
     LPTN"), size=3)+
  scale_shape_manual(name = "Method",values = c("OLS" = 21, 'LPTL+LPTN' = 25))+
  scale_color_manual(name = "Method",values = c('OLS' = 'green3','LPTL+LPTN' = '
     pink'))+
  scale_fill_manual(name = "Method",values = c('OLS' = 'darkgreen','LPTL+LPTN' = '
     yellow'))+
  ylab("Estimates")+xlab("Parameters")+scale_y_continuous(breaks=seq(-0.5, 0.8, by
     =0.25))+theme_bw()+
  theme(axis.title.x = element_text(size = 15), axis.title.y = element_text(size =
      15),axis.text = element_text(size = 13),legend.position = c(0.9,0.88),
        legend.text = element_text(size = 14),
        legend.background = element_rect(fill="lightskyblue1",size=0.5,
                                         linetype="solid", colour ="darkblue"),
        legend.title = element_text(size =16),legend.key.size = unit(1.6,"line"))

dev.off()
```