Bayesian Approaches to Penalised Regression

Anthony Burn

April 21, 2024

Preface

This piece of work is a result of my own and I have complied with the Department's guidance on multiple submission and on use of AI tools. Material from the work of others not involved in the project has been acknowledged, quotations and paraphrases suitably indicated, and all uses of AI tools have been declared.

Contents

Chapter 1

Introduction

In 1970, Arthur E. Hoerl and Robert W. Kennard [10] proposed an ad-hoc solution to problems faced in linear regression. They found that linear models often become more useful and exhibit superior predictive accuracy with the introduction of a penalising parameter to reduce the size of estimated $\hat{\boldsymbol{\beta}}$ coefficients. This project introduces the problems which plague ordinary least square regression and details how they are fixed through coefficient penalisation.

The ridge regression estimator and the lasso regression estimators both improve the predictive performance of linear models, and have surprisingly elegant links to Bayesian statistics which are detailed in this project. However, the transition from Frequentist to Bayesian statistics prompts extra difficulties and questions, for instance, how should we determine the penalisation parameter? How can we determine coefficients in the Bayesian setting? How do we determine reasonable prior distributions for our parameters? This project attempts to answer these questions while providing an intuitive and simple introduction to Bayesian thinking with regards to linear regression. Upon setting up our models, we then compare their efficacy and parsimony through a simulation study. The Bayesian approach has several advantages over traditional approaches such as:

- Intuitive derivations of popular frequentist penalisation techniques in the Bayesian setting, using well known distributions (for example the derivation of (4.4))
- Having multiple ways to calculate the optimal penalising parameter, not just through cross-validation (as is the common case)
- The opportunity to carry out a Bayesian analysis of our estimated parameters. This means we can set up credible intervals for all estimated parameters, which (depending on personal preference) can be nicer to interpret
- Including extra information in our models through the use of prior distributions on various parameters and hyper-parameters can often improve model performance
- Superior predictive and inferential accuracy in many cases (as data suggests in our simulation study)

We will show the derivation of Bayesian regression in a simple setting by hand and then use RStan, [22], to perform complicated derivations and algorithms quickly and effectively. Using RStan makes it so that we can create powerful and complicated models which perform very competitively with Frequentist equivalents. All code is stored in the appendix.

Chapter 2

Linear Regression

2.1 Ordinary Least Squares Regression

Let us say we wish to predict the height of a tall tree, as it is not easy to measure directly. Various factors may impact (or be related to) the height of the tree, for instance, the width of the base, the species of tree, soil density, rainfall and so on. Alongside knowledge of the true height of a few trees and their respective factors, we can fit a linear model to predict the height of trees using the various recorded factors, meaning we do not need to physically measure their heights to have a reasonable estimate of them. In this motivating example, the height of the tree is our **response variable**, $\boldsymbol{Y} = (y_1, y_2, ..., y_n)^T$, a $n \times 1$ matrix, and the various factors are our **predictor variables**, stored in the *design matrix*, $\boldsymbol{X}$, a $n \times p$[1] matrix. With n representing the number of observations, while p is the number of predictor variables. The linear model states $\boldsymbol{Y} = \boldsymbol{X}\boldsymbol{\beta} + \boldsymbol{\epsilon}$, with $\boldsymbol{\beta} = (\beta_0, \beta_1, ..., \beta_{p-1})^T$ being a $p \times 1$ vector. The components of the vector $\boldsymbol{\beta}$ are called the *regression parameters*[2] and these parameters relate the predictor variables with the response. And, $\boldsymbol{\epsilon} = (\epsilon_1, \epsilon_2, ..., \epsilon_n)^T$, an $n \times 1$ vector. Notice that $\boldsymbol{\epsilon} = \boldsymbol{Y} - \boldsymbol{X}\boldsymbol{\beta}$, and for each individual $i \in \{1, ..., n\}$, ϵ_i is the distance between the response y_i and $\sum_{j=0}^{p-1} x_{ij}\beta_j$. We assume that for $\forall i$, ϵ_i are independent and identically distributed such that $\epsilon_i \sim \mathcal{N}(0, \sigma^2)$.

In order to make inferences about the relationship between our response vector, $\boldsymbol{Y}$, and our data, $\boldsymbol{X}$, we must find an appropriate $\boldsymbol{\beta}$. We assume that there are "true" values of $\boldsymbol{\beta}$, which govern the true relationship between $\boldsymbol{Y}$ and $\boldsymbol{X}$. In reality we do not know these true values for $\boldsymbol{\beta}$, and so we must estimate them. The most simple method of estimating $\boldsymbol{\beta}$ is through least squares estimation, we also call this OLS (ordinary least squares) regression. OLS regression is called as such as it estimates $\boldsymbol{\beta}$ to be whichever $\hat{\boldsymbol{\beta}}$ which minimises the value of the sum of the square errors of our model, $\boldsymbol{\epsilon}^T\boldsymbol{\epsilon} = \sum_{i=1}^{n} \epsilon_i^2$.

$$
\begin{aligned}
\hat{\boldsymbol{\beta}} &= \arg\min_{\boldsymbol{\beta}} \boldsymbol{\epsilon}^T\boldsymbol{\epsilon}, \qquad (2.1)\\
\boldsymbol{\epsilon}^T\boldsymbol{\epsilon} &= (\boldsymbol{Y} - \boldsymbol{X}\boldsymbol{\beta})^T(\boldsymbol{Y} - \boldsymbol{X}\boldsymbol{\beta}),\\
&= \boldsymbol{Y}^T\boldsymbol{Y} - \boldsymbol{Y}^T\boldsymbol{X}\boldsymbol{\beta} - \boldsymbol{\beta}^T\boldsymbol{X}^T\boldsymbol{Y} + \boldsymbol{\beta}^T\boldsymbol{X}^T\boldsymbol{X}\boldsymbol{\beta},\\
\frac{\partial(\boldsymbol{\epsilon}^T\boldsymbol{\epsilon})}{\partial\boldsymbol{\beta}} &= -\boldsymbol{X}^T\boldsymbol{Y} - \boldsymbol{X}^T\boldsymbol{Y} + \boldsymbol{X}^T\boldsymbol{X}\boldsymbol{\beta} + (\boldsymbol{\beta}^T\boldsymbol{X}^T\boldsymbol{X})^T.
\end{aligned}
$$

[1] When our model includes an intercept term, β_0, the entries of the first column of the design matrix are all 1.

[2] The regression parameters, $\boldsymbol{\beta}$, are also referred to as the *coefficients*

And now since we wish to minimise $\boldsymbol{\epsilon}^T\boldsymbol{\epsilon}$, we set the above expression to zero.

$$\frac{\partial(\boldsymbol{\epsilon}^T\boldsymbol{\epsilon})}{\partial\hat{\boldsymbol{\beta}}} = -2\boldsymbol{X}^T\boldsymbol{Y} + 2\boldsymbol{X}^T\boldsymbol{X}\hat{\boldsymbol{\beta}} = 0,$$
$$\boldsymbol{X}^T\boldsymbol{X}\hat{\boldsymbol{\beta}} = \boldsymbol{X}^T\boldsymbol{Y}.$$

Hence, we derive the *normal equation,*

$$\hat{\boldsymbol{\beta}} = (\boldsymbol{X}^T\boldsymbol{X})^{-1}\boldsymbol{X}^T\boldsymbol{Y}. \tag{2.2}$$

The normal equation, (2.1), gives us the vector, $\hat{\boldsymbol{\beta}} = (\hat{\beta}_0, \hat{\beta}_1, ..., \hat{\beta}_{p-1})^T$, which minimises $\sum_{i=1}^n \epsilon_i^2$ among the data used to train the model. We introduce the idea of estimator bias to understand the basic properties of the estimate of $\boldsymbol{\beta}$ provided by the normal equation.

Definition 2.1.1. ***Bias*** *refers to the difference between the expectation of our estimator,* $\mathbb{E}[\hat{\boldsymbol{\beta}}]$*, and the true value of* $\boldsymbol{\beta}$*.*

$$Bias := \mathbb{E}[\hat{\boldsymbol{\beta}}] - \boldsymbol{\beta}.$$

For the normal equation estimate of $\boldsymbol{\beta}$, (2.1), we calculate;

$$\begin{aligned}\hat{\boldsymbol{\beta}}_{bias} &= \mathbb{E}[\hat{\boldsymbol{\beta}}] - \boldsymbol{\beta},\\ &= \mathbb{E}[(\boldsymbol{X}^T\boldsymbol{X})^{-1}\boldsymbol{X}^T Y] - \boldsymbol{\beta},\\ &= (\boldsymbol{X}^T\boldsymbol{X})^{-1}\boldsymbol{X}^T\mathbb{E}[Y] - \boldsymbol{\beta},\\ &= (\boldsymbol{X}^T\boldsymbol{X})^{-1}\boldsymbol{X}^T X\boldsymbol{\beta} - \boldsymbol{\beta},\\ &= 0.\end{aligned}$$

Since the bias of this estimator is zero, we say that it is an **unbiased** estimator of the "true" $\boldsymbol{\beta}$. Furthermore, we can calculate the variance of $\hat{\boldsymbol{\beta}}$ as follows,

$$\begin{aligned}\mathbb{V}ar(\hat{\boldsymbol{\beta}}) &= \mathbb{V}ar((\boldsymbol{X}^T\boldsymbol{X})^{-1}\boldsymbol{X}^T Y),\\ &= (\boldsymbol{X}^T\boldsymbol{X})^{-1}\boldsymbol{X}^T\mathbb{V}ar(Y)[(\boldsymbol{X}^T\boldsymbol{X})^{-1}\boldsymbol{X}^T]^T,\\ &= \sigma^2(\boldsymbol{X}^T X)^{-1}.\end{aligned}$$

Although the above expression for $\mathbb{V}ar(\hat{\boldsymbol{\beta}})$ is correct, a real situation in which we know σ^2 is very rare. Hence, it must be estimated. It can be shown that

$$s^2 = \frac{\hat{\boldsymbol{\epsilon}}^T\hat{\boldsymbol{\epsilon}}}{n-p}$$

is an unbiased estimator[3] of σ^2. Due to the need to estimate σ^2, we consider the standard error of $\hat{\boldsymbol{\beta}}$ ($SE[\hat{\boldsymbol{\beta}}] = s\sqrt{(\boldsymbol{X}^T X)^{-1}}$) rather than the standard deviation ($SD[\hat{\boldsymbol{\beta}}] = \sigma\sqrt{(\boldsymbol{X}^T X)^{-1}}$), and use the t-distribution with $n-p$ degrees of freedom when estimating its properties. The lack of bias and the easily calculated variance of $\hat{\boldsymbol{\beta}}$ make it an appealing option for fitting a linear regression model. However, in reality we often find that (2.1) does not provide us with the optimal $\hat{\boldsymbol{\beta}}$, in terms of predictive and inferential power.

2.2 Collinearity in Least Squares Regression

Definition 2.2.1. ***Collinearity*** *in linear regression refers to the event of two (or more) predictor variables being strongly linearly related [27].*

[3] Where $\hat{\boldsymbol{\epsilon}} = \boldsymbol{Y} - \boldsymbol{X}\hat{\boldsymbol{\beta}}$.

Collinearity manifests itself with regards to our linear model within the design matrix, $\boldsymbol{X}$. It results in multiple columns of $\boldsymbol{X}$ being very similar, and is especially common in models with lots of predictor variables (when p is large). While collinearity is present in predictor variables the individual components of $\hat{\boldsymbol{\beta}}$ representing the relationships between the predictors and the response will exhibit high standard errors.

Example 2.2.2. *(Code in appendix, 2.2.2 A). Consider data simulated as follows:*

$$\boldsymbol{X} \sim N_3\left(\begin{pmatrix}0\\0\\0\end{pmatrix}, \begin{pmatrix}1 & 0 & 0\\0 & 1 & 0.9\\0 & 0.9 & 1\end{pmatrix}\right).$$

We take $n = 10$ samples from this multivariate normal distribution. Notice that variable x_1 is independent of x_2 and x_3, however, x_2 and x_3 have a high covariance of 0.9[4]. During simulation this means that there will be high collinearity between x_2 and x_3. Consider a true vector of $\boldsymbol{\beta}$ given by $\boldsymbol{\beta} = (5, 5, 5)^T$ and a response variable, Y, given by $Y = \boldsymbol{X}\beta + \boldsymbol{\epsilon}$, where $\epsilon_i \sim N(0, 1)$. Fitting a linear regression model to use the generated data, $\boldsymbol{X}$, to predict the response, Y, results in estimates of the true $\boldsymbol{\beta}$ being calculated using (2.1). These estimates, $\hat{\boldsymbol{\beta}}$, will have standard errors which roughly represent how sure we are that the given $\hat{\beta}_j$ takes the value which we have assigned to it. Using these standard errors we can also run statistical tests to estimate the probabilities that any given β_j is insignificant (where $\beta_j = 0$). When simulated the data above gives the following statistics.

j	$\hat{\beta}_j$	Std. Error	t-value	p-value
1	5.25	0.253	20.7	8.22×10^{-7}
2	4.47	0.975	4.59	3.74×10^{-3}
3	5.08	0.752	6.75	5.15×10^{-4}

Table 2.1: A table of results for the simulation detailed above.

Table 2.1 above shows the high standard errors of coefficients for highly collinear predictors, in this case $j = 2$ and $j = 3$ are collinear. The standard error of $\hat{\beta}_1$ is much lower than that of $\hat{\beta}_2$ and $\hat{\beta}_3$. We say that a coefficient is insignificant if it equal to zero. The table also shows the probability that any given predictor is insignificant in predicting the value of Y. Since the standard errors for $j = 2$ and $j = 3$ are higher, the chance that the related predictor variables are insignificant is much greater (although in this case they are still all significant), in spite of the fact that the true value for each $\beta_j = 5$.

The standard errors of the $\boldsymbol{\beta}$ components are given by $\text{SE}(\hat{\beta}_i) = s\sqrt{[(\boldsymbol{X}^T\boldsymbol{X})^{-1}]_{ii}}$, where s is the unbiased estimate of the standard deviation of the error term, ϵ_i. In this collinear case, the collinearity results in $[(\boldsymbol{X}^T\boldsymbol{X})^{-1}]_{22}$ and $[(\boldsymbol{X}^T\boldsymbol{X})^{-1}]_{33}$ being larger. To aid with intuition, picture the model as unsure of which of the variables, $\hat{\beta}_2$ or $\hat{\beta}_3$, are causing the underlying error. The data influencing $\hat{\beta}_2$ and $\hat{\beta}_3$ is so similar that the model attributes high standard error to both of the estimates due to the uncertainty in which of the two estimates are furthest from their theoretical true values (in the above example the true values would be 5 for both betas).

Definition 2.2.3. ***Super-collinearity*** *occurs when multiple predictor variables are exactly linearly correlated.*

[4]In this case, since the variances are all 1, the covariance is equivalent to the correlation between the variables

When we observe super-collinearity, some of the columns of $\boldsymbol{X}$ will be linearly dependent and so, $\boldsymbol{X}$ will not be full of rank [27]. This indicates that $\boldsymbol{X}^T\boldsymbol{X}$ is singular, as follows:

Consider an $\boldsymbol{X} \in \mathbb{R}^{n\times p}$ such that $\text{rank}(\boldsymbol{X}) < p$. Hence, $\exists\ \boldsymbol{v} \in \mathbb{R}^p$ such that $\boldsymbol{X}\boldsymbol{v} = \boldsymbol{0}_n$, where $\boldsymbol{v}$ is non-trivial. Then,

$$\begin{aligned} \boldsymbol{X}\boldsymbol{v} &= \boldsymbol{0}_n, \\ \boldsymbol{X}^T\boldsymbol{X}\boldsymbol{v} &= \boldsymbol{X}^T\boldsymbol{0}_n, \\ &= \boldsymbol{0}_p \end{aligned}$$

Assuming $\boldsymbol{X}^T\boldsymbol{X}$ is invertible,

$$\begin{aligned} \boldsymbol{v} &= (\boldsymbol{X}^T\boldsymbol{X})^{-1}\boldsymbol{0}_p, \\ \implies \boldsymbol{v} &= \boldsymbol{0}_p. \end{aligned}$$

However, this is a contradiction as we assumed that $\boldsymbol{v}$ was non-trivial, and so our assumption that $\boldsymbol{X}^T\boldsymbol{X}$ is invertible is false. This tells us that we cannot use equation (2.1) for any models exhibiting super-collinearity since $(\boldsymbol{X}^T\boldsymbol{X})^{-1}$ will be undefined □.

2.3 Overfitting

Definition 2.3.1. ***Overfitting*** *occurs when a model is fit very closely to training data resulting in poor predictive accuracy.*

Within the context of the linear model, using the normal equation to obtain $\hat{\boldsymbol{\beta}}$ often results in the model compensating for outlying data points in the design matrix, $\boldsymbol{X}$. When dealing with a data set containing a low number of observations, n small, outliers often have a large influence on the values of $\hat{\boldsymbol{\beta}}$. Moreover, the model will often unnecessarily account for random patterns in the error terms, $\boldsymbol{\epsilon}$, which arise in the training data. This leads to $\hat{\boldsymbol{\beta}}$ values being too large, i.e. $\hat{\boldsymbol{\beta}} > \boldsymbol{\beta}$, resulting in poor predictive accuracy. This concept is highlighted in figure 2.1.

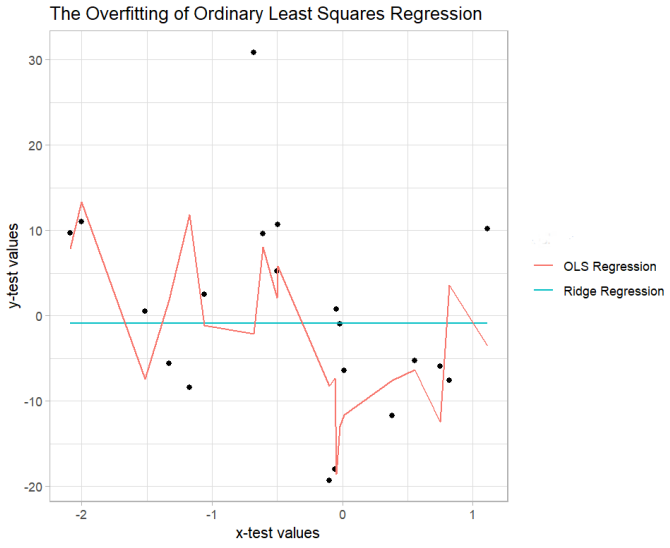

Figure 2.1: The figure above shows models trained and tested on simulated data, plotted with respect to one predictor variable.

Figure 2.1 shows that the OLS line accounts for random patterns found in the error terms, $\boldsymbol{\epsilon}$, in the data which the model was trained on. Accounting for errors in this way is an example of overfitting. This overfitting hinders the predictive performance of the model. The ridge regression line, is much simpler and is generally more effective at prediction. We introduce ridge regression in the following chapter. Overfitting is undetectable when testing a model on the data with which it has been trained, and it only becomes apparent through inaccurate predictions. This is why we test our model on data which was not used during the training of our model. This distinction is particularly important with respect to OLS regression. OLS regression provides the closest fit to the data points that it is trained upon which is mathematically possible, by minimising the residual sum of squares. Because of this, if we only tested our model using the data with which it was trained we would (potentially wrongly) believe that OLS regression provides the most optimal model possible.

There are alternative regression methods available to OLS regression which can reduce the impacts of collinearity and overfitting, such as principal component analysis (PCA) [12]. PCA can improve predictive performance by compressing our data into uncorrelated principal components, however, by changing the type of regression in this way we lose the human interpretability of our model. This makes it harder to infer the relationship between the response vector and our predictor variables. In general, when we seek to make inferences about which factors impact a response variable, we seek a parsimonious model. This means that we want our model to be simple. A simple model allows us to make clearer observations about individual predictor variable's effect on the response, and linear regression is very simple and easy to interpret. The penalised regression methods discussed in the next chapter reduce the problems of collinearity and overfitting in ordinary least squares regression while keeping the models intuitive to interpret.

Chapter 3

Ridge and Lasso Regression

3.1 Ridge Regression

The problems of collinearity and overfitting in linear regression lead to the creation of the ridge regression estimator [10].

Definition 3.1.1. ***The ridge regression estimator*** *is the estimate of* $\boldsymbol{\beta}$ *given by the following equation.*

$$\hat{\boldsymbol{\beta}}(\lambda) = (\boldsymbol{X}^T\boldsymbol{X} + \lambda I)^{-1}\boldsymbol{X}^T\boldsymbol{Y},\ \lambda \in \mathbb{R}^+. \tag{3.1}$$

The variable λ is known as the penalisation parameter and can be adjusted for best results. Notice the similarity between (3.1) and (2.1), the only difference being the replacement of $(\boldsymbol{X}^T\boldsymbol{X})^{-1}$ with $(\boldsymbol{X}^T\boldsymbol{X} + \lambda I)^{-1}$. Also, spot that if $\lambda = 0$ then (3.1) is identical to (2.1). This penalising term is a way to reduce the impacts of overfitting and collinearity within the data, the details of which we describe later in this section.

Definition 3.1.2. ***The*** $p-$***norm*** $\|\cdot\|_p$ *of an* n *dimensional vector,* $\boldsymbol{v}$*, is given by* $\|\boldsymbol{v}\|_p = (v_1^p + v_2^p + ... + v_n^p)^{\frac{1}{p}}$.

Definition 3.1.3. ***The ridge loss function*** *is given by*

$$L_{ridge}(\boldsymbol{\beta}; \lambda) = \|\boldsymbol{Y} - \boldsymbol{X}\boldsymbol{\beta}\|_2^2 + \lambda\|\boldsymbol{\beta}\|_2^2,$$

and we pick $\hat{\boldsymbol{\beta}}_{ridge}$ *such that,*

$$\begin{aligned}\hat{\boldsymbol{\beta}}(\lambda) &= \arg\min_{\boldsymbol{\beta}} \|\boldsymbol{Y} - \boldsymbol{X}\boldsymbol{\beta}\|_2^2 + \lambda\|\boldsymbol{\beta}\|_2^2\,, \\ &= \arg\min_{\boldsymbol{\beta}} \sum_{i=1}^{n}(y_i - \boldsymbol{x_i}^T\boldsymbol{\beta})^2 + \lambda\sum_{j=1}^{p}\beta_j^2,\end{aligned} \tag{3.2}$$

[28] where $\boldsymbol{x}_i^T$ *denotes the* i^{th} ***row*** *vector of* $\boldsymbol{X}$ *(which will have dimensions* $1 \times p$*).*

It is important to note that we do not penalise the intercept term $\hat{\beta}_0$ while using frequentist penalisation methods. We can understand why this is the case by considering a case raised in [24] (p.64). They raise the point that, if we were to penalise β_0, then shifting all response variables, $y_i \to y_i + c$ would not result in a shift of the predictions by the same amount as β_0 would shift to be less than $\beta_0 + c$ due to the penalisation. This is not desirable.

Another important practical note for applying ridge (and all other penalised regression methods) is that before we implement these techniques, we must first standardise our data matrix, $\boldsymbol{X}$. This is because the scale of our data impacts the values of $\boldsymbol{\beta}$. Consider a single

variable placed on a large scale, for instance we measure the volume of a swimming pool in cubic millimeters as one of our predictor variables in a model. This results in the associated column of data in our design matrix $\boldsymbol{X}$ being very large. Hence, the associated $\boldsymbol{\beta}_j$ coefficient will be much smaller than if we had used a different unit to measure the volume such as cubic meters. Because this coefficient will be smaller, it will contribute less to the penalty term. This is undesirable as we have no theoretical reason for this coefficient to contribute less to the penalty. This issue is fixed by standardising the columns of our data matrix. We do this using the equation for the standardised data cell, $\tilde{x}_{ij}$,

$$\tilde{x}_{ij} = \frac{x_{ij} - \bar{x}_j}{\sqrt{\mathbb{V}ar(\boldsymbol{x_j})}},$$

where $\bar{x}_j$ represents the mean of the j^{th} column of $\boldsymbol{X}$, and $\mathbb{V}ar(\boldsymbol{x_j})$ is the variance of column j of $\boldsymbol{X}$, and x_{ij} is the data in the i^{th} row and the j^{th} column of $\boldsymbol{X}$. The standardised design matrix, $\tilde{\boldsymbol{X}}$ will have column means of zero and column variances of 1.

We wish to compare the models which arise using the least squares model, with $\hat{\boldsymbol{\beta}}$ given by (2.1), and a ridge regression model using equation (3.1). So, we now examine the bias of the ridge regression estimator, and its penalising impact on the response variable.

Proposition 3.1.4. *The magnitude of the bias of the ridge estimator is monotone increasing with respect to the penalising parameter, λ.*

Proof. To find the bias of the ridge estimator we must first find its expectation.

$$\begin{aligned}
\hat{\boldsymbol{\beta}}(\lambda) &= (\boldsymbol{X}^T\boldsymbol{X} + \lambda I)^{-1}\boldsymbol{X}^T\boldsymbol{Y}, \\
\mathbb{E}[\hat{\boldsymbol{\beta}}(\lambda)] &= \mathbb{E}[(\boldsymbol{X}^T\boldsymbol{X} + \lambda I)^{-1}\boldsymbol{X}^T\boldsymbol{Y}], \\
&= (\boldsymbol{X}^T\boldsymbol{X} + \lambda I)^{-1}\boldsymbol{X}^T\mathbb{E}[\boldsymbol{Y}], \\
&= (\boldsymbol{X}^T\boldsymbol{X} + \lambda I)^{-1}\boldsymbol{X}^T\boldsymbol{X}\boldsymbol{\beta}.
\end{aligned}$$

Consider the expression,

$$\boldsymbol{X}^T\boldsymbol{X} = \boldsymbol{X}^T\boldsymbol{X} + \lambda I - \lambda I,$$

premultiplying both sides by $(\boldsymbol{X}^T\boldsymbol{X} + \lambda I)^{-1}$ gives us,

$$(\boldsymbol{X}^T\boldsymbol{X} + \lambda I)^{-1}\boldsymbol{X}^T\boldsymbol{X} = I - \lambda(\boldsymbol{X}^T\boldsymbol{X} + \lambda I)^{-1}.$$

We can substitute this into our expression for $\mathbb{E}[\hat{\boldsymbol{\beta}}(\lambda)]$

$$\mathbb{E}[\hat{\boldsymbol{\beta}}(\lambda)] = \boldsymbol{\beta} - \lambda(\boldsymbol{X}^T\boldsymbol{X} + \lambda I)^{-1}\boldsymbol{\beta},$$

hence, we find the bias of the ridge estimator to be:

$$Bias_{ridge} = -\lambda(\boldsymbol{X}^T\boldsymbol{X} + \lambda I)^{-1}\boldsymbol{\beta}. \tag{3.3}$$

The equation above shows that the bias of the ridge regression estimator has a linear relationship with the penalising parameter, λ. Hence, $|Bias_{ridge}(\lambda)|$ is a monotone increasing function with respect to λ, meaning that as λ increases, so does the bias.

Proposition 3.1.5. *The magnitude of the fitted values $\hat{y}_i(\lambda)$ which make up the vector $\hat{\boldsymbol{Y}} = \boldsymbol{X}\hat{\boldsymbol{\beta}}(\lambda)$ shrink towards zero as λ increases.*

Proof, [27] (p.9). By using a singular value decomposition (SVD) of our design matrix, $\boldsymbol{X}$, we know we can write that $\boldsymbol{X} = UDV^T$. SVD tells us that U and V are $n \times n$ and $p \times p$ respectively. Also they are both orthogonal ($UU^T = U^TU = I_{n\times n}$ and $VV^T = V^TV =$

$I_{p\times p}$. Furthermore, we know that D is an $n \times p$ diagonal matrix, meaning that it only has non-zero elements when the row number equals the column number. If $n > p$, then

$$D = \left(\begin{array}{cccccc} d_1 & 0 & 0 & ... & ... & 0 \\ 0 & d_2 & 0 & ... & ... & 0 \\ 0 & 0 & d_3 & ... & ... & 0 \\ \vdots & \vdots & \vdots & \vdots & \ddots & \\ 0 & 0 & 0 & 0 & ... & d_p \\ \hline \\ 0 & 0 & 0 & 0 & ... & 0 \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & 0 & ... & 0 \end{array}\right)$$

likewise if $p \geq n$, then

$$D = \left(\begin{array}{cccc|ccc} d_1 & 0 & 0 & ... & 0 & ... & 0 \\ 0 & d_2 & 0 & ... & 0 & ... & 0 \\ 0 & 0 & d_3 & ... & 0 & ... & 0 \\ \vdots & \vdots & \vdots & \ddots & 0 & ... & 0 \\ 0 & 0 & ... & d_n & 0 & ... & 0 \end{array}\right).$$

Now, we use this to rewrite equation (3.1).

$$\begin{aligned} \hat{\boldsymbol{\beta}}(\lambda) &= (\boldsymbol{X}^T\boldsymbol{X} + \lambda I_{p\times p})^{-1}\boldsymbol{X}^T\boldsymbol{Y}, \\ \hat{\boldsymbol{y}} &= \boldsymbol{X}\hat{\boldsymbol{\beta}}(\lambda), \\ &= UDV^T(VD^TU^TUDV^T + \lambda VV^T)^{-1}VD^TU^TY. \end{aligned}$$

Using the orthogonality properties of U and V means that $VV^T = I_{p\times p}$ and $U^TU = I_{n\times n}$,

$$\begin{aligned} &= UDV^T(VD^TDV^T + \lambda VV^T)^{-1}VD^TU^TY, \\ &= UDV^TV(D^2 + \lambda I_{p\times p})^{-1}V^TVD^TU^TY, \\ &= UD(D^2 + \lambda I_{p\times p})^{-1}D^TU^TY \end{aligned} \tag{3.4}$$

Let's carry out the same analysis for $\hat{\boldsymbol{\beta}}_{LS}$, where $\hat{\boldsymbol{\beta}}_{LS}$ represents the least squares estimate of $\boldsymbol{\beta}$.

$$\begin{aligned} \hat{\boldsymbol{\beta}}_{LS} &= (\boldsymbol{X}^T\boldsymbol{X})^{-1}\boldsymbol{X}^TY, \\ \hat{\boldsymbol{y}} &= \boldsymbol{X}\hat{\boldsymbol{\beta}}_{LS}, \\ &= \boldsymbol{X}(\boldsymbol{X}^T\boldsymbol{X})^{-1}\boldsymbol{X}^TY \text{ [(2.1)]}, \\ &= UDV^T(VD^TU^TUDV^T)^{-1}VDU^TY, \\ &= UDV^TV(D^2)^{-1}V^TVDU^TY, \\ &= UD(D^2)^{-1}DU^TY. \end{aligned} \tag{3.5}$$

Using the notation $d_i = D_{ii}$, we now compare the i^{th} element in both (3.4) and (3.5). This gives us $\hat{y}_i(\lambda) = d_i/(d_i^2 + \lambda)$ and $\hat{y}_i = d_i^{-1}$. From these two equations we notice that $d_i/(d_i^2 + \lambda) < d_i^{-1} \implies \hat{y}_i(\lambda) < \hat{y}_i$, and this holds for $\forall\lambda > 0$. And from this inequality we also notice that as $\lambda \to \infty$, $\hat{y}(\lambda) \to 0$. [27] (p.9). This is the reason why λ is often referred to as the shrinkage parameter as well as the penalisation parameter. The shrinkage caused by using the ridge regression estimator helps combat the inflating effects of overfitting and

collinearity, leading to a better fit with regards to prediction (than OLS regression in general). We highlight the shrinkage of coefficients relative to the size of the penalising parameter, λ, in ridge regression using figure 3.1 (obtained from generic linear regression simulated data). Notice that all coefficients shrink towards zero. The true values of β_i are an equally spaced sequence between -4 and 4.

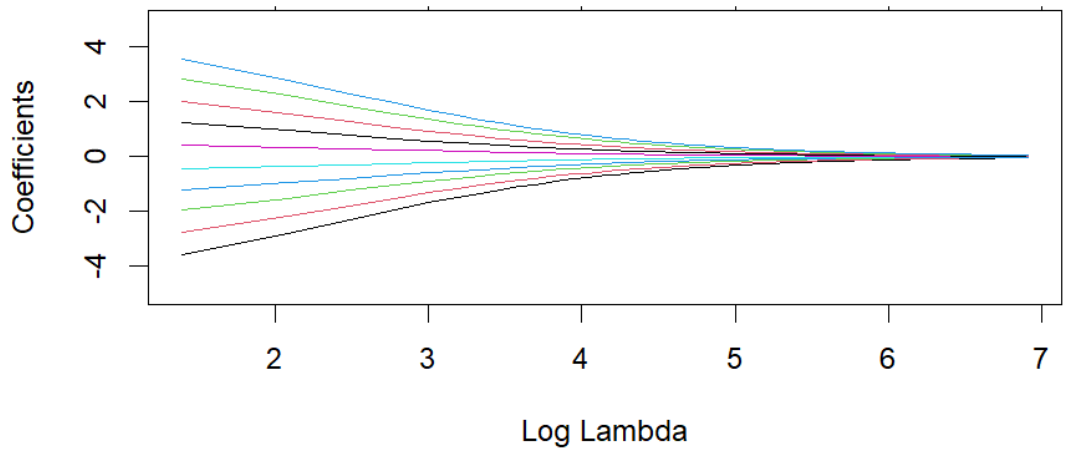

Figure 3.1: This graph shows values of $\hat{\beta}_i$, and how they vary as λ is increased (in this case for ease of reading we use $\log(\lambda)$, since for extreme shrinkage we need extreme values of λ).

3.2 The Bias-Variance Trade off

If using ridge regression gives us biased estimates, $\hat{\boldsymbol{\beta}}(\lambda)$, of the true $\boldsymbol{\beta}$ then how can we trust the coefficients we obtain by using its equations? To answer this question we must understand the nuanced idea of the bias-variance trade off. We have shown that increasing the value of λ increases the bias of our estimate and in this subsection we will go on to show that increasing λ has the useful effect of decreasing the variance of our fitted values.

Proposition 3.2.1. $\mathbb{V}ar(\hat{\boldsymbol{\beta}}(\lambda)) < \mathbb{V}ar(\hat{\boldsymbol{\beta}}_{OLS})$ *for* $\forall \lambda > 0$.

Proof. Consider the operator $\boldsymbol{W}_\lambda := (\boldsymbol{X}^T\boldsymbol{X} + \lambda I_p)^{-1}\boldsymbol{X}^T\boldsymbol{X}$ [27] (p.6). We first show this operator is linear with respect to any vectors $\boldsymbol{a}, \boldsymbol{b} \in \mathbb{R}^p$. For $\phi, \rho \in \mathbb{R}$, we have that:

$$\begin{aligned}
\boldsymbol{W}_\lambda(\phi\boldsymbol{a} + \rho\boldsymbol{b}) &= (\boldsymbol{X}^T\boldsymbol{X} + \lambda I_p)^{-1}\boldsymbol{X}^T\boldsymbol{X}(\phi\boldsymbol{a} + \rho\boldsymbol{b}), \\
&= (\boldsymbol{X}^T\boldsymbol{X} + \lambda I_p)^{-1}\boldsymbol{X}^T\boldsymbol{X}(\phi\boldsymbol{a}) + (\boldsymbol{X}^T\boldsymbol{X} + \lambda I_p)^{-1}\boldsymbol{X}^T\boldsymbol{X}(\rho\boldsymbol{b}), \\
&= \phi(\boldsymbol{X}^T\boldsymbol{X} + \lambda I_p)^{-1}\boldsymbol{X}^T\boldsymbol{X}\boldsymbol{a} + \rho(\boldsymbol{X}^T\boldsymbol{X} + \lambda I_p)^{-1}\boldsymbol{X}^T\boldsymbol{X}\boldsymbol{b}, \\
&= \phi\boldsymbol{W}_\lambda(\boldsymbol{a}) + \rho\boldsymbol{W}_\lambda(\boldsymbol{b}). \qquad (3.6)
\end{aligned}$$

(3.6) is the condition needed to show that $\boldsymbol{W}_\lambda$ is a linear operator. We now apply $\boldsymbol{W}_\lambda$ to $\hat{\boldsymbol{\beta}}_{OLS}$.

$$\begin{aligned}
\boldsymbol{W}_\lambda\hat{\boldsymbol{\beta}}_{OLS} &= (\boldsymbol{X}^T\boldsymbol{X} + \lambda I_p)^{-1}\boldsymbol{X}^T\boldsymbol{X}(\boldsymbol{X}^T\boldsymbol{X})^{-1}\boldsymbol{X}^TY, \\
&= (\boldsymbol{X}^T\boldsymbol{X} + \lambda I_p)^{-1}\boldsymbol{X}^TY, \\
&= \hat{\boldsymbol{\beta}}(\lambda).
\end{aligned}$$

Hence, $\mathbb{V}ar(\hat{\boldsymbol{\beta}}(\lambda)) = \mathbb{V}ar(\boldsymbol{W}_{\boldsymbol{\lambda}}\hat{\boldsymbol{\beta}}_{OLS})$. Using that $\boldsymbol{W}_\lambda$ is linear and non-random we deduce $\mathbb{V}ar(\boldsymbol{W}_{\boldsymbol{\lambda}}\hat{\boldsymbol{\beta}}_{OLS}) = \boldsymbol{W}_{\boldsymbol{\lambda}}\mathbb{V}ar(\hat{\boldsymbol{\beta}}_{OLS})\boldsymbol{W}_{\boldsymbol{\lambda}}^T$. Using $\mathbb{V}ar(\hat{\boldsymbol{\beta}}_{OLS}) = \sigma^2(\boldsymbol{X}^T\boldsymbol{X})^{-1}$:

$$\mathbb{V}ar(\hat{\boldsymbol{\beta}}) - \mathbb{V}ar(\hat{\boldsymbol{\beta}}(\lambda)) = \sigma^2((\boldsymbol{X}^T\boldsymbol{X})^{-1} - \boldsymbol{W}_{\boldsymbol{\lambda}}(\boldsymbol{X}^T\boldsymbol{X})^{-1}\boldsymbol{W}_{\boldsymbol{\lambda}}^T).$$

And with $\boldsymbol{W}_{\boldsymbol{\lambda}}^{-1} := (\boldsymbol{X}^T\boldsymbol{X})^{-1}(\boldsymbol{X}^T\boldsymbol{X}+\lambda I_p)$ alongside $(\boldsymbol{W}_{\boldsymbol{\lambda}}^T)^{-1} := (\boldsymbol{X}^T\boldsymbol{X}+\lambda I_p)(\boldsymbol{X}^T\boldsymbol{X})^{-1}$.

$$
\begin{aligned}
&= \sigma^2\boldsymbol{W}_{\boldsymbol{\lambda}}(\boldsymbol{W}_{\boldsymbol{\lambda}}^{-1}(\boldsymbol{X}^T\boldsymbol{X})^{-1}(\boldsymbol{W}_{\boldsymbol{\lambda}}^T)^{-1} - (\boldsymbol{X}^T\boldsymbol{X})^{-1})\boldsymbol{W}_{\boldsymbol{\lambda}}^T,\\
&= \sigma^2\boldsymbol{W}_{\boldsymbol{\lambda}}((\boldsymbol{X}^T\boldsymbol{X})^{-1}(\boldsymbol{X}^T\boldsymbol{X}+\lambda I_p)(\boldsymbol{X}^T\boldsymbol{X})^{-1}(\boldsymbol{X}^T\boldsymbol{X}+\lambda I_p)(\boldsymbol{X}^T\boldsymbol{X})^{-1} - (\boldsymbol{X}^T\boldsymbol{X})^{-1})\boldsymbol{W}_{\boldsymbol{\lambda}}^T,\\
&= \sigma^2\boldsymbol{W}_{\boldsymbol{\lambda}}((I_p+\lambda(\boldsymbol{X}^T\boldsymbol{X})^{-1})(\boldsymbol{X}^T\boldsymbol{X})^{-1}(I_p+\lambda(\boldsymbol{X}^T\boldsymbol{X})^{-1}) - (\boldsymbol{X}^T\boldsymbol{X})^{-1})\boldsymbol{W}_{\boldsymbol{\lambda}}^T,\\
&= \sigma^2\boldsymbol{W}_{\boldsymbol{\lambda}}(((\boldsymbol{X}^T\boldsymbol{X})^{-1}+\lambda(\boldsymbol{X}^T\boldsymbol{X})^{-2})(I_p+\lambda(\boldsymbol{X}^T\boldsymbol{X})^{-1}) - (\boldsymbol{X}^T\boldsymbol{X})^{-1})\boldsymbol{W}_{\boldsymbol{\lambda}}^T,\\
&= \sigma^2\boldsymbol{W}_{\boldsymbol{\lambda}}((\boldsymbol{X}^T\boldsymbol{X})^{-1}+\lambda(\boldsymbol{X}^T\boldsymbol{X})^{-2}+\lambda(\boldsymbol{X}^T\boldsymbol{X})^{-2}+\lambda^2(\boldsymbol{X}^T\boldsymbol{X})^{-3} - (\boldsymbol{X}^T\boldsymbol{X})^{-1})\boldsymbol{W}_{\boldsymbol{\lambda}}^T,\\
&= \sigma^2\boldsymbol{W}_{\boldsymbol{\lambda}}(2\lambda(\boldsymbol{X}^T\boldsymbol{X})^{-2}+\lambda^2(\boldsymbol{X}^T\boldsymbol{X})^{-3})\boldsymbol{W}_{\boldsymbol{\lambda}}^T,\\
&= \sigma^2(\boldsymbol{X}^T\boldsymbol{X}+\lambda I_p)^{-1}(\boldsymbol{X}^T\boldsymbol{X})^{-1}(2\lambda(\boldsymbol{X}^T\boldsymbol{X})^{-2}+\lambda^2(\boldsymbol{X}^T\boldsymbol{X})^{-3})(\boldsymbol{X}^T\boldsymbol{X})(\boldsymbol{X}^T\boldsymbol{X}+\lambda I_p)^{-1},\\
&= \sigma^2\ \underbrace{(\boldsymbol{X}^T\boldsymbol{X}+\lambda I_p)^{-1}}_{\text{Positive Semi-Definite}}\ \underbrace{(2\lambda I_p+\lambda I_p+\lambda(\boldsymbol{X}^T\boldsymbol{X})^{-1})}_{\text{Positive Semi-Definite}}\ \underbrace{(\boldsymbol{X}^T\boldsymbol{X}+\lambda I_p)^{-1}}_{\text{Positive Semi-Definite}}.
\end{aligned}
$$

Since each of the products in this expression for $\mathbb{V}ar(\hat{\boldsymbol{\beta}}) - \mathbb{V}ar(\hat{\boldsymbol{\beta}}(\lambda))$ are positive semi-definite ($\forall\lambda > 0$), [27] (p.12), we can deduce $\mathbb{V}ar(\hat{\boldsymbol{\beta}}) - \mathbb{V}ar(\hat{\boldsymbol{\beta}}(\lambda)) \geq 0$ $\square$.

And so, as we increase the penalisation parameter, λ, we decrease the variance of our predictors at the expense of introducing bias (specifically towards zero).

Definition 3.2.2. ***The PMSE****, predictive mean square error, of a model is given by the following expression;*

$$PMSE := \frac{1}{n}\sum_{i=1}^{n}(y_i - \boldsymbol{x}_{\boldsymbol{i}}^T\hat{\boldsymbol{\beta}})^2,$$

in which y_i are response values which were not used in training, and $\boldsymbol{X}\hat{\boldsymbol{\beta}}$ is our prediction of the value of y_i given the corresponding testing data $\boldsymbol{x}_{\boldsymbol{i}}^T$.

The PMSE is a measure of the predictive accuracy of a model. Models generally seek to minimise the PMSE. To accurately measure the PMSE of a model we must test it on data which has not been used in the training of the model. If we used the training dataset to calculate the PMSE the returned value be the mean of the residual sum of squares (also known as the mean squared error, MSE) which is minimised by $\hat{\boldsymbol{\beta}}_{OLS}$, which we know can offer sub-optimal estimates of the coefficients. If the model is tested on data which has been used in training, the coefficients will have been changed as a result of those data points. This means that the model has already accounted for the specific data it is being tested on, and so, it performs better.

Example 3.2.3. *Using the code found in 3.2.3 A, we wish to make an inference about which of two models is more useful for prediction. Firstly, we randomly select 80% of our data to use for training our model, leaving the other 20% to use in comparing our models. Once we have separated and scaled our data, we train both a least squares regression model and a ridge regression model. While doing this we only feed our models the training data. Now, we give the models the predictor variables relating to the test set. With this we estimate what the response of the test set will be. After this prediction process, we are ready to calculate the PMSEs of both models, given this particular dataset.*

$$PMSE = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2$$

Where in this case y_i is from the test set of responses, and $\hat{y}_i$ is from the test set of predictors. Running this calculation for both models returns (after readjusting the scale of our predictions),

$$PMSE_{OLS} = 6.80,$$
$$PMSE_{Ridge} = 5.38.$$

These values suggest that the ridge regression model is preferable to the least squares regression model as $PMSE_{Ridge} < PMSE_{OLS}$. This is what we would expect due to the least squares regression model being vulnerable to overfitting and collinearity. These factors lead to the respective values of $\hat{\boldsymbol{\beta}}$ being inflated. This inflation is tackled by the penalisation parameter present in the ridge regression model's formula for $\hat{\boldsymbol{\beta}}$.

3.3 Lasso Regression

Lasso regression, similar to ridge regression involves minimising a loss function dependent on the penalisation parameter, λ.

Definition 3.3.1. ***The lasso loss function*** *is given by*

$$L_{lasso}(\boldsymbol{\beta}; \lambda) = \|\boldsymbol{Y} - \boldsymbol{X}\boldsymbol{\beta}\|_2^2 + \lambda\|\boldsymbol{\beta}\|_1, \tag{3.7}$$

and we pick $\hat{\boldsymbol{\beta}}_{lasso}$ such that,

$$\begin{aligned}\hat{\boldsymbol{\beta}}(\lambda) &= \arg\min_{\boldsymbol{\beta}} \|\boldsymbol{Y} - \boldsymbol{X}\boldsymbol{\beta}\|_2^2 + \lambda\|\boldsymbol{\beta}\|_1 \ , \\ &= \arg\min_{\boldsymbol{\beta}} \sum_{i=1}^{n}(y_i - \boldsymbol{x_i}^T\boldsymbol{\beta})^2 + \lambda\sum_{j=1}^{p}|\beta_j|,\end{aligned}$$

*[23](p.2), where $\boldsymbol{x}_i^T$ denotes the i^{th} **row** vector of $\boldsymbol{X}$ (which will have dimensions $1 \times p$).*

As with ridge regression (3.1), λ can be adjusted for best results, typically via cross-validation. Cross-Validation (CV) splits up the data used to train the model into smaller training and testing sets (the number of splits considered within the data is known as the number of folds). More folds leads to better performance, but requires more computation power. CV seeks to improve various performance metrics (such as PMSE and MSE) using training sets within the data supplied to the model with comparison to test sets also within the data given to model. The intricacies of CV are complicated and beyond the scope of this project, see [21] for more information.

The Lasso Estimator was first proposed in 1996 by Robert Tibshirani, [23]. It has various advantages over the original ridge estimator, principally its ability to perform parameter selection. This means that as the penalty parameter, λ, the less significant coefficients, $\hat{\beta}_j$ are set to zero. This is different to ridge regression which keeps shrinking the parameters but fails to set coefficients to zero. Parameter selection is very useful for making inference, as if we can reduce the number of predictors, p, then we can better understand the underlying relationship between the significant predictors and the response variable. Furthermore, when used in conjunction with CV, it has been found recently that even better predictive performance can be achieved in a process known as model relaxation. This process involves running an initial lasso and increasing λ, eliminating insignificant predictors, and then executing the lasso on the dataset again, but having removed the less significant predictors from the start of analysis at $\lambda = 0$, see [25]. The use of the Lasso to create a more parsimonious model is very powerful, and this property is one advantage of the Frequentist approach to linear regression over the Bayesian approach which fails to automatically perform variable selection. However, the Bayesian approach to linear regression has many benefits which we explore in the next few chapters of this project. We will highlight the methodology behind Bayesian linear regression, and how shrinkage priors placed on our coefficients can improve regression similarly to frequentist penalisation methods. The frequentist ridge and lasso methods we have already outlined make for a good baseline of linear regression models for which to compare their Bayesian equivalents.

Chapter 4

Bayesian Linear Regression

4.1 Bayesian Methodology

In this chapter we demonstrate how we can apply the Bayesian methodology to linear regression, starting with a simple model and gradually building up theory so that we can obtain powerful models. The main difference between the Bayesian models and the traditional models is in how we treat the coefficients. Previously, we used deterministic formulae to determine our estimators of $\boldsymbol{\beta}$. This changes in the Bayesian case, where we now consider $\boldsymbol{\beta}$ to be a random variable, with its own prior distribution. The 'true' values of $\boldsymbol{\beta}$ are considered to be a realisation of the posterior distribution for $\boldsymbol{\beta}$ in the Bayesian formalism. Our choices of prior distributions for our coefficients impacts the posterior distribution for $\boldsymbol{\beta}$. We will particularly analyse the cases in which we apply shrinkage priors to $\boldsymbol{\beta}$. We begin by reiterating the assumptions of the linear model. Recall the model:

$$\boldsymbol{Y} = \boldsymbol{X}\boldsymbol{\beta} + \boldsymbol{\epsilon},$$

where $\boldsymbol{X}$ represents the $n \times p$ data matrix and $\boldsymbol{\beta} \in \mathbb{R}^p$ contains the information about the relationship between the data and the response, and $\boldsymbol{\epsilon}$ is the $n \times 1$ matrix representing the inherent error in the model. This error is distributed such that its i components $\epsilon_i \overset{iid}{\sim} N(0, \sigma^2)$. Using this information we can say that[1],

$$\boldsymbol{y}|\boldsymbol{\beta}, \sigma^2 \sim N(\boldsymbol{X}\boldsymbol{\beta}, \sigma^2 I_n). \tag{4.1}$$

As with the frequentist case, we are interested in estimating the parameter $\boldsymbol{\beta}$. OLS regression used the maximum likelihood estimate of $\boldsymbol{\beta}$ for this, however, in the Bayesian case we consider each β_j for $j \in \{1, \dots, p\}$ to be a random variable which is distributed according to an unknown distribution. In Bayesian linear regression, we attempt to learn about this unknown distribution using Bayes' Theorem, which tells us that[14](p.59),

$$p(\boldsymbol{\beta}|\boldsymbol{y}) = \frac{f(\boldsymbol{y}|\,\boldsymbol{\beta})p(\boldsymbol{\beta})}{f(\boldsymbol{y})}.$$

This is interpreted as follows:

- $p(\boldsymbol{\beta}|\boldsymbol{y})$ is the posterior distribution of $\boldsymbol{\beta}$, and making inferences about $\boldsymbol{\beta}$ from this probability density function (pdf) is our goal. The posterior, $p(\boldsymbol{\beta}|\boldsymbol{y})$, can be viewed as a compromise between our prior beliefs, and the likelihood, with the likelihood's influence on the posterior increasing as we observe more training data
- $f(\boldsymbol{y}|\,\boldsymbol{\beta})$ is the likelihood of $\boldsymbol{y}$. It is important to note that this function is not a pdf, rather it represents a distribution. As such it is not restricted to taking values

[1] In the Bayesian formalism, it is more common to refer to the response vector, $\boldsymbol{Y}$, using the lower case $\boldsymbol{y}$. We shall switch to this notation, but note that they have identical meanings to us.

between zero and one. In linear regression this can be found using (4.1). Using the assumptions of the linear model, we assume that each y_i component of $\boldsymbol{y}$ is independent of one another. Hence, the likelihood of $\boldsymbol{y}$ can be written as a product of the y_i normal distributions. $y_i \sim N(\boldsymbol{x}_i^T\boldsymbol{\beta}, \sigma^2)$ (where $\boldsymbol{x}_i$ represents the i^{th} row vector of $\boldsymbol{X}$). As such,

$$f(\boldsymbol{y}|\,\boldsymbol{\beta}) = \prod_{i=1}^{n} \frac{1}{\sigma\sqrt{2\pi}} \exp\left(-\frac{1}{2\sigma^2}(y_i - \boldsymbol{x}_i^T\boldsymbol{\beta})^2\right). \tag{4.2}$$

- $p(\boldsymbol{\beta})$ is the pdf which represents our prior beliefs about $\boldsymbol{\beta}$. We can pick many options for this pdf, and it is the mechanism through which we introduce coefficient shrinkage in the Bayesian paradigm
- $f(\boldsymbol{y})$ is the normalising constant. This constant ensures $\int_{-\infty}^{+\infty} p(\boldsymbol{\beta}|\boldsymbol{y})\, d\boldsymbol{\beta} = 1$. It can be written as $\int_{-\infty}^{+\infty} f(\boldsymbol{y}|\,\boldsymbol{\beta})p(\boldsymbol{\beta})\, d\boldsymbol{\beta}$. This term is usually ignored because using Markov chain Monte Carlo (MCMC) methods, which we describe later, sample directly from the posterior and so normalisation is not a concern to us

Definition 4.1.1. ***The Maximum A Posteriori (MAP)*** *estimate of a random variable, for instance $\boldsymbol{\beta}$, is written as $\hat{\boldsymbol{\beta}}_{MAP}$ and is the value of $\boldsymbol{\beta}$ which maximises it's respective posterior distribution $p(\boldsymbol{\beta}|\boldsymbol{y})$.*

$$\begin{aligned}\hat{\boldsymbol{\beta}}_{MAP} :&= \arg\max_{\boldsymbol{\beta}} p(\boldsymbol{\beta}|\boldsymbol{y}),\\ &= \arg\max_{\boldsymbol{\beta}} \frac{f(\boldsymbol{y}|\boldsymbol{\beta})p(\boldsymbol{\beta})}{f(\boldsymbol{y})}.\end{aligned}$$

Since $f(\boldsymbol{y})$ is independent of $\boldsymbol{\beta}$, this is equivalent to:

$$\hat{\boldsymbol{\beta}}_{MAP} = \arg\max_{\boldsymbol{\beta}} f(\boldsymbol{y}|\boldsymbol{\beta})p(\boldsymbol{\beta}). \tag{4.3}$$

This is very similar to taking the maximum likelihood estimate, the difference being that we multiply the likelihood by the prior, and seek to maximise this product, rather than maximising the likelihood alone, [26] (p.4). The MAP estimate of $\boldsymbol{\beta}$ is useful in analysing the behaviour of our posterior distribution. However, it is important to note that we still believe that $\boldsymbol{\beta}$ is a random realisation of our posterior distribution, the MAP estimate is merely the value of $\boldsymbol{\beta}$ with the highest posterior density. In fact, the continuous nature of the posterior pdf implies that $\mathbb{P}(\boldsymbol{\beta} = \hat{\boldsymbol{\beta}}_{MAP}) = 0$, as the chance that a realisation of the posterior distribution for $\boldsymbol{\beta}$ takes any given singular value is zero.

4.2 Choosing Priors

When it comes to picking a prior distribution, $p(\beta)$, we have unlimited possibilities. Our choice of prior depends on what we wish to achieve with our model, and our data. A popular choice is that of the flat prior [13], $p(\boldsymbol{\beta}) \propto 1$. This prior choice is not a valid probability density function, as $\int_{\mathbb{R}^p} p(\boldsymbol{\beta})d\boldsymbol{\beta}$ does not converge, and so, it is known as an improper prior density. Despite being improper it usually leads to valid posterior distributions. These posterior distributions are exclusively formed by the data in the likelihood, and so, we find that,

$$\hat{\boldsymbol{\beta}}_{MAP} = \hat{\boldsymbol{\beta}}_{MLE} = \arg\max_{\boldsymbol{\beta}} f(\boldsymbol{y}|\boldsymbol{\beta}).$$

It is not uncommon for these priors to be chosen naively, believing them to be the most objective reflection of reality[2]. Due to their improperness, the posterior distribution must also be checked for properness. To ensure properness but still have a posterior which is very heavily data dependent, sometimes people use a prior such as $\beta_j \sim N(0, 1000^2)$ or another very flat appearing diffuse prior. These priors are subject to the same problems as MLE estimates from the normal equation, (2.1), such as overfitting and collinearity inflating estimates of $\boldsymbol{\beta}$. Furthermore, we argue that the belief that these priors are objective reflections of reality is flawed.

We will now discuss the philosophy of our prior choices. Similar ideas about prior choices to the following are discussed here [13] (within the context of ecology). We argue that useful models tend to be based upon multiple assumptions. In the case of linear regression, we assume that our response variables, y_i are independent and normally distributed. This assumption is rarely flawless, however, it allows us to make useful inferences about relationships between predictors and responses. These assumptions themselves can be viewed as "prior beliefs", even in the frequentist case, and as we explored in chapter 3, adding even more "prior beliefs" can greatly improve model performance. In chapter 3 the additional "prior beliefs" were that we want to pick the $\hat{\boldsymbol{\beta}}$ which maximises the likelihood, while not being too large (this is the purpose of the penalty term). And so, as in the frequentist case, in the Bayesian case we consider choices of prior not based upon their "subjectivity" or "objectivity" (the degree to which any given prior choice is either of these is itself subjective), rather we consider prior choices based on their usefulness and results in modelling. Different priors are appropriate in different settings; and assumptions we make can greatly influence our posterior distributions, but this is not itself a bad thing. To quote the late George Box, "All models are wrong, but some are useful".

We can pick informative priors, which specifically pull posterior values of $\boldsymbol{\beta}$ towards certain values perceived to be likely based upon expert advice or previous experiments. These priors need justification if used, but can be very helpful especially in scenarios with not many observations. A common method of determining these priors is through expert elicitation [2]. Also, it should be noted that it is possible to naively pick an informative prior based upon the data which we have observed and are trying to model. This is a mistake as it results in the data being represented twice, once in the likelihood, and once in the prior. This can lead to a false confidence in the posterior model parameters' distributions.

We can also pick weakly informative priors, [13] (p.2). These priors may not be picked because they provide valid information for a certain dataset, rather, they have general helpful properties. Shrinkage priors are an example of weakly informative priors as they do not introduce much unique information to our model. Rather, they are used to combat common issues with regression. We use them to greatest effect when we have a prior belief that our model's estimates of the β_j coefficients will likely be inflated. We shall apply the principles of penalised regression in a Bayesian context, through weakly informative priors.

In the frequentist paradigm, the addition of the penalty term in the ridge and lasso estimators can be seen as slightly arbitrary as it raises the question; 'why add on these specific penalty terms and not others?'. Meanwhile, Bayesians have a natural way to include the penalisation of coefficients; through the prior distributions. Through MAP estimates many Bayesian penalising priors have clear links to popular frequentist penalisation techniques. Furthermore, these connections arise often by placing fairly intuitive prior distributions upon β_j, leading to pleasant relationships between common distributions and penalisation. We first investigate the simplest, and most intuitive Bayesian penalising prior, the ridge prior, after a motivating example.

Example 4.2.1. *This example highlights an issue with constant and uninformative priors.*

[2] In fact, Stan (a popular probabilistic programming language) takes flat priors by default for estimated parameters (such as $\boldsymbol{\beta}$ in our case) unless the code specifies a different prior.

We generate data, $\boldsymbol{X}_{ij} \sim N(0,1)$, with $i \in \{1, \dots, 8\}$ (so we have 8 observations, $n = 8$), and $j \in \{1, , \dots, 10\}$ (with 10 predictor variables, $p = 10$). This data is connected to a response vector $\boldsymbol{y}$ through the equation $\boldsymbol{y} = \boldsymbol{X}\boldsymbol{\beta} + \boldsymbol{\epsilon}$, with each component of $\boldsymbol{\beta}$, $\beta_j = 5$, and $\epsilon_i \overset{iid}{\sim} N(0, 3^2)$. We fit a standard linear model, a frequentist ridge model, a Bayesian ridge model (introduced in the following section) and a non-informative flat prior Bayesian model (all without intercept terms).

Regression	$\hat{\beta}_1$	$\hat{\beta}_2$	$\hat{\beta}_3$	$\hat{\beta}_4$	$\hat{\beta}_5$	$\hat{\beta}_6$	$\hat{\beta}_7$	$\hat{\beta}_8$	$\hat{\beta}_9$	$\hat{\beta}_{10}$
OLS	-1.6	23.9	-0.5	11.3	11.3	-13.0	9.33	2.6	NA	NA
Ridge	2.1	2.9	1.6	3.5	1.0	2.5	1.0	1.0	0.9	2.8
Bayes' Ridge	6.3	4.9	4.1	6.1	5.8	1.8	4	3.4	6.0	6.0
Bayesian (flat prior)	-1.3×10^{14}	6.2×10^{14}	9.8×10^{13}	6.5×10^{14}	1.2×10^{14}	-1.2×10^{15}	3.9×10^{14}	-2.9×10^{14}	-1.1×10^{14}	-2.6×10^{14}

Table 4.1: Table of obtained $\boldsymbol{\beta}$ estimates. For Bayes' regressions a mean of posterior samples is reported.

Firstly, we shall begin our analysis of the above table by considering the ordinary least squares estimates of the coefficients. Recalling that the true values of $\beta_j = 5$, $\forall j \in \{1, \dots, 10\}$, we see that these estimates are not very good. Furthermore, it fails to estimate β_9 and β_{10}. This is because in RStudio, the function, $lm(\cdot)$, uses the normal equation, (2.1), to calculate estimate $\hat{\boldsymbol{\beta}}$. The problem is that since $p > n$ in this case, we have that the $\boldsymbol{X}^T\boldsymbol{X}$ term required for the calculation of the normal equation is rank deficient. Hence, the solution does not exist as $(\boldsymbol{X}^T\boldsymbol{X})^{-1}$ does not exist. The $lm(\cdot)$ function deals with this by crudely pretending that β_9 and β_{10} are zero. This effectively removes the 9^{th} and 10^{th} columns of $\boldsymbol{X}$, removing the rank deficiency by discarding both data columns, creating a new simplified data matrix $\tilde{\boldsymbol{X}}$ and using this to find coefficient estimates. Now we consider the Bayesian flat prior regression. We know that the posterior modes for each β_j will be the MLE estimates $\hat{\beta}_j$ given by the normal equation. We have already seen that because $p > n$ in this case, the MLE estimate does not exist for all of our β_j coefficients. The relationship between the MAP of the flat prior regression and the MLE in OLS regression appears to cause issues in determining coefficients in the case where $p > n$, resulting in huge and non-sensicle coefficient estimates. This highlights the importance of prior selection in Bayesian regression, especially in low-data scenarios. The Frequentist ridge manages to estimate each parameter due to the inclusion of the penalty term in the calculation of $\hat{\beta}_j$ which means we use $(\boldsymbol{X}^T\boldsymbol{X} + \lambda I)$ in our calculation as opposed to the singular $\boldsymbol{X}^T\boldsymbol{X}$. This gives us answers for each $\hat{\beta}_j$ which are not too far off of the correct $\beta_j = 5$. However, we see that when we use the prior associated with ridge regression for our Bayesian model, we achieve a very fair estimate of $\boldsymbol{\beta}$ considering the low sample size relative to the number of covariates we have. The difference between the two Bayesian models we employed in this example are night and day, and this difference is solely due to using different prior distributions for our β_j. This stark improvement caused by our choice in prior, and strength compared to traditional frequentist penalised regression methods is why the remainder of this project is dedicated to our choices of prior distributions with respect to various parameters within Bayesian regression, and how they impact their respective posterior distributions.

We begin our exploration of Bayesian penalisation with the simplest penalising prior for β_j, the ridge prior, after a note about how we will be treating the intercept term within this project.

4.3 Bayesian Ridge

Note about the intercept term, β_0 (more commonly referred to as α in Bayesian regression). We do not assign β_0 the same penalising prior distributions as we do for β_j, $\forall j \in \{1, \dots, p\}$. This is due to the fact that β_0 will not suffer from the inflating effects of overfitting and

collinearity as the other coefficients will. As such, we recommend when doing Bayesian regression in practice, assigning β_0 a prior density based upon expert information about the given data or, using a non-informative prior such as a flat prior. In this project, to more fairly and simply compare different penalising techniques, we assume that the intercept is 0 (and not included in our model unless stated otherwise). For this assumption to be valid, we shall generate data such that the mean response value is zero ($\bar{\boldsymbol{y}} = 0$) when we do not include an intercept in our model. When we will use an intercept term, the assumption of a flat prior is useful for our purposes as we can focus more exclusively on prior penalisation, without needing to concern ourselves with how the intercept term impacts our model.

Consider a normal prior distribution for $\beta_j|\lambda,\, \sigma^2 \sim N(0, \frac{\sigma^2}{\lambda})\ \ \forall j \in \{1, \ldots, p\}$. We wish to find our MAP estimate of $\boldsymbol{\beta}$ under this prior.

$$\begin{aligned}\hat{\boldsymbol{\beta}}_{MAP} &= \arg\max_{\boldsymbol{\beta}} f(\boldsymbol{y}|\boldsymbol{\beta})p(\boldsymbol{\beta}|\lambda,\sigma^2), \qquad (4.3)\\ &= \arg\max_{\boldsymbol{\beta}}[\log(f(\boldsymbol{y}|\boldsymbol{\beta})) + \log(p(\boldsymbol{\beta}|\lambda,\sigma^2))].\end{aligned}$$

Firstly, consider $\log(f(\boldsymbol{y}|\boldsymbol{\beta}))$,

$$\begin{aligned}\log(f(\boldsymbol{y}|\boldsymbol{\beta})) &= \log\left[\prod_{i=1}^{n}\frac{1}{\sigma\sqrt{2\pi}}\exp\left(-\frac{1}{2\sigma^2}(y_i - \boldsymbol{x}_i^T\boldsymbol{\beta})^2\right)\right],\\ &\propto \log\left[\exp\left(-\frac{1}{2\sigma^2}\sum_{i=1}^{n}(y_i - \boldsymbol{x}_i^T\boldsymbol{\beta})^2\right)\right],\\ &= -\frac{1}{2\sigma^2}\sum_{i=1}^{n}(y_i - \boldsymbol{x}_i^T\boldsymbol{\beta})^2,\end{aligned}$$

where we have eliminated constants which are independent of $\boldsymbol{\beta}$. Now, consider $\log(p(\boldsymbol{\beta}|\lambda,\sigma^2))$,

$$\begin{aligned}\log(p(\boldsymbol{\beta}|\lambda,\sigma^2)) &= \log(p(\beta_1,\, ...,\, \beta_p|\lambda,\, \sigma^2)),\\ &= \log(p(\beta_1|\lambda,\, \sigma^2)\cdots p(\beta_p|\lambda,\, \sigma^2)),\ \text{[as the priors for } \beta_j \text{ are independent of each other]},\\ &= \log\left(\prod_{j=1}^{p}p(\beta_j|\lambda,\sigma^2)\right),\\ &= \log\left[\prod_{j=1}^{p}\frac{\sqrt{\lambda}}{\sigma\sqrt{2\pi}}\exp\left(\frac{-\lambda}{2\sigma^2}\beta_j^2\right)\right],\\ &\propto -\frac{\lambda}{2\sigma^2}\sum_{j=1}^{p}(\beta_j)^2.\end{aligned}$$

Substituting these into our $\hat{\boldsymbol{\beta}}_{MAP}$ equation gives,

$$\begin{aligned}\hat{\boldsymbol{\beta}}_{MAP} &= \arg\max_{\boldsymbol{\beta}} -\frac{1}{2\sigma^2}\sum_{i=1}^{n}(y_i - \boldsymbol{x}_i^T\boldsymbol{\beta})^2 - \frac{\lambda}{2\sigma^2}\sum_{j=1}^{p}\beta_j^2,\\ &= \arg\min_{\boldsymbol{\beta}} \frac{1}{2\sigma^2}\sum_{i=1}^{n}(y_i - \boldsymbol{x}_i^2\boldsymbol{\beta})^2 + \frac{\lambda}{2\sigma^2}\sum_{j=1}^{p}\beta_j^2,\\ &= \arg\min_{\boldsymbol{\beta}} \|\boldsymbol{Y} - \boldsymbol{X}\boldsymbol{\beta}\|_2^2 + \lambda\|\boldsymbol{\beta}\|_2^2. \qquad (4.4)\end{aligned}$$

Notice that the equation for the MAP of the Bayesian ridge estimator is identical to the MLE of the frequentist ridge loss function, this tells us that in the Bayesian setting,

using the normal distribution specified above as a prior for each β_j results in an identical minimisation problem as the Frequentist ridge case. Note: in general, when we will refer to the Bayesian version of a traditional penalisation technique, such as lasso, we are implying that the Bayesian maximum a posteriori estimate is equivalent to the choice of $\hat{\beta}_j$ which minimises the associated frequentist loss function. For example in the Bayesian lasso we expect;

$$\hat{\boldsymbol{\beta}}_{\text{MAP (lasso)}} = \arg\min_{\boldsymbol{\beta}} \|\boldsymbol{Y} - \boldsymbol{X}\boldsymbol{\beta}\|_2^2 + \lambda\|\boldsymbol{\beta}\|_1.$$

This means we should expect the penalisation achieved by both approaches to be theoretically similar between the two different paradigms. In reality, the results achieved by both techniques will differ, due to the differences in methodology and analysis of results. The impact of varying λ in ridge regression has clear parallels with the Frequentist case.

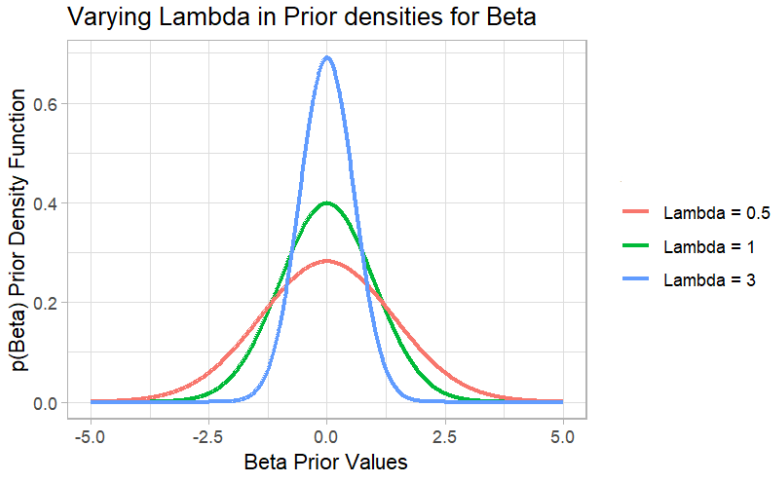

Figure 4.1: A graph of the impact of the shrinkage parameter, λ, on the prior density.

When we increase λ, we are effectively increasing our confidence in the fact that β_j is near to zero. This has the impact of shrinking our $\boldsymbol{\beta}$ parameters. However, looking at the prior distribution for $\beta_j|\lambda$, $\sigma^2 \sim N(0, \frac{\sigma^2}{\lambda})$, we notice that for $\lambda \in (0, 1)$, although our prior is still centered on zero, we will be increasing the variance of our prior. Picking λ very low indicates low confidence in β_j being small. Note: Within the Bayesian paradigm we cannot set $\lambda = 0$ as this would result in an undefined variance, unlike in the Frequentist case, however, the closer λ is to zero, the more similar the prior will perform to a flat prior.

4.3.1 Dealing with σ^2

The ridge prior, $\beta|\sigma^2, \lambda \sim N(0, \frac{\sigma^2}{\lambda})$, is dependent upon the value of σ^2. In other words, to estimate $\boldsymbol{\beta}$ using this prior, we must also estimate the error in our model. As such, we estimate σ^2 and so must assign it a prior and involve it in the equation for our posterior as follows:

$$p(\boldsymbol{\beta}, \sigma^2|\lambda, \boldsymbol{y}) = \frac{f(\boldsymbol{y}|\,\boldsymbol{\beta}, \sigma^2)p(\boldsymbol{\beta}|\,\sigma^2, \lambda)p(\sigma^2)}{f(\boldsymbol{y})},$$

in which we have a prior for σ^2, $p(\sigma^2)$, and our posterior becomes joint with respect to $\boldsymbol{\beta}$ and σ^2, $p(\boldsymbol{\beta}, \sigma^2|\lambda, \boldsymbol{y})$. A popular prior specification for σ^2 is the inverse-gamma distribution, and it is also helpful as it results in a tractable joint posterior with our ridge prior, meaning that we can manually derive an expression for our joint posterior distribution. Figure 4.2 gives a look at the shape of this prior distribution which is commonly placed upon σ^2. A higher value for a increases the size of the peak near zero, while larger b makes the tail of the distribution more pronounced. Practically, this means

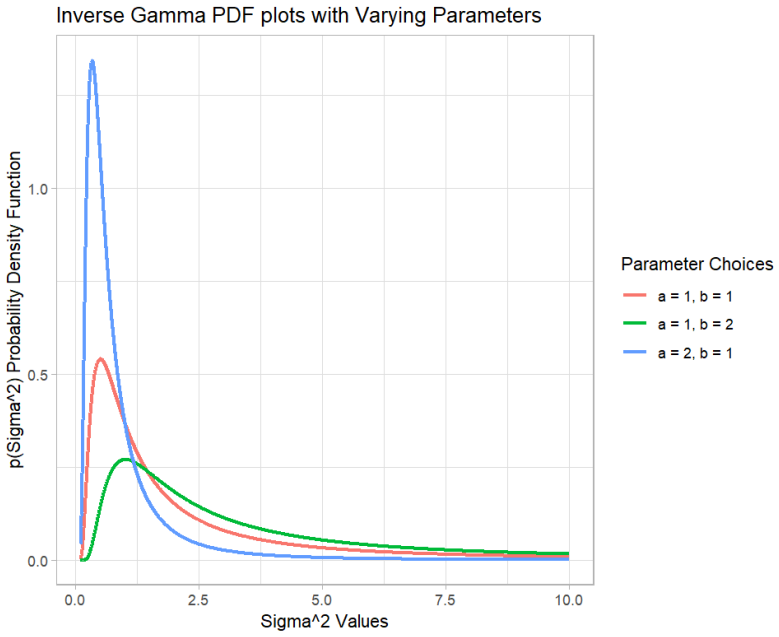

Figure 4.2: A figure to show the impacts of varying the parameters of the inverse-gamma distribution.

that if we are confident in our variance being relatively small, we will want a larger value for a, however, if we are very uncertain, a bigger value for b will be appropriate.

4.3.2 The Normal-Inverse-Gamma (NIG) Prior

The NIG prior refers to the case in which we are using a normal prior for $\boldsymbol{\beta}$ (such as the ridge prior), and an inverse-gamma prior for σ^2. This prior specification is conjugate, and so we can solve for a closed form of our posterior distribution.

$$\begin{aligned} p(\boldsymbol{\beta}, \sigma^2|\lambda) &= p(\boldsymbol{\beta}|\sigma^2, \lambda)p(\sigma^2), \\ &= \prod_{j=1}^{p} \left[\left(\frac{\lambda}{\sigma^2 2\pi} \right)^{\frac{1}{2}} \exp\left(\frac{\lambda}{2\sigma^2}(\beta_j)^2 \right) \right] \times \frac{b^a}{\Gamma(a)} \left(\frac{1}{\sigma^2} \right)^{a+1} \exp\left(\frac{b}{\sigma^2} \right), \\ &\propto \left(\frac{1}{\sigma^2} \right)^{a+\frac{p}{2}+1} \exp\left(-\frac{1}{\sigma^2}(b + \frac{\lambda}{2}\boldsymbol{\beta}^T\boldsymbol{\beta}) \right). \end{aligned} \tag{4.5}$$

And, the likelihood is given by;

$$f(\boldsymbol{y}|\boldsymbol{\beta}, \sigma^2) = \left[\frac{1}{\sigma\sqrt{2\pi}} \right]^n \exp\left(-\frac{1}{2\sigma^2} \sum_{i=1}^{n} (y_i - \boldsymbol{x}_i^T\boldsymbol{\beta})^2 \right). \tag{4.6}$$

Taking the product of (4.5) and (4.6) and using the multivariate completion of squares [3] (p.2):

$$\boldsymbol{u}^T A\boldsymbol{u} - 2\boldsymbol{\alpha}^T\boldsymbol{u} = (\boldsymbol{u} - A^{-1}\boldsymbol{\alpha})^T A(\boldsymbol{u} - A^{-1}\boldsymbol{\alpha}) - \boldsymbol{\alpha}^T A^{-1}\boldsymbol{\alpha},$$

we find the posterior density for these specifications $p(\boldsymbol{\beta}, \sigma^2|\boldsymbol{y}, \lambda)$ is of the same NIG form as our prior, with the following transformed parameters:

$$p(\boldsymbol{\beta}, \sigma^2|\boldsymbol{y}, \lambda) = \left(\frac{1}{\sigma^2} \right)^{a^*+\frac{p}{2}+1} \exp\left(-\frac{1}{\sigma^2}(b^* + \frac{1}{2}(\boldsymbol{\beta} - \boldsymbol{\mu}^*)^T V^{*-1}(\boldsymbol{\beta} - \boldsymbol{\mu}^*) \right). \tag{4.7}$$

The transformed parameters are given as follows:

$$\mu^* = \left(\frac{\lambda}{\sigma^2} I_p + \boldsymbol{X}^T \boldsymbol{X}\right)^{-1} \boldsymbol{X}^T \boldsymbol{y},$$
$$V^* = \left(\frac{\lambda}{\sigma^2} I_p + \boldsymbol{X}^T \boldsymbol{X}\right)^{-1},$$
$$a^* = a + \frac{n}{2},$$
$$b^* = b + \frac{1}{2}(\boldsymbol{y}^T \boldsymbol{y} - \mu^{*T} V^{*-1} \mu^*) \text{ [3].}$$

We have now derived the joint posterior for this prior specification, and we notice that it is of the same form as our prior specification, the NIG prior. This is a result of our prior exhibiting conjugacy. Using this we can immediately sample from our posterior, as we just need to sample from the same distributions specified in our prior, but using our updated parameters listed above. Hence, we sample (for $l = \{1, 2, ...\}$ iterations)

$$\sigma^{2(l)} \sim IG(a^*, b^*),$$
$$\boldsymbol{\beta}^{(l)} \sim MVN(\boldsymbol{\mu}^*, \sigma^{2(l)} V^*) \text{ [3].}$$

Using these samples we can perform Bayesian inference on our posterior parameters. Often the most useful inference we can do is determining which of our β_j coefficient parameters are insignificant (if there are any). If we determine a parameter is insignificant we can decide to remove it from our model. In the Bayesian paradigm a useful method for diagnosing the significance of our coefficients is the credible interval.

Example 4.3.1. *Consider data generated, using 4.3.1 A, such that* $\boldsymbol{X}_{i,j} \sim N(0, 1)$, $i \in \{1, \ldots, 100\}$, $j \in \{1, 2, 3\}$. *We will set our error term such that,* $\epsilon \sim N(0, 3)$. *Note that this error is quite large relative to our generated data, we expect this to reduce the accuracy of our model. We generate our response vector,* $\boldsymbol{y}$, *using the equation;*

$$\boldsymbol{y} = \boldsymbol{X}\boldsymbol{\beta} + \boldsymbol{\epsilon},$$
$$\beta_1 = 0.5,\ \beta_2 = 3, \beta_3 = 0.$$

Using an NIG prior, with code 4.3.1 Stan, with $\beta \overset{iid}{\sim} N(0, \sigma^2/\lambda)$, *for now using* $\lambda = 1$, *and* $\sigma^2 \sim IG(0.5, 2)$, *we generate the following results;*

Coefficient	Mean	Std. Dev.	2.5%	97.5%	25%	75%	True
β_1	0.40	0.27	-0.11	0.92	0.23	0.58	0.50
β_2	2.00	0.29	1.42	2.58	1.80	2.20	2.00
β_3	0.18	0.25	-0.32	0.68	0.01	0.35	0.00

Table 4.2: The percentages are to be interpreted as the probability that our given coefficient is less than the value listed. For example $\mathbb{P}(\beta_1 \leq -0.11) = 2.5\%$. Hence, the 95% credible interval for β_1 will be $[-0.11, 0.92]$. This means that $\mathbb{P}(-0.11 \leq \beta_1 \leq 0.92) = 0.95$.

The table above highlights the caution we must exercise when performing inference around our estimated parameters. If we solely used 95% *credible intervals to determine which coefficients were significant we would say that both* β_1 *and* β_3 *are insignificant. This would be incorrect as* $\beta_1 = 0.5 \neq 0$. *However, if we decided that* 95% *credible intervals were too large and switched to* 90% *intervals we would run into the opposite problem. We would correctly identify that* β_1 *is significant, however, we would also conclude that* β_3 *is significant. In a model with these specifications, it is generally safer to keep parameters we are uncertain about within our model.*

Before we examine more penalising prior distributions and their various features, we shall consider hierarchical models. The ideas of hierarchical models are necessary in deriving more complicated prior distributions, and often form intuitive relationships with common frequentist models through marginalisation.

Chapter 5

General Bayesian Shrinkage Priors

5.1 Hierarchies in Bayesian Penalised Regression

Definition 5.1.1. ***A hierarchical model*** *is a statistical model in which we tune our model using parameters which share dependencies with other parameters based upon a group which they belong to, where we have more than one group.*

Many Bayesian shrinkage priors rely on hierarchies in their specifications and or derivations. These hierarchical representations of shrinkage priors are used to relate the minimisation solutions of frequentist penalisation methods to Bayesian shrinkage prior MAPs. However, if it is possible, we generally prefer to model using a non-hierarchical specification of a model as hierarchies decrease the efficiency of our sampling algorithms. Introducing hierarchies to our prior specifications can often lead to differing penalisation. Consider the following hierarchical model, using a normal prior, and using the full-Bayes method of λ estimation. We specify our full model, [20] (assuming α flat prior):

$$\begin{aligned}
\boldsymbol{y}|\boldsymbol{\beta}, \sigma^2 &\sim N_n(\alpha\mathbb{1} + \boldsymbol{X}\boldsymbol{\beta}, \sigma^2 I_n) \\
\beta_j|\sigma^2, \tau_j^2 &\sim N(0, \sigma^2\tau_j^2) \\
\tau_j^2|\nu, \lambda &\sim IG(\frac{\nu}{2}, \frac{\nu}{2\lambda}) \\
\sigma^2 &\sim IG(\frac{1}{2}, 2) \\
\lambda &\sim \text{half-Cauchy}(0, 1).
\end{aligned}$$

($\mathbb{1}$ is the $n \times 1$ vector of ones, $\mathbb{1} = (1, \ldots, 1)^T$), and α represents the intercept term). We find that if we marginalise out τ_j^2 we find that $\beta_j|\nu, \lambda, \sigma^2 \sim \text{Student-t}(\nu, 0, \sigma^2\lambda^{-1})$, with this being the local student's t distribution, another commonly used prior distribution for our coefficients, [20]. With the introduction of the hierarchical term τ_j, we have changed the shrinkage of our coefficients. Before, we had ridge shrinkage, but now, β_j are shrunk in accordance to the student-t distribution. The performance of the above hierarchical model is identical to the performance of the marginalised version of the model:

$$\begin{aligned}
\boldsymbol{y}|\boldsymbol{\beta}, \sigma^2 &\sim N_n(\alpha\mathbb{1} + \boldsymbol{X}\boldsymbol{\beta}, \sigma^2 I_n) \\
\beta_j|\sigma^2 &\sim \text{Student-t}(\nu, 0, \sigma^2/\lambda) \\
\sigma^2 &\sim IG(\frac{1}{2}, 2) \\
\lambda &\sim \text{half-Cauchy}(0, 1).
\end{aligned}$$

This marginalised version is preferable in practice as we do not have to estimate the additional τ_j, $j \in \{1, \ldots, p\}$ terms. In this case the hierarchy simply provides an interesting link between the ridge prior and the student-t prior, but the hierarchical model itself is impractical. The parameter ν coincides with the degrees of freedom of the t distribution

and larger values for ν result in a more concentrated peak at 0, with lighter tails. The lowest value of ν, $\nu = 1$, coincides with a Cauchy prior for β_j. As ν increases, the Student-t distribution resembles the normal distribution more and more, with $\nu \to \infty$ making the distribution asymptotically normal $N(0, \sigma^2/\lambda)$. The Cauchy prior is considered to have heavy tails. Practically, in regression this increases the chance that β_j can take a large value with less penalisation, if the data gives rise to a likelihood which suggests β_j is large. The Cauchy distribution's tails are proportional to $1/\beta_j^2$, which leads to much slower probability decay for large $|\beta_j|$ than the normal distributions tails which are proportional to $\exp(-\beta_j^2)$.

Example 5.1.2. *We wish to compare the effectiveness of our hierarchical model above and a standard Bayesian ridge regression. We use the data from A.3, but with $n = 10$ training observations, so that the prior has a larger influence on posterior distributions. We employ the full-Bayes method of calculating λ for both the hierarchical model, and in the standard Bayesian ridge regression (which involves treating λ as a random variable to be estimated alongside other parameters, discussed in more detail later), and use an inverse gamma prior for σ^2. We will use $\nu = 1$. As the hierarchical model, with stan code at A.11, is estimating a new vector of values $\boldsymbol{\tau}$, it takes longer to sample from, averaging 10 seconds to achieve 4000 samples over 10 runs, while the non-hierarchical alternative averaged a mere 3 seconds. The non-hierarchical model is approximately $\times 3$ faster. We plot our posterior densities for both models in figure A.15, alongside 90% credible intervals and mean point estimates.*

The posterior densities for our coefficients in this case appear very similar, but we do see slightly heavier tails in the posteriors of the hierarchical model. This could be due to the link between our hierarchical model and the Cauchy distribution. When we marginalise over τ_j^2 in our prior specification we end up with $\beta_j \sim$ Student-t$(\nu = 1, 0, \sigma^2/\lambda)$, which is equivalent to β_j being Cauchy distributed. This results in heavier tails for β_j.

The drawback of hierarchical representations of models is the increased number of parameters our sampler has to estimate. We acquire samples using advanced Gibbs sampling MCMC, [8], techniques which are implemented in Stan, [22], and these sampling techniques take longer and are less efficient when we have to estimate many parameters. Hierarchies by nature introduce many parameters which require estimation, which results in greater sampling time.

5.2 Bayesian Lasso and Elastic Net

The ridge prior is merely one example of a shrinkage prior specification for $\hat{\boldsymbol{\beta}}$. A shrinkage (or penalised) prior is any prior specification for $\boldsymbol{\beta}$ which is symmetrical and centered on zero[1], and so there are theoretically infinitely many possible penalised priors. The ridge priors' normality makes it a great introduction to these priors since it allows for a conjugate posterior distribution, due to the likelihood, $f(\boldsymbol{y}|\,\boldsymbol{\beta}, \sigma^2)$, also being a normal distribution. Many popular shrinkage priors result in penalisation equivalent to popular frequentist methods. Another popular example of this is the Bayesian lasso.

5.2.1 Bayesian Lasso

The Bayesian lasso uses the shrinkage prior, $p(\beta_j|\sigma^2, \lambda) = \frac{\lambda}{2\sqrt{\sigma^2}} \exp\left(-\sqrt{\sigma^{-2}}\lambda|\beta_j|\right)$. This is equivalent to proposing $\beta_j \sim Laplace(0, \sigma/\lambda)$. This prior results in shrinkage equivalent to that of the frequentist loss function minimisation, (3.7). Figure 5.2.1 shows the probability density of the Bayesian lasso prior, with varying values of λ. Notice the much sharper

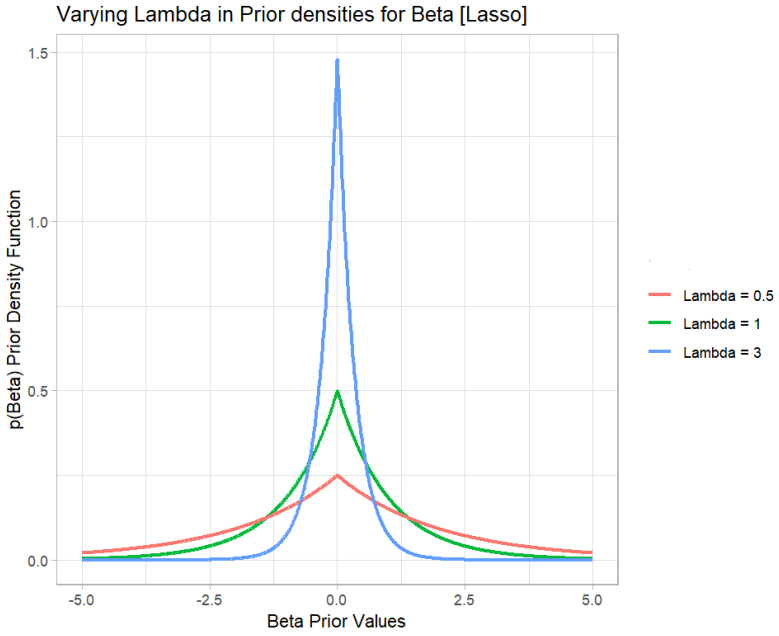

Figure 5.1: Plot to show how a lasso prior for β_j varies with different λ values.

peak at $\beta = 0$ compared to peak in the ridge prior graph, 4.3. This difference in densities results in different posterior coefficient distributions. In practice, this difference in prior shape results in the Bayesian lasso pulling less significant parameters closer to zero as λ increases faster than ridge regression, [17]. While this property can be useful, it is not as powerful as the variable selection provided by frequentist lasso. This variable selection feature is what lead to the original lasso's prevalence. Although the traditional lasso is a very popular method of penalisation, it has several shortcomings, these include (list from [20]):

- In situations with $p > n$, with high multi-collinearity, the ridge estimator often outperforms the lasso
- Dealing poorly with collinearity, when multiple predictors have a collinear relationship the lasso often picks one predictor and sets the others to zero, which can neg-

[1] Centering our prior on zero is consistent with the frequentist ideas of hypothesis testing, where the null hypothesis often coincides with $\beta_j = 0$. Essentially, it can be thought that we are giving some 'weight' to the frequentist null beliefs.

atively impact interpretability (a penalising method which effectively deals with collinearity is said to exhibit grouping)

- It can over-shrink large coefficients

These are issues with the frequentist lasso, but because the Bayesian lasso's MAP estimate of β_j coincides with the frequentist Lasso's $\hat{\beta}_j$ minimisation of the loss function, we expect that the issues associated with the frequentist case are also issues with the Bayesian case. These problems are well encapsulated by the scenario in which we are using the presence of certain genes to infer something about a patient. We often will have many predictor variables, as there are so many genes, and not many observations because it can be difficult/expensive to detect genes ($p \gg n$). Furthermore, many genes will have high correlations between them as several different genes may be related through similar biological 'pathways', [29]. We say these related genes form a group. Ideally we wish to know which genes are insignificant, and which genes belong to significant groups. We hope for grouped predictors to have similar coefficients, and in the scenario where several predictors are identical, their coefficients should also be identical.

These faults in traditional lasso penalisation are remedied by variations of lasso penalisation, and these variations of the frequentist lasso also have their own Bayesian shrinkage prior equivalents. The Bayesian lasso can be represented hierarchically using the following specification:

$$\begin{aligned} y_i|\boldsymbol{\beta}, \sigma^2 &\sim N(\boldsymbol{x_i}^T\boldsymbol{\beta}, \sigma^2) \\ \beta_j|\sigma^2, \tau_j^2 &\sim N(0, \sigma^2\tau_j^2) \\ \tau_j^2|\lambda &\sim \text{Exponential}(\lambda^2/2) \end{aligned}$$

Where σ^2 and λ also have to be determined, but there are many possibilities for them in this case. It is possible to show that when we marginalise the parameter τ_j^2 out of the expression for our joint prior distribution, $p(\beta_j, \tau_j^2|\sigma^2, \lambda) = p(\beta_j|\tau_j^2, \sigma^2)p(\tau_j^2|\lambda)$, we find,

$$\begin{aligned} p(\beta_j|\sigma^2, \lambda) &= \int_0^\infty p(\beta_j|\tau_j^2, \sigma^2)p(\tau_j^2|\lambda)d\tau_j^2, \\ &= \frac{\lambda}{2\sigma}\exp\left(-|\beta_j|\frac{\lambda}{\sigma}\right), \\ &\implies \beta_j \sim \text{Laplace}(0, \frac{\sigma}{\lambda}). \text{ [16] (p.31).} \end{aligned}$$

The fact that we can represent the Bayesian lasso in a hierarchical form allows us to use popular generalisations of the lasso. These generalisations exist to combat the lasso's problems. Using this hierarchical structure we can intuitively introduce several of the frequentist generalisations to the Bayesian paradigm. The first generalisation we mention is also the most popular, and is called the elastic net.

5.2.2 The Elastic Net

The elastic net penalisation technique seeks to be a compromise between ridge regression and lasso regression. This is clearly observable from its loss function:

Definition 5.2.1. ***The elastic net loss function*** *is given by the following expression [29](p.3):*

$$L_{EN}(\boldsymbol{\beta}; \lambda_1, \lambda_2) = \|\boldsymbol{Y} - \boldsymbol{X}\boldsymbol{\beta}\|_2^2 + \lambda_1\|\boldsymbol{\beta}\|_1 + \lambda_2\|\boldsymbol{\beta}\|_2^2, \tag{5.1}$$

with the estimates of $\boldsymbol{\beta}$ whichever values for the components of $\boldsymbol{\beta}$ will minimises this loss function,

$$\hat{\boldsymbol{\beta}}_{EN} = \arg\min_{\boldsymbol{\beta}} \|\boldsymbol{Y} - \boldsymbol{X}\boldsymbol{\beta}\|_2^2 + \lambda_1\|\boldsymbol{\beta}\|_1 + \lambda_2\|\boldsymbol{\beta}\|_2^2. \tag{5.2}$$

In the extreme case where $\lambda_1 = 0$, $\hat{\boldsymbol{\beta}}_{EN} \equiv \hat{\boldsymbol{\beta}}_{ridge}$. Similarly, if $\lambda_2 = 0$, $\hat{\boldsymbol{\beta}}_{EN} \equiv \hat{\boldsymbol{\beta}}_{lasso}$. The addition of the ridge penalty term, $\lambda_2\|\boldsymbol{\beta}\|_2^2$, gives the elastic net performance which is superior to the lasso in situations with $p > n$ and high multicollinearity, where ridge regression usually outperforms the lasso. This is due to its ability to perform grouping. Furthermore, in situations where many predictors are insignificant, it can behave similarly to lasso regression and set coefficients to zero. Because of these factors, the elastic net can be seen as a best-of-both-worlds approach to penalisation, with the disadvantage of needing to account for two penalising parameters, rather than the usual one. This is more computationally expensive, and when using cross-validation often results in over-shrinkage, [20]. This problem can be solved using Bayesian regression. The elastic net can be translated into the Bayesian paradigm using the following hierarchical specification [20],

$$\begin{aligned} y_i|\boldsymbol{\beta}, \sigma^2 &\sim N(\boldsymbol{x_i}^T\boldsymbol{\beta}, \sigma^2) \\ \beta_j|\sigma^2, \tau_j^2, \lambda_1, \lambda_2 &\sim N\left(0, \left(\frac{\lambda_2}{\sigma^2}\frac{\tau_j}{\tau_j - 1}\right)^{-1}\right) \\ \tau_j^2|\lambda_1, \lambda_2, \sigma^2 &\sim \text{Truncated-Gamma}\left(\frac{1}{2}, \frac{8\lambda_2\sigma^2}{\lambda_1^2}\right), \tau_j^2 \in (1, \infty). \end{aligned}$$

Where σ^2, λ_1 and λ_2 also have to be determined, but there are many possibilities for them in this case. When we marginalise our prior distribution $p(\beta_j|\sigma^2, \tau_j^2, \lambda_1, \lambda_2)$ over all the possible values of τ_j^2, we get the resulting density for β_j;

$$p(\beta_j|\sigma^2, \lambda_1, \lambda_2) = C(\sigma^2, \lambda_1, \lambda_2)\exp\left(-\frac{1}{2\sigma^2}(\lambda_1|\beta_j| + \lambda_2\beta_j^2)\right),$$

for $j = 1, 2, \ldots, p$, where $C(\sigma^2, \lambda_1, \lambda_2)$ is the normalising constant. This marginal prior for β_j is where we can gain an intuition for how the values of λ_1 and λ_2 impact our elastic net regression. We notice that, similar to the frequentist case, a relatively large value of λ_1 results in increased lasso characteristics, while λ_2 is related to the ridge characteristics. We expect while using highly grouped datasets with high collinearity between predictor variables, that we will have more ridge characteristics, as this is the scenario in which ridge regression generally outperforms the lasso. Likewise, if we have a sparse dataset, with many insignificant predictors, we would expect λ_1 to be larger, to perform better variable selection.

Example 5.2.2. *(Data generation scheme found in appendix A.16). We wish to compare different frequentist and Bayesian shrinkage regression techniques to test their efficacy in dealing with grouped data. To do this we simulate a dataset with the following specifications (the specifications of this simulation were inspired by [20](p.41)). We simulate* $Z_k \sim N(0, 1)$ *for* $k \in \{1, \ldots, 4\}$ *and group our data based on these* Z_k*. We also simulate some noise for each individual data point in our design matrix* $\boldsymbol{X}$*, given by* $\omega_j \sim N(0, 0.1^2)$ *for* $j \in \{1, \ldots, 50\}$*, with:*

- $X_j = Z_1 + \omega_j$, *for* $j \in \{1, \ldots, 10\}$
- $X_j = Z_2 + \omega_j$, *for* $j \in \{11, \ldots, 20\}$
- $X_j = Z_3 + \omega_j$, *for* $j \in \{21, \ldots, 30\}$
- $X_j = Z_4 + \omega_j$, *for* $j \in \{31, \ldots, 40\}$
- $X_j \sim N(0, 1)$, *for* $j \in \{41, \ldots, 50\}$
- $\boldsymbol{\beta} = (\underbrace{10, \ldots, 10}_{20\text{ times}}, \underbrace{-10, \ldots, -10}_{20\text{ times}}, \underbrace{0, \ldots, 0}_{10\text{ times}})^T$

- $\epsilon_i \sim N(0, 10^2)$
- $\boldsymbol{Y} = X\boldsymbol{\beta} + \boldsymbol{\epsilon}$

This specification describes the entry for one row of our design matrix, $\boldsymbol{X}$. For every row we re-simulate all of the elements above. Applying our regression methods and plotting their $\hat{\boldsymbol{\beta}}$ estimates for $n = 300$ observations (rows) gives figure[2] 5.2.

Figure 5.2: The 50 estimates of the 50 beta, provided by each regression method for $n = 300$ observations. The true values of beta are displayed in pink on the right-most section of the graph.

To aid in the comparison of each regression method's ability to estimate $\boldsymbol{\beta}$, we introduce a new quantity, beta distance, which we define as:

$$\textit{Beta Distance} := \frac{1}{p}\sum_{j=1}^{p}|\hat{\beta}_j - \beta_j|.$$

The beta distance is the average distance of our estimated value of the coefficient, $\hat{\beta}_j$, and the true value of β_j. We only know the true value of β_j in simulations studies such as this. A model having a low beta distance implies that coefficients are estimated accurately, and so, we can make more accurate inferences about the true values of β_j based on the values of our estimated $\hat{\beta}_j$. The simulations carried out in this example provide the following table of beta distances 5.1. Table 5.1 effectively highlights the ability of these regression techniques in dealing with grouped data. The clear winner in terms of ability to predict the true values of $\boldsymbol{\beta}$ in this example is ridge regression, which has a significantly lower beta distance than all the other shrinkage regressions tested. We can also observe its superiority in this category of data by looking at 5.2. It is clear that the ridge regression estimates $\hat{\boldsymbol{\beta}}$ are closest to the true values of $\boldsymbol{\beta}$. The elastic net appears to be second best at handling this type of data. The Bayesian techniques all dealt with the data fairly similarly, outperforming

[2]We plot mean samples as our estimates of $\boldsymbol{\beta}$ for the Bayesian regressions. For the frequentist elastic net a value of $\alpha = 0.5$ was selected.

Observations	Bayes' Ridge	Bayes' Lasso	Bayes' Elastic Net	Ridge	Lasso	Elastic Net
$n = 300$	3.149	3.578	3.138	0.333	4.297	1.959
$n = 100$	3.949	4.642	4.381	0.545	7.486	3.010
$n = 50$	4.806	5.342	4.836	3.330	9.346	4.514

Table 5.1: The respective beta distances for each type of regression.

the frequentist lasso, but not being as powerful as the frequentist elastic net. However, it would appear from this example simulation that of these Bayesian regression methods, the Bayesian lasso performed the weakest. This is what we expected as the Bayesian lasso's MAP estimates are identical to the lasso's loss function minimisation estimates, and the lasso performs the worst of all tested regressions on this grouped data. Although, it should be noted that this example is not exhaustive, and many more tests should be run on different variations of grouped datasets to test the notion that the Bayesian lasso does not perform as well as other Bayesian regressions on grouped data as is done in [20](p.42).

As well as the Bayesian elastic net, there are many more variations of the Bayesian lasso, which correspond to frequentist variations of the lasso. While these variations effectively deal with problems the lasso's problems in the frequentist paradigm, we found it hard to motivate pursuing many more variations of the lasso translated into the Bayesian paradigm. While methods exist to employ the group lasso and adaptive lasso (known as hyperlasso in Bayesian regression) into the Bayesian counterparts, we found that the issues which these methods attempt to fix in the frequentist setting do not necessarily transfer perfectly over to Bayesian regression. We can see from example 5.2.2 that when dealing with grouped predictors the 3 different Bayesian methods we employed all lead to fairly similar results in terms of coefficient estimation, even though there were stark differences between their frequentist equivalents. This was also observed when using the Bayesian hyperlasso and Bayesian group lasso, and so we leave them out of this project. Generally, with grouped data, it is difficult to say whether any one is better than another, and the simulation data from [20](p.42) confirms this suspicion. Particularly, they find that with various examples of generated grouped data, the Bayesian grouped lasso performs extremely similarly to the standard Bayesian lasso! And so, out of the problems with the lasso listed in (5.2.1), we conclude that they do not necessarily apply in the Bayesian case, in spite of the relationship between the frequentist and Bayesian lasso.

5.3 Determining Optimal Penalisation Parameters

In this section we explore the different methods of estimating our penalising parameter/s. There are 3 main approaches to this problem,

- The full-Bayes approach. This involves treating λ as a random variable to be estimated alongside other parameters through posterior sampling, such as $\boldsymbol{\beta}$ and σ^2, and involves specifying a prior distribution for λ
- The empirical-Bayes approach. Using the marginal likelihood given by,

$$f(\boldsymbol{y}|\lambda) = \int_0^\infty \int_{\mathbb{R}^p} f(\boldsymbol{y}|\boldsymbol{\beta}, \sigma^2) p(\boldsymbol{\beta}|\sigma^2, \lambda) p(\sigma^2) d\boldsymbol{\beta} d\sigma^2.$$

  We estimate $\lambda_{EB} = \arg\max_\lambda f(\boldsymbol{y}|\lambda)$
- Cross-validation is another approach. This is the method used in frequentist penalisation techniques and involves splitting our data into training and testing subsets, and using the value for λ which provides the best accuracy at predicting the testing subsets, using their respective training subsets

5.3.1 Full-Bayes

The full-Bayes approach to penalisation is very easy to implement and consistent with the Bayesian methodology used thus far. It involves specifying a prior distribution for λ. Since the idea of a penalising parameter is abstract and hard to make any inference on based on a dataset, people typically use the half-Cauchy distribution for λ. This idea originates from [9]. This prior distribution is an example of a weakly informative prior. Although most of its probability density is concentrated close to zero, its large tails allow for potential extreme values of λ to be chosen. These extreme values will be taken when the likelihood of our data suggests that extreme penalisation is necessary.

Definition 5.3.1. ***The half-Cauchy distribution*** *is such that* $\lambda \sim$ *half-Cauchy*(a, b) *implies,*

$$p(\lambda) = \frac{2}{\pi b\left[1 + \left(\frac{\lambda - a}{b}\right)^2\right]}, \quad a \geq 0, \quad b > 0, \quad \lambda \in [0, \infty), \tag{5.3}$$

This distribution is counted as weakly informative as $p(\lambda) \propto \lambda^{-2}$. This proportionality results in much heavier tails than other potential distributions with $\lambda > 0$. For example, the half-normal distribution (where $p(\lambda) \propto \exp(-\lambda^2)$) has tails which decay exponentially, which leads to much lower probability density around extreme values. The half-Cauchy

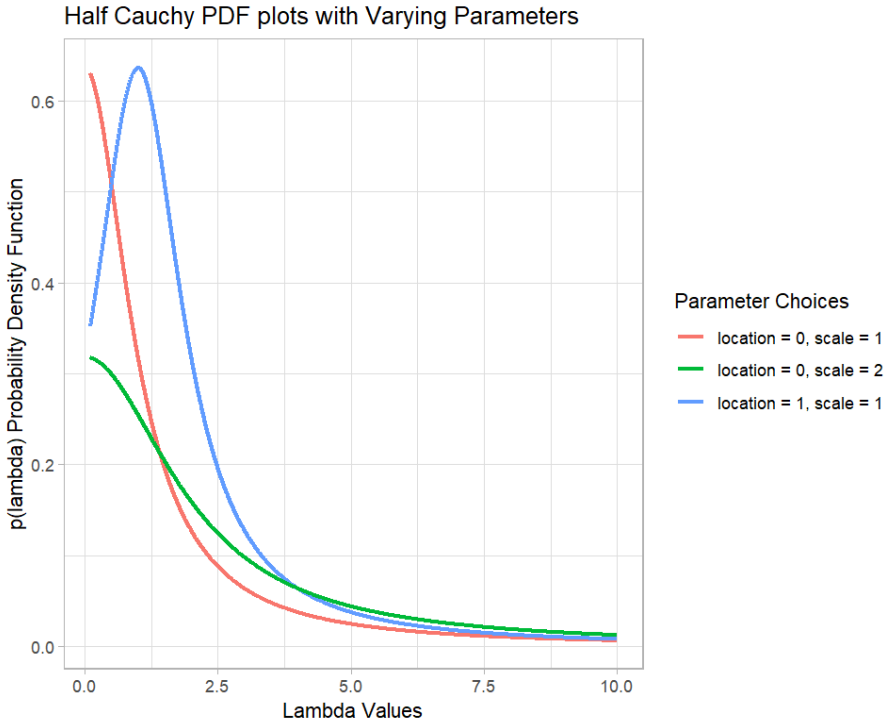

Figure 5.3: A plot of half-Cauchy prior densities with varying parameters.

prior for λ is typically chosen such that $\lambda \sim$ half-Cauchy$(0, 1)$, with a pdf, $p(\lambda)$ which peaks at $\lambda = 0$ and has heavy tails. This is a practical choice of prior distribution as we typically find that λ is quite small, however occasionally λ takes larger values, and the half-Cauchy allows for this. A half-Normal distribution would potentially assign too low probabilities for extreme values of λ. To aid our understanding of this distribution we

calculate the mean and median quantities of the $\lambda \sim$ half-Cauchy$(0, 1)$.

$$\begin{aligned}\mathbb{E}(\lambda) &= \int_0^\infty \lambda p(\lambda) d\lambda, \\ &= \frac{2}{\pi} \int_0^\infty \frac{\lambda}{1+\lambda^2}, \\ &= \frac{2}{\pi} \left[\frac{1}{2} \log\left(1+\lambda^2\right) \right]_{\lambda=0}^{\lambda=\infty}.\end{aligned}$$

Hence, $\mathbb{E}(\lambda) =$ undefined. Intuitively, this is a result of the tails of our distribution being very large. When we try and calculate an empirical estimate of $\hat{\lambda} = \frac{1}{n}\sum_{i=1}^n p_i(\lambda) \approx \mathbb{E}(\lambda)$, we find that $\hat{\lambda}$ never converges as a result of so many outliers being predicted. An intuitive explanation of this is that no matter how many samples we take from the half-Cauchy distribution, our empirical estimate of the mean never gets more accurate. We can also calculate the median, using the substitution $s = \tan u$ as follows:

$$\begin{aligned}\frac{2}{\pi} \int_0^{\lambda_{\text{med}}} \frac{1}{1+s^2} ds &= \frac{1}{2}, \\ \frac{2}{\pi} \int_0^{\arctan(\lambda_{\text{med}})} \cos^2(u) \sec^2(u) du &= \frac{1}{2}, \\ \frac{2}{\pi} \left[u\right]_{u=0}^{u=\arctan(\lambda_{\text{med}})} &= \frac{1}{2}, \\ \arctan(\lambda_{\text{med}}) &= \frac{\pi}{4}, \\ \lambda_{\text{med}} &= 1.\end{aligned}$$

This tells us that half of our prior density for $\mathbb{P}(\lambda \leq 1) = 0.5$. This shows that there is a significant probability density around small values of λ, which is helpful as we usually expect λ to be relatively small. The heavy tails are useful as they allow our posterior estimate to be greatly impacted based upon our likelihood, making the half-Cauchy a flexible prior distribution for λ.

5.3.2 Empirical-Bayes

The empirical-Bayes method of calculating λ involves the following equation,

$$\lambda_{EB} = \arg\max_\lambda f(\boldsymbol{y}|\lambda) = \arg\max_\lambda \int_0^\infty \int_{\mathbb{R}^p} f(\boldsymbol{y}|\boldsymbol{\beta}, \sigma^2) p(\boldsymbol{\beta}|\sigma^2, \lambda) p(\sigma^2) d\boldsymbol{\beta} d\sigma^2.$$

The calculation of the marginal likelihood, $f(\boldsymbol{y}|\lambda)$, involves the above integral which is often very impractical or impossible to calculate analytically. Because of this, in practice we will use our posterior samples to estimate λ_{EB}. The empirical Bayes approach to selecting λ is a compromise between Bayesian and frequentist reasoning. Of course we are applying Bayesian methodology to obtain samples and assign parameters probability density functions, however, in the empirical Bayes case we are treating λ as a constant to be estimated, as a frequentist would. In the estimation of λ_{EB} we must first estimate it in a preliminary sampling step, because of this we are required to assign λ a prior. Due to the nature of this approach, the obvious choice would be a flat prior where $p(\lambda) \propto 1$ for $\lambda \in (0, \infty)$, but this approach can often result in issues with sample convergence, [20]. This is a result of the flat prior not being a proper density. As such, [20] proposes an ad-hoc solution to this issue. If we use a prior for λ which is approximately constant along $\lambda \in (0, \infty)$ we could then keep the spirit of using a constant and uninformative prior while having a valid posterior distribution. Hence, we use $\lambda \sim$ half-Cauchy$(0, 100000)$. As the shape parameter is so large, it is effectively non-informative as it is so flat. Using this prior density, we simulate from our posterior distribution as we usually would in the

full-Bayes approach, however, after retrieving these samples, we calculate the posterior mode of λ. This is an estimate of $\hat{\lambda}_{MAP}$, which itself is an estimate of λ_{EB}. Once we obtain this estimate of λ_{EB}, we rerun the model, but now treating λ as a constant term, not to be estimated. Hence, we no longer have a prior for λ and we see it as a frequentist would. This technique offers fair predictive performance, and can be computationally more efficient than full-Bayes if we are fitting a complicated model multiple times. In the full-Bayes approach every time we fit a model our sampler must calculate new values of λ, but using empirical-Bayes, it only has to run these calculations once.

Example 5.3.2. *We once again consider the data simulated in A.3. We wish to estimate the empirical-Bayes value of λ, using the NIG model, (4.5). As such we use RStan to simulate from the posterior with the given data, and priors from the NIG model, with the exception of using the (extremely) weakly informative half-Cauchy$(0, 100000)$ prior for λ. Upon receiving the samples of λ, we can use functions to plot an approximate density of our marginal posterior for λ, $p(\lambda|\boldsymbol{y})$. The mode represents our estimate of λ_{EB} and is*

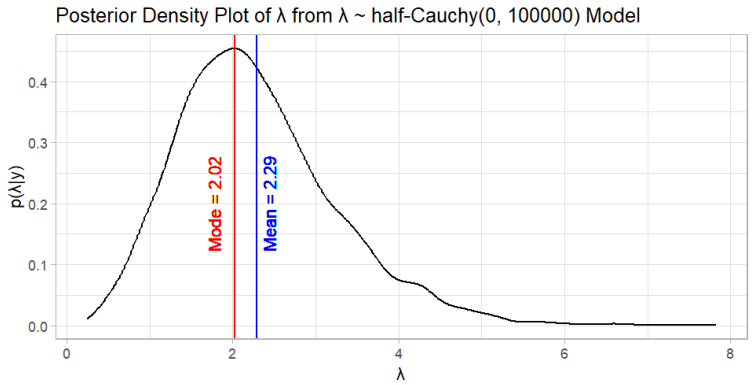

Figure 5.4: A plot of the posterior density of λ using the weakly informative half-Cauchy$(0, 100000)$ prior.

shown in 5.4 by the vertical red line. This is the λ estimate we will use in our model. We also mark the mean as it may be of interest to compare the model's performance with the modal estimate of λ against the performance with the mean estimate of λ. The mean captures some of the right-skew of our distribution, while the mode just selects the point of highest density. Using the estimate $\lambda_{EB} = 2.02$ in a model which treats λ as a constant, with no prior or posterior, will result in a model which has been determined using the empirical-Bayes method. This model is the one which we will use for predictions and estimation.

5.3.3 Cross-Validation

Cross-validation (CV) is the method of calculating λ which is used in frequentist methods, typically, using a process called K-fold CV. This involves partitioning the subset of data used to train the model into K segments. We denote the k^{th} segment of data $\boldsymbol{y}_k$, for $k = 1, \ldots, K$. After this partition, we fit k different models using the data $\boldsymbol{y}_{-k}$ to train the models, and testing the predictive performance by using each model to predict the values of $\boldsymbol{y}_k$. The predictive performance can measured through multiple different methods. One such method is that of the predictive mean-squared-error, PMSE (3.2.2). We will then analyse which training model gives us the lowest value of PMSE and select that model as our best[3]. This works well in the frequentist setting as fitting models is often

[3]In this case of CV we are searching for the model we believe will have the greatest predictive accuracy. The model which provides the lowest PMSE, when predicting the data $\boldsymbol{y}_k$ is likely to have the 'best' predictive accuracy. However, as a point of nuance, there are different measures of predictive accuracy, which may provide evidence for a different model having superior predictive accuracy to the model with the lowest PMSE.

very computationally efficient and predictions are simple to make. In the context of the Bayesian linear model we run into some issues, including that:

- It is computationally costly to create so many models, as Bayesian sampling schemes take significantly longer to run than their frequentist counterparts
- It is not obvious what the best method of prediction is for a Bayesian model. We believe that each parameter has it's own probability density and we can simulate draws from these densities, however, there are many possible choices of $\hat{\boldsymbol{\beta}}$ which we could hypothetically use to generate predictions. For instance, we could take the mean of all our samples for each parameter and use those values for prediction, or take the mode ($\boldsymbol{\beta}_{MAP}$ estimates) or even just use a random draw from our joint posterior distribution. We also have the option of treating our new predictions as random variables with their own densities, this method is perhaps the most conducive to the Bayesian paradigm, but still leaves us with the problem of being unsure which value to report as our prediction if we are using a predictive metric such as PMSE

Due to these issues, it is ineffective to use K-fold validation for Bayesian modelling, because of this, we must look to more nuanced methods if we still wish to use CV. An example of a CV method applicable to Bayesian models is the leave-one-out (LOO) CV.

LOO CV

This subsection about LOO CV is a summary of the paper written by the people responsible for the 'loo' package in R, the original paper is found here, [1]. LOO is the case of K-fold CV where $K = N$. This means that we only 'leave out' one data observation in the training of our model. We say the data which does not include observation k is the set of $\boldsymbol{y}_{-k}$ observations. And, instead of using a frequentist metric such as PMSE, which has problems in a Bayesian context, we calculate the expected log pointwise predictive density (ELPD).

$$ELPD := \sum_{i=1}^{N} \int p_t(y_i^*) \log f(y_i^*|\boldsymbol{y}, \boldsymbol{\beta}, \sigma^2, \lambda) dy_i^*,$$

in which $p_t(y_i^*)$ is the theoretical true data generation process which y_i^* is an observation from and $p(y_i^*|\boldsymbol{y}, \boldsymbol{\beta}, \sigma^2, \lambda)$ is our predictive density.

$$p(y_i^*|\boldsymbol{y}) = \iiint f(y_i^*|\boldsymbol{\beta}, \sigma^2, \lambda) p(\boldsymbol{\beta}, \sigma^2, \lambda|\boldsymbol{y}) d\boldsymbol{\beta} d\sigma^2 d\lambda,$$

Of course, in practice (unless we are using a simulation study) we never know what the true data generating process $p_t(y_i^*)$ is, and so we must estimate the ELPD. We can do this with $ELPD_{loo}$,

$$\hat{ELPD}_{loo} = \sum_{i=1}^{n} \log p(y_i^*|\boldsymbol{y}_{-i}),$$

where $p(y_i^*|\boldsymbol{y}_{-i}) = \iiint p(y_i^*|\boldsymbol{\beta}, \sigma^2, \lambda) p(\boldsymbol{\beta}, \sigma^2, \lambda|\boldsymbol{y}_{-i}) d\boldsymbol{\beta} d\sigma^2 d\lambda$. We can estimate the value of $p(y_i^*|\boldsymbol{y}_{-i})$ using importance sampling, which uses all of our samples from our joint posterior distribution, $(\boldsymbol{\beta}^{(s)}, \sigma^{2(s)}, \lambda^{(s)})$. If we are using stan to simulate from our posterior distribution we by default obtain 4000 draws, so $s \in \{1, \dots, 4000\}$. Firstly, we consider importance ratios, r_i^s,

$$r_i^s = \frac{1}{f(y_i|\boldsymbol{\beta}^{(s)}, \sigma^{2(s)}, \lambda^{(s)})} \propto \frac{p(\boldsymbol{\beta}^{(s)}, \sigma^{2(s)}, \lambda^{(s)}|\boldsymbol{y}_{-i})}{p(\boldsymbol{\beta}^{(s)}, \sigma^{2(s)}, \lambda^{(s)})|\boldsymbol{y})},$$

which we use to approximate:

$$p(y_i^*|\boldsymbol{y}_{-i}) \approx \frac{\sum_{s=1}^{4000} r_i^s f(y_i^*|\boldsymbol{\beta}^{(s)}, \sigma^{2(s)}, \lambda^{(s)})}{\sum_{s=1}^{4000} r_i^s},$$

and applying this formula to LOO gives,

$$p(y_i|\boldsymbol{y}_{-i}) \approx \frac{1}{\frac{1}{4000}\sum_{s=1}^{4000}\frac{1}{f(\boldsymbol{y}_i|\boldsymbol{\beta}^{(s)},\sigma^{2(s)},\lambda^{(s)})}}.$$

This means that if we know the log likelihood for each of our $s \in \{1, \dots, 4000\}$ posterior samples, we can estimate our predictive accuracy. Using this we can compare the predictive accuracy of models and choose models which have the highest $\hat{ELPD}_{loo}$. The key reason why LOO CV is so convenient for comparing Bayesian models is because of the approximation that $p(y_i|\boldsymbol{y}_{-i}) \approx p(y_i|\boldsymbol{y})$, which is used by the LOO-package in RStudio. This approximation practically means that we do not 'leave one [observation] out' when calculating this predictive probability. The computation time saved using this approximation is huge, as we no longer need to fit n different models to compare predictive accuracy, as we would otherwise need to calculate $p(y_i|\boldsymbol{y}_{-i})$, rather, we can fit one model and use the values of $p(y_i|\boldsymbol{y})$ for each $i \in \{1, \dots, n\}$ to estimate predictive accuracy. The larger the training data set, the stronger this approximation is.

We can use these ELPD values to estimate model weights for different values of λ_p, with $p \in \{1, \dots, P\}$ as follows:

$$w_x \approx \frac{\exp\left[\sum_{i=1}^{i=n}(p(y_i|\boldsymbol{y}_{-i},\lambda_x))\right]}{\sum_{p=1}^{p=P}\exp\left[\sum_{i=1}^{i=n}(p(y_i|\boldsymbol{y}_{-i},\lambda_p))\right]}. \tag{5.4}$$

The value of w_x is useful as we can interpret it as the probability that the model which uses $\lambda = \lambda_x$ is the 'best' predictive model, and so $w_x \approx 1$ implies the chance that using λ_x gives us our optimal predictive model[4] is very high. If we are comparing P values of λ, we will need to compare the predictive accuracy of the model with each λ_p, and the one which has the greatest weight, w_p, is the model which is most likely to be our best in terms of prediction.

Furthermore, the model weights w_x which we can calculate from $\hat{ELPD}$ values can be used to compare the weights of completely different Bayesian models. For instance, we could calculate the $\hat{ELPD}$ of a model which uses a ridge prior, $\hat{ELPD}_1 = p(y_i^*|\boldsymbol{y}_{-i}, \text{Ridge})$, and the $\hat{ELPD}$ of a lasso prior model, $\hat{ELPD}_2 = p(y_i^*|\boldsymbol{y}_{-i}, \text{Lasso})$. Using these $\hat{ELPD}$ values we can approximately estimate which model is preferable in terms of prediction, using the same methodology as in (5.4), but instead of comparing models with varying λ values, we compare completely independent models acting on the data,

$$w_1 = \frac{\exp\left[\sum_{i=1}^{i=n}(p(y_i|\boldsymbol{y}_{-i}, \text{Ridge}))\right]}{\exp\left[\sum_{i=1}^{i=n}(p(y_i|\boldsymbol{y}_{-i}, \text{Ridge}))\right] + \exp\left[\sum_{i=1}^{i=n}(p(y_i|\boldsymbol{y}_{-i}, \text{Lasso}))\right]}.$$

Example 5.3.3. *We wish to obtain a value for λ using LOO-CV, using the data from 4.3.1. Employing general purpose optimisation functions we can find the λ values which minimise LOO-CV error and, for comparison, a measure of PMSE. The RStudio code can be found here, 5.3.3 A. Note that the Stan model must report the log likelihood with each iteration, in order to calculate the LOO estimate of predictive accuracy. We find that when trying to optimise both LOO and PMSE, when our number of training observations is high, the optimising functions do not converge. This is to be expected as the penalising parameter λ is used by the model exclusively in the prior specification of $\boldsymbol{\beta}$. As such, when n is large, the value of λ does not significantly impact predictive performance, as our data is much more influential on our posterior density than our prior, making this form of CV an impossible tool to meaningfully use. However, when n is small, the use of cross-validation in tuning the penalising parameter, λ, is an interesting alternative to full-Bayes and empirical-Bayes. When we change the number of observations to $n = 10$*

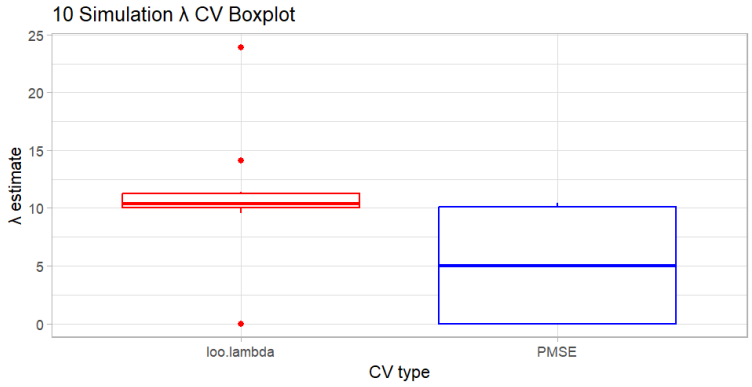

Figure 5.5: A box plot of λ values which are selected while trying to optimise LOO values (in red) and PMSE values (in blue).

LOO-CV	PMSE-CV
0.0001	10.00
10.00	0.0001
9.53	0.0001
11.41	0.0001
10.62	0.0001
10.12	10.11
14.09	10.11
10.77	0.0001
10.11	10.04
23.90	10.47

Table 5.2: The table of the λ values obtained when optimising LOO values and PMSE values.

from $n = 100$, and box plot our estimated λ_{CV} values for both methods of cross-validation we obtain figure 5.5 and table 5.2. Notice from box plot 5.5 that PMSE, has a very large spread, as the λ value changes between $\lambda \approx 10$ and $\lambda = 0.0001$. LOO-CV shows potential convergence around $\lambda_{CV} \approx 10.50$, although it does also have a few outlying observations. Table 5.2 shows the data used in plotting the above box plot. Notice that for PMSE-CV half of the λ_{CV} estimates are at $\lambda_{CV} = 0.0001$. This is because we took 0.0001 as the lower limit as we cannot use 0[5]. The estimate being $\lambda_{CV} = 0.0001$ suggests that the optimising function is trying to get λ extremely low. This could mean that to optimise PMSE, we potentially want to have a flat prior for our coefficients, because as $\lambda \to 0$ the prior variance of our coefficients is increasing, suggesting minimal shrinkage is necessary. However, we should be skeptical of this notion as although around half of our estimates of λ when optimising PMSE are near zero, the other half are around 10, which is a large difference. Moreover, we would not expect $\lambda \approx 0$ to be optimal due to the true values of the $\boldsymbol{\beta}$ coefficients being relatively close to zero, $\boldsymbol{\beta} = (0.5, 2, 0)^T$. These factors make the value of $\lambda = 10$ seem more appropriate. The wild performance of the PMSE-CV is partly why it is not optimal in the Bayesian setting for model optimisation. Although, when it does converge to a sensible value, it is fairly consistent with the value of the loo estimate, and from looking at this data, it seems that $\lambda_{CV} \approx 10.50$ would be a sensible choice for the penalising parameter.

[4]'Best' compared to the other models which are accounted for in the calculation, not necessarily the best model which is theoretically possible.

[5]As the prior coefficient variance is undefined at $\lambda = 0$

5.4 Variable Selection

Variable selection involves setting coefficients which relate to predictor variables which are deemed to be weakly related (insignificant) to the response variable to zero. Variable selection is very desirable for aiding the interpretability and parsimony of our model, and can also improve predictive accuracy if we adjust our model after determining some coefficients to be insignificant. The ordinary least squares (OLS) model, ridge regression and all Bayesian linear regression methods do not perform variable selection.

Example 5.4.1. *We again consider the data from 4.3.1. Recall the true coefficients are* $\boldsymbol{\beta} = (0.5, 2, 0)^T$. *We fit the frequentist models using* $n = 40$ *observations of training data and without intercepts. We notice from the table above that lasso regression can*

Coefficient	OLS	Ridge	Lasso	TRUE
β_1	0.26	0.06	0.00	0.5
β_2	2.11	0.45	0.99	2.0
β_3	0.16	-0.20	0 .00	0.0

perform variable selection, although it of course is not perfect. Assigning the β_3 *coefficient a non-zero value can be seen as a form of overfitting, as the* 3^{rd} *predictor variable will unnecessarily influence predicted values.*

Having to select relevant variables is a very common issue, especially when dealing with a model which has a high number of predictor variables (p large). An example of a case where variable selection must be performed is when considering how the presence of genes can influence blood-pressure. We would have potentially several thousand genes, and most of them will most likely be unrelated to blood-pressure, but a naive model such as OLS will account for these insignificant genes, decreasing our predictive accuracy and inferential ability. The inferential ability is very important in many cases including this one, as if we know the presence of a gene is correlated to extreme blood-pressure, it can be tested for and then further action can be taken. We call a model which performs variable selection "sparse", and Bayesian linear models are not automatically sparse. In the Bayesian paradigm, even if there is a supposedly true $\beta_j = 0$, we will be able to observe that $\beta_j \neq 0$ infinitely many times. Further, as a result of the continuous nature of it's probability density function we have that $\mathbb{P}(\beta_j = 0) = 0$. We can attempt to counteract this shortfall of Bayesian linear models by checking whether or not zero lies within a sensibly sized credible interval of our posterior coefficient density. If it does, then we have the (non-rigorous but practical) option of manually removing the predictor from our model to increase parsimony. However, even with this method of inducing manual model sparsity, we find in practice that for this proposed method of Bayesian variable selection we need a large proportion of the posterior coefficient density of insignificant coefficients to be concentrated around zero. If a prior is capable of concentrating posterior density around zero then we say that the model induces sparsity, in spite of it not performing automatic variable selection in the traditional sense. The ridge prior struggles with concentrating enough posterior density of insignificant predictors around zero [5] for credible intervals to reliably identify insignificant coefficients.

5.4.1 Lack of Sparsity Inducing Power of the Ridge Prior

To motivate our pursuit of a sparsity inducing model, we first discuss why standard Bayesian ridge regression prior does not effectively concentrate enough posterior coefficient density around zero for insignificant coefficients. The posterior distribution of a coefficient, β_j, can be seen as a compromise between the information contained within its prior distribution and its likelihood. Using the ridge prior we have that all $\beta_j \overset{iid}{\sim} N(0, \sigma^2\lambda^{-1})$, and we struggle to greatly reduce the size of our insignificant coefficients while not overly

penalising significant coefficients, [6](2.1). This is a result of all prior variances for β_j being identical. To fix this we must find a way to shrink potentially insignificant coefficients more than significant coefficients. Furthermore, the normal distribution itself does not have a sharp-peak at zero or any particularly large density around zero, leading to coefficients being penalised similarly regardless of their significance. Moreover, due to the conjugacy of the ridge prior with the normal likelihood which we have in the context of linear regression, our posterior distribution for β_j will always also be a normal distribution. This is a nice property for calculations and inference, however, in the case that a coefficient is insignificant the probability density is often too spread out to allow for variable selection. Because of this, to encourage sparsity[6] and close to zero posterior draws, we wish to concentrate posterior probability density around zero for potentially insignificant variables. One way of doing this is by using spike-and-slab priors.

5.4.2 Spike-And-Slab

Spike-and-slab models use spike-and-slab priors. These priors are hierarchical. The prior specification we analyse first seeks to increase the number of observations of $\beta_j = 0$ by setting $\mathbb{P}(\beta_j = 0) \neq 0$ as follows [11];

$$\begin{aligned} \boldsymbol{y}_i|\boldsymbol{\beta}, \sigma^2 &\sim N(\boldsymbol{x_i}^T\boldsymbol{\beta}, \sigma^2) \\ \beta_j|z_j, \sigma^2 &\sim N(0, \frac{\sigma^2}{\lambda} \cdot z_j) \\ z_j &\sim \text{Bernoulli}(\theta). \end{aligned}$$

This prior specification means that β_j assumes the usual value it would have without being conditioned on z_j with probability θ, and is zero with probability $1-\theta$. Or, $\mathbb{P}(\beta_j \neq 0) = \theta$ and $\mathbb{P}(\beta_j = 0) = 1 - \theta$. This is because when we are using a penalising prior, the mean of our distribution will be zero, and if we have a variance of zero, this means that all of our probability density (both prior and posterior) will be concentrated solely at zero. The case in which $z_j = 0$ results in a "spike" in probability at $\beta_j = 0$. With this prior specification, the spike can be represented as $p(\beta_j|z_j = 0) = \delta(\beta_j)$, where $\delta(\cdot)$ is the Dirac delta function. The case with $z_j = 1$ is when we have our "slab" of probability density for β_j, where our β_j behaves as usual. We plot the two cases of $z_j = 0$ and $z_j = 1$ in figure 5.6. While this model is simple and easy to understand, it is computationally unpleasant

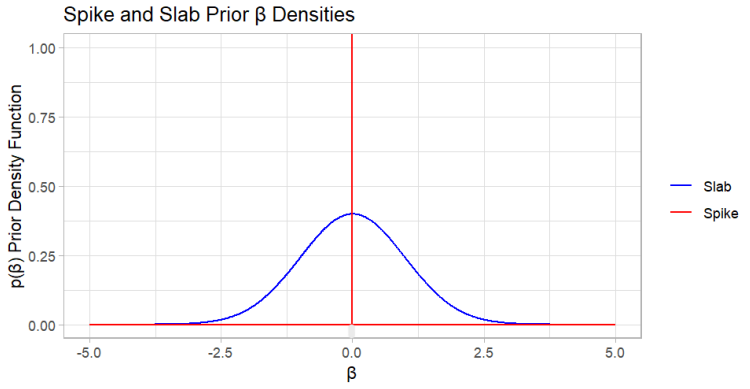

Figure 5.6: A graph to highlight the different prior densities. $\beta \sim N(0, z \cdot \sigma^2/\lambda)$, with $z = 1$ resulting in the slab and $z = 0$ resulting in the spike.

to deal with. The spike density being infinitesimally thin and infinitely tall introduces issues with sampling, and Stan does not allow $z_j \sim$ bernoulli(θ) as estimated parameters

[6]Sparsity in the sense that a credible interval for an insignificant coefficient's posterior density will likely include zero.

cannot be integers under its coding framework. Furthermore, it is not very cohesive with Bayesian methods to have $p(\beta_j|z_j = 0) = \delta(\beta_j)$, as it is odd to conclude that we are so sure about the value of β_j being exactly equal to zero before accounting for observations, even if it only occurs with probability θ. As such we formulate a compromise with the following prior specification, for the continuous spike-and-slab model [6](2.2.1);

$$\begin{aligned} \boldsymbol{y}_i|\boldsymbol{\beta}, \sigma^2 &\sim N(\boldsymbol{x_i}^T\boldsymbol{\beta}, \sigma^2) \\ \beta_j|\theta_j, \sigma^2, \lambda &\sim \left[\theta_j N(0, \frac{\sigma^2}{\lambda}) + (1-\theta_j)N(0, \epsilon^2)\right], \epsilon^2 \ll \frac{\sigma^2}{\lambda} \\ \theta_j &\sim Beta(a, b). \end{aligned} \tag{5.5}$$

This prior specification[7] avoids the computational difficulties and Bayesian inferential issues which plagued the previously considered spike-and-slab model, however due to it's hierarchical nature it can encounter effective sample size difficulties. As such it is useful to marginalise θ_j out of our model, as follows,

$$\begin{aligned} p(\beta_j|\lambda, \sigma^2, a, b) &= \int_0^1 p(\beta_j, \theta_j|\lambda, \sigma^2, a, b)d\theta_j, \\ &= \int_0^1 \underbrace{p(\beta_j|\theta_j, \lambda, \sigma^2, a, b)}_{\text{Mixture of normal densities}} \underbrace{p(\theta_j|a, b)}_{\text{Beta}(a,b)} d\theta_j, \\ &= \left[\int_0^1 \theta_j \text{Beta}(\theta_j|a, b)d\theta_j\right] N(\beta_j|0, \sigma^2\lambda^{-1}) + \\ &\quad \left[\int_0^1 (1-\theta_j)\text{Beta}(\theta_j|a, b)d\theta_j\right] N(\beta_j|0, \tau^2), \end{aligned}$$

$$\int_0^1 \theta_j \text{Beta}(\theta_j|a, b)d\theta_j = \mathbb{E}(\theta_j) = \frac{a}{a+b},$$

$$\implies p(\beta_j|\lambda, \sigma^2, a, b) = \frac{a}{a+b}N(\beta_j|0, \frac{\sigma^2}{\lambda}) + \frac{b}{a+b}N(\beta_j|0, \tau^2), \text{ [6] (2.2.1).}$$

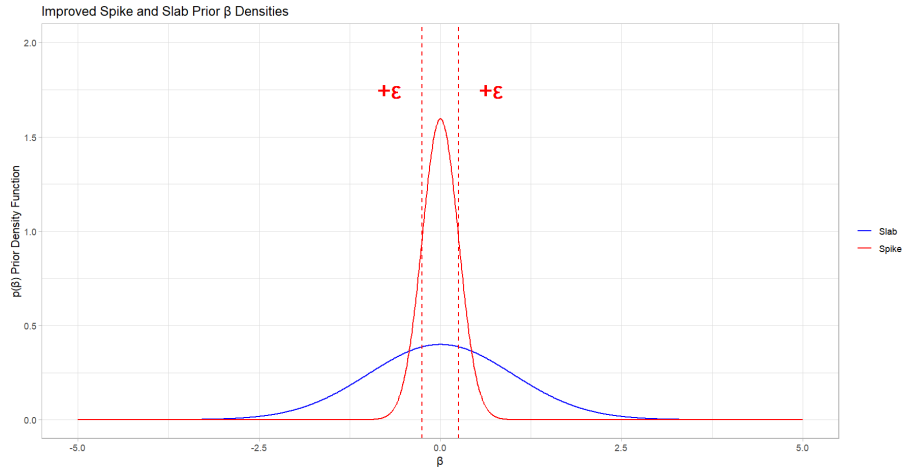

Figure 5.7: A plot continuous spike-and-slab's prior density (which is a mixture of two normal distributions).

When applying this prior specification we are effectively assuming that the probability density function of our β_j coefficients lie somewhere in between the slab and spike normal

[7] θ_j does not necessarily have to be Beta distributed, any distribution can work as long as the support is [0,1]

distributions, graphed in figure 5.7. This is useful as when we are dealing with a sparse dataset we know that some of our coefficients will be close to zero, and this specification allows for that while at the same time allowing for much larger coefficients, which are more closely aligned with the slab prior distribution. Furthermore, the large coefficients will still be penalised under the slab prior, which we have chosen to have the same variance and distribution as a ridge prior.

Example 5.4.2. *We wish to compare the prior specification of the continuous spike-and-slab model (5.5), found in A.7, against a base model in terms of sparsity. We will take* $\tau^2 = 0.01$. *The base model we will use is the NIG model A.4. We will also briefly compare the variable selection with that of the frequentist lasso. Our data is simulated such that we use* $n = 80$ *observations to train where* $\boldsymbol{X}_{ij} \sim N(5, 3^2)$ *and* $\boldsymbol{Y} = \boldsymbol{X}\boldsymbol{\beta} + \boldsymbol{\epsilon}$. $\boldsymbol{\beta}$ *is made up of a sequence between* $5 \to 10$ *of length* 25, *and a sequence between* $-5 \to -10$ *of length* 25, *and* 50 *zeroes.* $\epsilon \overset{iid}{\sim} N(0, 3^2)$. *Note that this is a challenging dataset, as* $p > n$ *and so models can find it difficult to select relevant coefficients. In fact, if we try and use the normal equation (2.1) and calculate the maximum likelihood estimate* $\hat{\boldsymbol{\beta}}$ *we are stopped as we have collinear columns and so we cannot calculate* $(\boldsymbol{X}^T\boldsymbol{X})^{-1}$, *hence if we wish to compare a Bayesian model with a frequentist model here, the frequentist model must be penalised. We found that for this simulation study, the* 95% *equal-tailed credible intervals were too wide to be useful, this is a result of the low sample size relative to the number of predictors. This low sample size means that the likelihood of the data is less influential in determining the form of the posterior than if we were to have more samples. The impact of this diffuse likelihood is that we have lots of posterior parameter uncertainty, and so we used the* 90% *intervals instead. Table 5.3 shows the power of the continuous spike-and-slab*

Regression	$\hat{\beta}_j \vert (\beta_j = 0) = 0$	$\hat{\beta}_j \vert (\beta_j \neq 0) = 0$
Lasso	96%	44%
NIG	62%	4%
Continuous Spike-Slab	100%	2%

Table 5.3: A table comparing the sparsity inducing power of the continuous spike-and-slab model against a NIG model and a frequentist lasso model.

prior in linear regression problems[8] *in which we have many insignificant coefficients. It also shows that frequentist lasso regression can recklessly set significant values to zero. In this example, we shall consider the following* β_j *with true values:* $\beta_1 = 5$, $\beta_{50} = -10$ *and* $\beta_{97} = 0$. *We plot the respective posterior densities of these coefficients for both Bayesian models tested in figure 5.8. We can see the spike of our prior in the spike-and-slab model greatly influencing our posterior distributions when considering* β_1 *and* β_{97}. *Our posterior for* β_1 *is an interesting case in which we can see clearly the bi-modality of our posterior distribution. When our data suggests a parameter is insignificant, the spike distribution will have a great influence on our posterior due to its high concentration of density about zero. This spike combines with the density suggested by our likelihood from our observed data. The likelihood suggests a modal value of around five which is deemed to be near enough to zero for our spike density to be preferred over our slab density. This results in the prior spike giving us a mode at* $\beta_1 = 0$, *and so we have bi-modality. This bi-modality leads to excessive penalisation and results in* $\mathbb{E}(\beta_1 \vert \lambda, a, b, \sigma^2, \boldsymbol{y}) = 2.6 < 5.0$. *This bi-modality disappears for our posterior distribution of* β_{50} *as the likelihood function provides strong enough evidence for* $\beta_{50} \neq 0$ *that the spike density becomes insignificant. We can see the spike-and-slab prior working as intended on* β_{97}, *which is an insignificant coefficient, and our model correctly notices this, by showing a large spike in posterior density at* $\beta_{97} = 0$. *We notice that the posterior densities using the standard ridge prior*

[8] Again, note that the lasso is the only regression to automatically perform variable selection, and for true sparsity in the Bayesian models we would need to remove insignificant predictors manually.

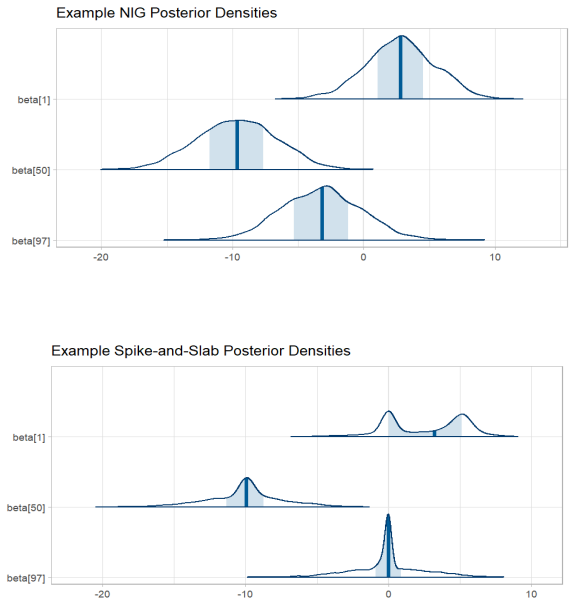

Figure 5.8: Graphs showing the posterior density of three β_j coefficients. The top graph shows posterior densities of the ridge prior, while the bottom shows the posterior densities resulting from the spike-and-slab prior.

*are much vaguer and more spread out in comparison. This is a result of the model being trained on so few pieces of data, and so the likelihood does not give us much certainty in the value of our coefficients. Hence, we are provided with reasonable, yet vague posterior densities. This vagueness results in many insignificant coefficients (*38% *of them) being incorrectly identified as significant using our* 50% *equal tailed credible intervals, with one of these misidentified coefficients being* β_{97}.

In spite of the excellent variable selection shown by the spike-and-slab model, it is not perfect. Our choice of ϵ has a large impact on the posterior distribution in spite of it being fairly arbitrary. And, the bi-modality of small yet significant coefficients does not make interpretative sense. If we are to believe that each β_j has it's own true probability distribution, we would most likely not expect it to be bi-modal in most cases, especially with simulated data which perfectly abides by the assumptions of the linear model. As such, we wish to consider other prior specifications which serve to induce sparsity. In the following subsection we consider prior specifications which induce sparsity without specifying our β_j prior as a discrete mixture of two normal distributions as we have done so far.

5.4.3 Continuous Sparse Priors

Generally, continuous sparse priors seek to have more probability density close to zero which allows for variable selection, and at the same time have heavy tails which results in the larger valued β_j not being overly penalised. This contrasts with the spike-and-slab prior which resulted in bi-modal priors causing extreme shrinkage to smaller values of β_j, even when they were significant. An example of one of the continuous sparse priors is the horseshoe prior which is specified as follows;

$$\begin{aligned} \boldsymbol{y_i}|\boldsymbol{\beta}, \sigma^2 &\sim N(\boldsymbol{x_i}^T\boldsymbol{\beta}, \sigma^2) \\ \beta_j|\lambda_j &\sim N(0, \lambda_j) \\ \lambda_j|\tau &\sim \text{half-Cauchy}(0, \tau) \\ \tau|\sigma &\sim \text{half-Cauchy}(0, \sigma). \end{aligned} \tag{5.6}$$

[19](p.4)[9]. The intuition behind this prior specification is that we obtain a small value of τ, which results in a strongly penalising model for all coefficients, but the parameter λ_j is considered for each coefficient and has the ability to potentially counteract this penalisation, as it determines how much penalisation our posterior distributions of β_j will experience. The half-Cauchy prior for λ_j places half of its prior probability density below τ, but the heavy tails of the half-Cauchy result in fair probability that λ_j is large. This is desirable as it theoretically has the potential to penalise insignificant coefficients much greater than it penalises significant coefficients. Most $\lambda_j \approx 1$, which will result in heavy penalisation in sparse populations, however, large valued λ_j allow for lower penalisation in certain scenarios. We can analyse the horseshoe prior density using figure 5.9. The red dashed line represents a vertical asymptote of the horseshoe prior at zero, resulting in the top of the density being cutoff. This extreme density near $\beta_j = 0$ (theoretically) results in harsh penalisation of insignificant coefficients. Also, note that the jaggedness of the density is a result of there being no tractable analytic expression for the marginal density of β_j with the horseshoe prior. Because of this, the plot was done using MCMC samples to estimate the shape of the density. We also would expect medium-small coefficients to be more penalised (compared to the ridge prior) as the horseshoe density could be described as a "pinched" version of the normal density. This means that medium-small values are less likely according to the horseshoe prior. Figure 5.10 shows that the horseshoe

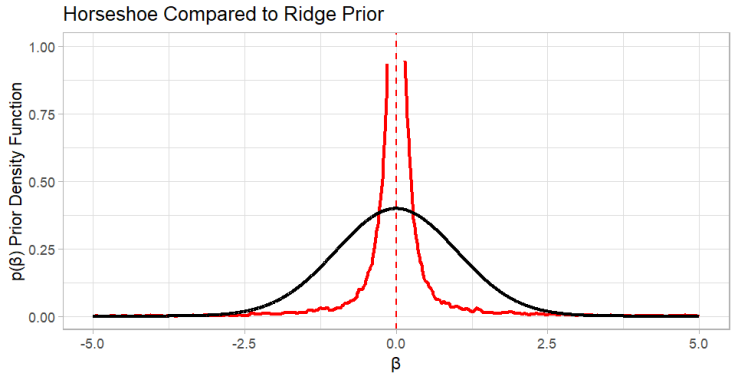

Figure 5.9: The horseshoe prior density (red) compared to the ridge prior density (black).

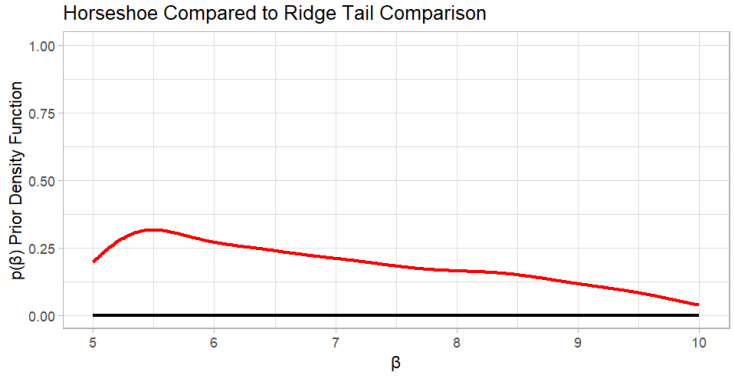

Figure 5.10: The horseshoe density (estimated by sampler) compared to the ridge prior density in (part of) the tail of the distributions, $\beta_j \in [5, 10]$.

prior density's tail dominates the ridge prior's tail. This makes a big difference in posterior results because the ridge prior decays much more severely for large values of β_j. Intuitively, the horseshoe prior will have less of an impact on our posterior distribution of β_j when

[9]The horseshoe prior explicitly places a prior distribution onto λ_j, our penalising parameter, and so we are effectively limited to the full-Bayes approach when using the horseshoe prior, [20](p.37).

β_j is large, and much more of an impact when the coefficient is near zero, theoretically inducing sparsity without over penalisation.

Just as we had to pick ϵ for our prior in the continuous spike-and-slab prior, (5.5), we must pick τ in the horseshoe prior. We call λ_j our local shrinkage terms, as they are unique for each β_j and we call τ our global shrinkage term as it is accounted for in every β_j. We define the shrinkage coefficient as,

$$\kappa_j := \frac{1}{1+\lambda_j^2},$$

and κ_j represents a way to visualise how our prior penalises based on the value of λ_j. This is because we can estimate the posterior mean under the horseshoe model as (assuming $\sigma^2 = \tau = 1$),

$$\mathbb{E}[\beta_j|\boldsymbol{y}] = \int_0^1 (1-\kappa_j) y_i p(\kappa_j|\boldsymbol{y}) d\kappa_j = \{1 - \mathbb{E}[\kappa_i|\boldsymbol{y}]\} y_i,$$

where we interpret $\mathbb{E}[\kappa_i|\boldsymbol{y}]$ as the amount of shrinkage of β_j towards zero, [7](p.1). Using this we observe that small values of $\kappa_j \approx 0$ result in negligible shrinkage, but larger values of κ_j where $\kappa_j \approx 1$ result in extreme shrinkage towards zero. We now calculate the prior probability density function of κ_j, $p(\kappa_j)$, up to a positive constant by doing a transformation of variables using $p(\lambda_j)$ (where $\lambda_j \sim$ half-Cauchy$(0, \tau = 1)$, specified in 5.6),

$$\begin{aligned}
p(\kappa_j) &= p(\lambda_j)\left|\frac{d\lambda_j}{d\kappa_j}\right|, \\
p(\lambda_j) &\propto (1+\lambda_j^2)^{-1} = \kappa_j, \\
\lambda_j &= (1-\kappa_j)^{\frac{1}{2}} \kappa_j^{-\frac{1}{2}}, \\
\left|\frac{d\lambda_j}{d\kappa_j}\right| &\propto (1-\kappa_j)^{-\frac{1}{2}} \kappa_j^{-\frac{1}{2}} + (1-\kappa_j)^{\frac{1}{2}} \kappa_j^{-\frac{3}{2}}, \\
&= (1-\kappa_j)^{-\frac{1}{2}} \kappa_j^{-\frac{3}{2}}, \\
p(\kappa_j) &= p(\lambda_j)(1-\kappa_j)^{-\frac{1}{2}} \kappa_j^{-\frac{3}{2}}, \\
&\propto (1-\kappa_j)^{-\frac{1}{2}} \kappa_j^{-\frac{1}{2}}.
\end{aligned}$$

This expression for the probability density function $p(\kappa_j)$ is the expression which gives the horseshoe its name. This expression is identical to that of the kernel of a Beta$(0.5, 0.5)$ distribution, graphed in figure 5.11. This figure shows us the characteristic horseshoe shape, implying a high amount of penalisation when κ_j is near zero and near one. This means that our posterior mean estimate of β_j will be near zero when κ_j takes these values. The vertical asymptotes of $p(\kappa_j)$ at $\kappa_j = 1$ and $\kappa_j = 0$ is a reflection of the probability density's singularities at these points. The density towards $\kappa_j = 0$ impacts the robustness of the tail of our prior [7](p.486), which influences the ability of our prior to give us posterior distributions for β_j that go against our prior assumption that β_j is distributed around zero. In other words, $\lim_{\kappa_j \to 0} p(\kappa_j) = \infty$ implies that our prior does not penalise significant β_j much at all. While this is useful for identifying significant β_j, we often would prefer some penalisation on significant β_j, to combat overfitting and collinearity's inflating effects. This is the problem which lead to the creation of the Finnish horseshoe, discussed later in 5.4.4. The density towards $\kappa_j = 1$ has influence over our prior's ability to reduce the unnecessary noise created by insignificant β_j.

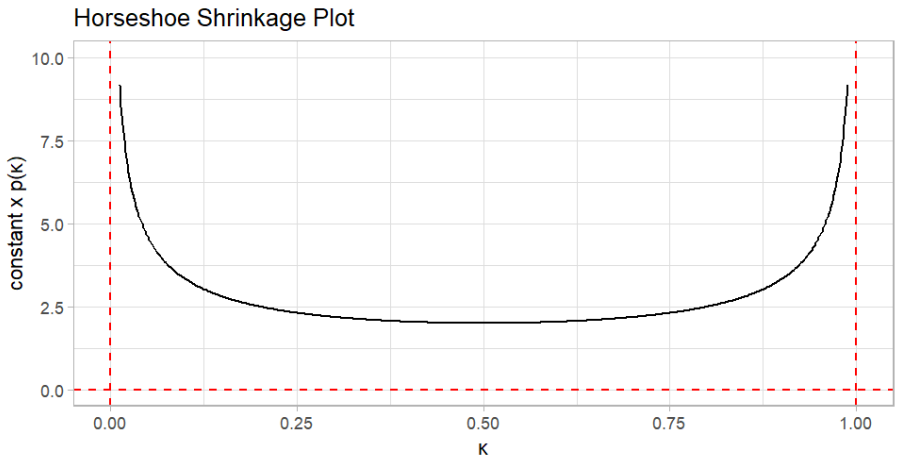

Figure 5.11: A plot of $p(\kappa_j)$ against κ_j for the horseshoe distribution.

To compare and contrast this graph with another sparsity inducing prior we consider the Strawderman-Berger prior specification for β_j, graphed in 5.12, and specified with respect to κ_j for convenience, [4];

$$\begin{aligned} \boldsymbol{y}_i|\boldsymbol{\beta}, \sigma^2 &\sim N(\boldsymbol{x_i}^T\boldsymbol{\beta}, \sigma^2) \\ \beta_j|\kappa_j &\sim N(0, \kappa_j^{-1} - 1) \;\; [N(0, \kappa_j^{-1} - 1) \equiv N(0, \lambda_j)] \\ \kappa_j &\sim Beta(1/2, 1), \text{ and so, } \; p(\kappa_j) \propto \kappa_j^{-\frac{1}{2}} \end{aligned}$$

From figure 5.12, we again observe a vertical asymptote at $\kappa_j = 0$, meaning that the

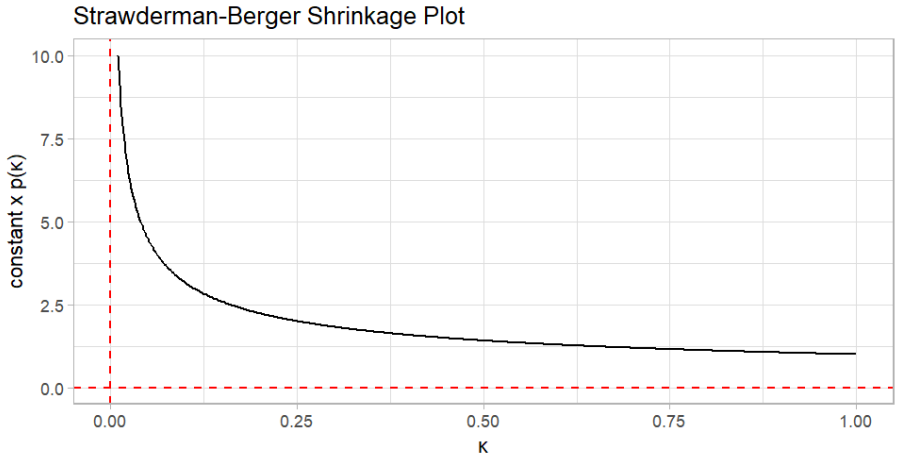

Figure 5.12: A plot of $p(\kappa_j)$ against κ_j for the Strawderman-Berger distribution.

Strawderman-Berger prior is capable of accounting for large values of β_j, however as $\kappa \to 1$ we observe that $p(\kappa) \to 0.5$ which suggests that it is inferior to the horseshoe prior with respect to noise reduction of insignificant β_j.

Example 5.4.3. *We wish to compare the effectiveness of our discussed continuous sparse priors against several other models. Using the same data as 5.4.2, we briefly assess the sparsity generated by the horseshoe and Strawderman-Berger priors. Code for these specifications can be found in the appendix, A.8 and A.9. We append the table from 5.4.2, and using the same data, we also measure the PMSE and we measure and report the beta distance. The beta distance column in our table suggests that in terms of inferential ability, the NIG prior offers the most accurate values of the β_j coefficients, however, for the purposes of variable selection it is lacking. Table 5.4 shows us that the horseshoe and*

Regression	$\hat{\beta}_j \mid (\beta_j = 0) = 0$	$\hat{\beta}_j \mid (\beta_j \neq 0) = 0$	PMSE	Beta Distance
Lasso	96%	44%	4538.0	3.25
NIG	62%	4%	1414.5	1.78
Continuous Spike-Slab	100%	2%	1261.6	2.25
Horseshoe	90%	12%	2999.9	2.12
Strawderman-Berger	84%	30%	3435.6	2.44

Table 5.4: Table comparing different regression types based on their ability to identify the significance/insignificance of coefficients, their PMSE values and their beta distances.

Strawderman-Berger priors result in stronger identification of insignificant β_j than the NIG prior, but weaker than the continuous spike-and-slab. Their identification of significant β_j as insignificant is between the very impressive continuous spike-and-slab and the frequentist lasso. The lasso's desirable trait is automatic variable selection. To achieve a similar result using Bayesian methods we must manually remove the data corresponding to the insignificant β_j coefficients. This is a time consuming and non-rigorous practice, but can potentially lead to improvements in predictive accuracy, for example upon carrying out manual variable selection for the horseshoe model by removing the covariates which have coefficients which have 0 contained within their 25% and 75% credible intervals, and then entering this new dataset into the model offers an incredible decrease in PMSE, going from $2999.9 \rightarrow 54.19$. This stark difference in predictive accuracy is a result of the reduction in random noise caused by removing the coefficients deemed irrelevant. The removal of insignificant coefficients also results in the model requiring less observations to learn more about the values of the remaining coefficients as it has less of them to estimate. Using this, when removing insignificant coefficients manually, it may be practical to remove the least significant coefficients (with the most posterior density concentrated around zero) first. After doing this we can rerun the regression, and as we are estimating fewer parameters we may be able to identify other insignificant predictors more accurately, leading to a powerful iterative approach to Bayesian variable selection. Because of the Bayesian framework, even coefficients we can identify as insignificant can potentially have large cumulative impacts on our predictive performance due to insignificant coefficients not being assigned an exact zero value (if they are not removed manually). Our Bayesian model is therefore accounting for the random noise generated by this irrelevant data, essentially our model is initially overfitted, and we can remedy this by removing the data which does not impact our response variable. From the figures in 5.13, we notice that the Strawderman-Berger prior results in posterior modes at zero in all coefficients plotted, even the significant $\beta_{50} = -10$. Compared to the horseshoe prior, it too greatly penalised significant coefficients. This is to be expected as it lacks the same asymptote at $\kappa_j = 1$ which is possessed by the horseshoe prior. The asymptote manifests itself in allowing larger values of β_j to go unpenalised. This characteristic leads to better variable selection using the 50% credible intervals. This is visible in the interval for β_1, which the Strawderman-Berger posterior deems insignificant, while the horseshoe posterior does not.

While the horseshoe prior seems to offer fair variable selection, it still exhibits large flaws, [19](p.1). Namely that horseshoe penalisation does not penalise significant coefficients effectively, as indicated by the κ_j graph, 5.11. While this is useful for inferring coefficient significance, it also has undesirable impacts on predictive performance, because significant coefficients are inflated due to collinearity and overfitting. An improvement upon the horseshoe could account for this problem. Furthermore, it would be useful if we could incorporate prior information about how sparse our model is. For example, let's consider the problem of determining what impacts human blood pressure. In this problem we may be dealing with thousands of insignificant predictor variables (as there are so many genes/life style choices which could possible impact it) and maybe only tens of significant predictors, which is vastly different to the data considered in 5.4.3, where 50%

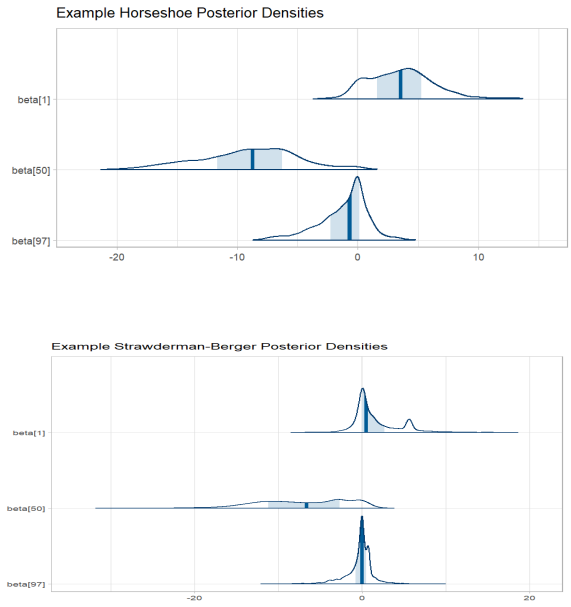

Figure 5.13: A plot of the posterior coefficient densities of β_1, β_{50}, β_{97} arising from the horseshoe prior model (top) and from the Strawderman-Berger model (bottom).

of predictor variables were significant. If we could inform our model about the amount of sparsity we expect to observe, it could perform better. Furthermore, the Cauchy priors used in the horseshoe (5.6), can result in an unstable MCMC sampler, [20](p.7). To tackle these issues, we introduce the Finnish horseshoe prior specification (also known as the regularised horseshoe), which was proposed in 2017 [19].

5.4.4 The Finnish Horseshoe

We introduce the Finnish horseshoe by drawing a comparison between the normal horseshoe, (5.6), and the spike-and-slab, (5.5) [5]. The spike-and-slab model uses a mixture of two normal distributions, one with low variance, the spike, and one with higher variance, the slab. Part of the power of the spike-and-slab is that even if the data indicates that the slab prior is more appropriate for a specific coefficient, it will still be penalised according to the normal prior specification we assign to the slab. This can be a ridge prior as we used, or any other possible value we could use as the variance term. This means that it is sparsity inducing and it also executes shrinkage of significant parameters, which is desirable as linear regression tends to obtain inflated values of $\hat{\boldsymbol{\beta}}$. The normal horseshoe is sparse, as it strongly penalises insignificant coefficients, however, it often lacks predictive power due to its inflation of significant coefficients. This issue is what the Finnish horseshoe attempts to solve. We specify;

$$
\begin{aligned}
\boldsymbol{y}|\boldsymbol{\beta}, \sigma^2 &\sim N_n(\alpha\mathbb{1} + \boldsymbol{X}\boldsymbol{\beta}, \sigma^2 I_n) \\
\beta_j|\tau, \tilde{\lambda_j} &\sim N(0, \tau^2\tilde{\lambda}_j^2) \\
\tilde{\lambda_j} &= \frac{c^2\lambda_j^2}{c^2 + \tau^2\lambda_j^2} \\
\lambda &\sim \text{half-Cauchy}(0, 1) \\
c^2 &\sim p(\cdot) \\
\tau &\sim p(\cdot).
\end{aligned}
$$

This prior is given by [19]p.6. We have used $p(\cdot)$ to denote that there are many possible prior distributions for τ and c^2. The intuition for our newly introduced parameter c^2 is

that when $c^2 \gg \tau^2\lambda_j^2$, we have that $\tilde{\lambda} \to \lambda_j$. This case is equivalent to that of the normal horseshoe, and we expect this when a parameter is insignificant. However, this prior differs to the normal horseshoe when $c^2 \ll \tau^2\lambda_j^2$, where we find that $\tilde{\lambda}_j^2 \to c^2\tau^{-2}$. Plugging this into our conditional prior for β_j gives us $\beta_j|\tau, \tilde{\lambda}_j \sim N(0, c^2)$. This situation can be viewed as equivalent to the slab prior part of the spike-and-slab, where for c^2 we can pick many different choices. In [19] they suggest the following prior specification for c^2,

$$c^2 \sim IG\left(\frac{\nu}{2}, \frac{\nu}{2}s^2\right).$$

If we marginalise out c^2 with this proposed prior, we obtain the marginal conditional prior of β_j to be $\beta_j \sim \text{Student-t}_\nu(0, s^2)$. This prior for β_j is a good choice for shrinkage in general. Furthermore, the inverse-gamma specification results in a heavy right-tail, allowing for the normal horseshoe properties to take over in sparse situations, as this prior specification is not averse to selecting large values of c^2. We select the hyper-parameters, ν and s^2 according to how much penalisation of larger coefficients we believe we will need, higher ν implies that c^2 will be lower and therefore our prior will act like the slab more readily. Higher values of s^2 imply that c^2 will be more varied and have higher values generally.

We must also specify a prior distribution for τ. A powerful suggestion by Michael Betancourt, [5], is the following;

$$\tau \sim \text{half-Cauchy}(0, \tau_0),$$
$$\tau_0 = \frac{m_0}{p - m_0}\frac{\sigma}{\sqrt{n}},$$

with m_0 being the number of β_j we expect to be significant, p as our number of coefficients, σ is the standard deviation of ϵ_i and n is our number of observations. Note that if $m_0 = p$, ie, we expect all of our coefficients to be significant we cannot use this prior due to τ_0 being a singularity. To understand this choice of prior, we must recall the meaning of τ. τ represents our beliefs about how spread out our significant coefficients are, in terms of standard deviation around 0. If we have no significant slopes, $m_0 = 0$, then $\tau_0 = 0$, and so $\tau \sim \text{half-Cauchy}(0, 0) = 0$ every time. This makes sense as then our $\beta_j \sim N(0, 0) = 0$, which is what we would want if we knew all coefficients were insignificant. Hence, a low m_0 tells our model to penalise more. Similarly, a low value of observations, n, suggests we are less certain about our coefficient values, hence we should have a higher variance for β_j. This prior is powerful as it allows us to incorporate more prior knowledge into our model to improve its performance. If we have no idea of m_0, we can set $m_0 = p/2$ to have this disappear from our prior.

Example 5.4.4. *We investigate the difference in results of the Finnish horseshoe and the normal horseshoe. The stan code for the Finnish Horseshoe is found here, A.10. Using the same data as in 5.4.2, we obtain table 5.5;*

Regression	$\hat{\beta}_j\|(\beta_j = 0) = 0$	$\hat{\beta}_j\|(\beta_j \neq 0) = 0$	PMSE	Beta Distance
Finnish Horseshoe	86%	16%	1254.7	3.78
Horseshoe	90%	12%	2999.9	2.12

Table 5.5: A table to compare the variable selection and predictive performance of the Finnish horseshoe prior against the horseshoe prior. Obtained using data from 5.4.2.

We observe very similar performance with respect to variable selection in this example, we could theoretically improve the identification of insignificant β_j by decreasing our value of m_0. For this simulation, we used $m_0 = 50$. A more thorough simulation study would be required to properly compare the two models, with multiple runs of the simulation with

multiple different varied datasets. This example is merely here to highlight the main, and most apparent advantages/disadvantages for each prior specification. The main difference we find in our results is the huge difference in PMSE. The Finnish horseshoe has a PMSE less than half that of the horseshoe. This highlights the importance of penalising significant coefficients. Particularly, when comparing the mean values of various coefficients, we find that the horseshoe often overestimates significant coefficients by a fair margin. For example, (with true values $\beta_{24} = 9.792$ *and* $\beta_{48} = -9.583$*) the horseshoe estimated* $\mathbb{E}(\beta_{24}) = 15.78$ *and* $\mathbb{E}(\beta_{48}) = -14.18$*, while the Finnish horseshoe estimated* $\mathbb{E}(\beta_{24}) = 11.90$ *and* $\mathbb{E}(\beta_{48}) = -10.88$*. The Finnish horseshoe has still overestimated these values, as the data will be suggesting that these two coefficients are larger than they actually are (which is not unlikely given* $n = 80$ *and* $p = 100$*), but it recognised that they were potentially inflated and penalised them accordingly. The horseshoe lacks this ability. Figure 5.14 shows another potential benefit of this shrinkage prior is that it appears to have lessened the amount of bi-modality present in the low but significant coefficients, like* $\beta_1 = 5$*. This results in small coefficients experiencing less extreme penalisation then, for example, the continuous spike-and-slab prior, as seen in figure 5.8. This lack of bi-modality is also good for interpretability.*

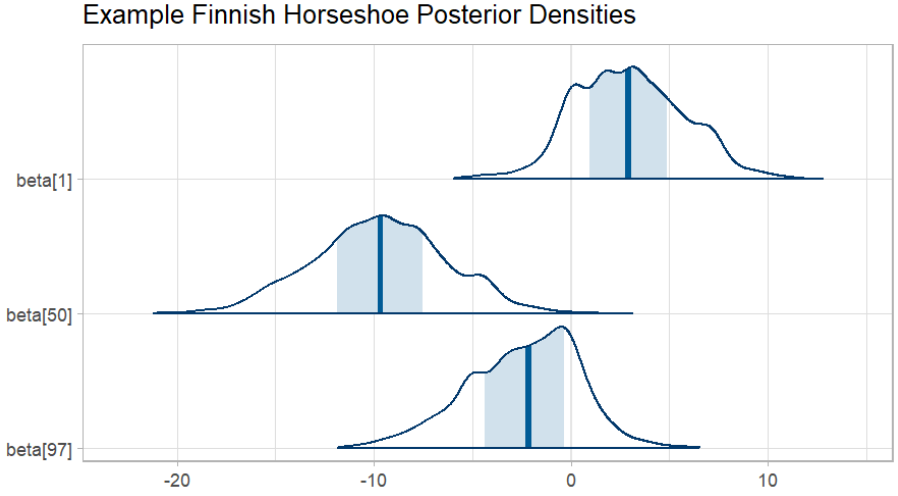

Figure 5.14: A plot of the Finnish horseshoe posterior densities of coefficients β_1, β_{50} and β_{97}.

There exist disadvantages to the Finnish horseshoe prior. The most significant being related to our specification of priors for the parameters c^2 and τ. Our choices of m_0, ν and s^2 have a large influence over the performance of our model, and if chosen poorly can lead to less useful models. However, the tunability of this model can also be seen as a part of its power. When used well, it executes desirable variable selection while not compromising on predictive accuracy.

5.5 Specifying a Custom Prior

In this section we devise our own penalising prior specification, which has certain desirable theoretical properties. We want our prior to be sparse and penalising, similar to the Finnish horseshoe. We use a similar "skeleton" as the Finnish horseshoe so we can use a

similar interpretation of our parameters. This goes as follows,

$$\begin{aligned}
\boldsymbol{y}|\boldsymbol{\beta},\sigma^2 &\sim N_n(\alpha\mathbb{1}+\boldsymbol{X}\boldsymbol{\beta},\sigma^2 I_n)\\
\beta_j|\tau,\lambda_j &\sim \text{distribution with zero mean, and standard deviation given by } \tau\lambda_j\\
\lambda_j &\sim \text{distribution with desirable properties}\\
\tau &\sim \text{half-Cauchy}\left(0,\frac{m_0}{p-m_0}\frac{\sigma}{\sqrt{n}}\right).
\end{aligned}$$

We start by choosing how we wish our graph of κ_j to appear, as this gives us a visualisation of how our prior will penalise. We wish for $\lim_{\kappa_j\to 0} p(\kappa_j) = 0$. This is because $\kappa_j \to 0 \implies \lambda_j \to \infty$, which means that $\kappa_j = 0$ suggests that our significant β_j will go unpenalised. This is undesirable and so we want the chances of this happening to be low, hence $\lim_{\kappa_j\to 0} p(\kappa_j) = 0$. Similarly, we want $\lim_{\kappa_j\to 1} p(\kappa_j) = \infty$. Because, in the case that $\kappa_j \to 1$ ($\implies \lambda_j \to 0$), we wish to see extreme penalisation of insignificant parameters. Essentially, we want a high chance of κ_j being large so that insignificant coefficients can be penalised to a greater extent.

These conditions lead to our proposed prior for κ_j, $p(\kappa_j) \propto -\kappa_j \log(1-\kappa_j)$. We can convert this prior into a prior for λ_j using the equation $\kappa_j = (1+\lambda_j^2)^{-1}$, as follows;

$$\begin{aligned}
p(\lambda_j) &= p(\kappa_j)\left|\frac{d\kappa_j}{d\lambda_j}\right|,\\
\left|\frac{d\kappa_j}{d\lambda_j}\right| &= \left|\frac{2\lambda_j}{(1+\lambda_j^2)^2}\right|,\\
p(\kappa_j) &\propto -\frac{1}{1+\lambda_j^2}\log\left(1-\frac{1}{1+\lambda_j^2}\right),\\
&= -\frac{1}{1+\lambda_j^2}\log\left(\frac{\lambda_j^2}{1+\lambda_j^2}\right),\\
p(\lambda_j) &\propto -\frac{2\lambda_j}{(1+\lambda_j^2)^3}\log\left(\frac{\lambda_j^2}{1+\lambda_j^2}\right),
\end{aligned}$$

To check the properness (and to find the normalising constant) of $p(\lambda_j)$ we wish to integrate,

$$\begin{aligned}
I &= \int_0^\infty -\frac{2\lambda_j}{(1+\lambda_j^2)^3}\log\left(\frac{\lambda_j^2}{1+\lambda_j^2}\right),\\
&= \frac{3}{4}(\text{using numerical methods}),\\
\implies p(\lambda_j) &= -\frac{8\lambda_j}{3(1+\lambda_j^2)^3}\log\left(\frac{\lambda_j^2}{1+\lambda_j^2}\right).
\end{aligned}$$

We further increase the sparsity of our model by using a double-exponential prior for β_j. This distributions sharp peak results in increased sparsity compared to a normal prior. As such we have our completed prior specification (which we dub the 'Burn prior') given

by,

$$\begin{aligned}
\boldsymbol{y}|\boldsymbol{\beta}, \sigma^2 &\sim N_n(\alpha\mathbb{1} + \boldsymbol{X}\boldsymbol{\beta}, \sigma^2 I_n) \\
\beta_j|\tau, \lambda_j &\sim \text{Laplace}(0, \tau\lambda_j) \\
\lambda_j &\sim p(\cdot), \text{where } p(\lambda) = -\frac{8\lambda_j}{3(1+\lambda_j^2)^3}\log\left(\frac{\lambda_j^2}{1+\lambda_j^2}\right) \\
\tau &\sim \text{half-Cauchy}\left(0, \frac{m_0}{p-m_0}\frac{\sigma}{\sqrt{n}}\right).
\end{aligned}$$

The graphs in figure 5.15 and figure 5.16 aid in the visualisation of what we expect this prior specification to do. We observe from figure 5.15 that our proposed prior has a sharper peak and looks again like it is a pinched version of the ridge prior (similar to the horseshoe prior), this should increase sparsity. Figure 5.16 helps us visualise the penalising effects our prior distribution will have, we expect it to penalise significant coefficients greatly, as $\lim_{\kappa_j \to 0} p(\kappa_j) = 0$ and insignificant coefficients also greatly as $\lim_{\kappa_j \to 1} p(\kappa_j) \to \infty$.

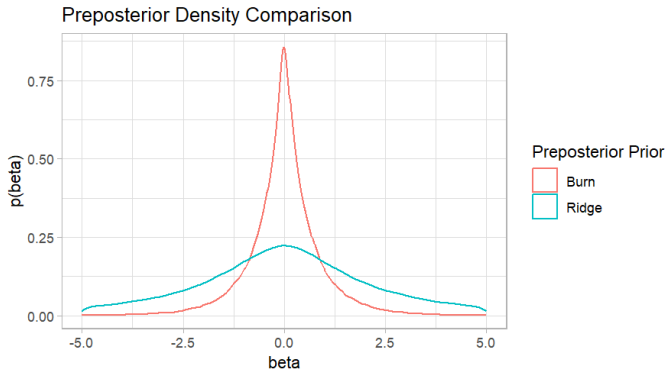

Figure 5.15: This graph shows the difference between our proposed prior (the Burn prior) and a standard ridge prior.

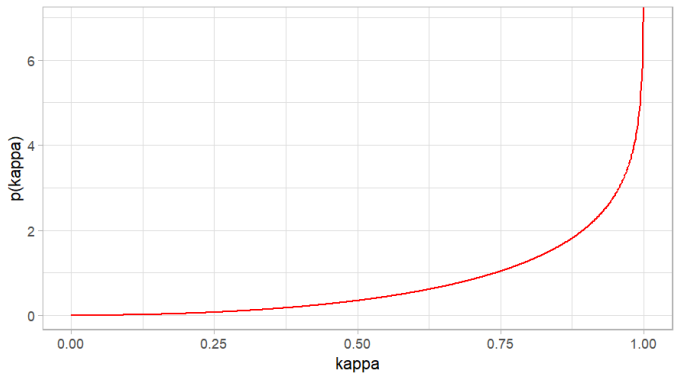

Figure 5.16: A plot of $p(\kappa_j)$ against κ_j for our proposed prior distribution.

Regression	$\hat{\beta}_j\|(\beta_j = 0) = 0$	$\hat{\beta}_j\|(\beta_j \neq 0) = 0$	PMSE	Beta Distance
Finnish Horseshoe	86%	16%	1254.7	3.78
Horseshoe	90%	12%	2999.9	2.12
Burn Prior	84%	8%	2304.2	2.05

Table 5.6: The table obtained using data from 5.4.2. We have extended table 5.5 to include our proposed Burn prior.

We apply our proposed prior to data generated in 5.4.2 and create table 5.6. Our proposed prior seems to be competitive with the horseshoe and Finnish horseshoe priors, with it generally appearing as a compromise between the two in terms of predictive accuracy and competitive variable selection. As we expected, it is good at identifying and applying extreme shrinkage to many insignificant coefficients. However, investigating deeper shows that, in spite of overall fair variable selection, many of the estimates of the zero coefficients, β_j for $j \in \{51, \dots, 100\}$, are quite large. This may be a consequence of the shape of our prior distribution. If we refer to the Burn prior distribution, 5.15, we notice that we have a large amount of probability density in the region $\beta = [-2, 2]$. In other words, our prior looks like a spike as the others do, but the base of the spike is potentially too wide to allow for proper penalisation of many insignificant coefficients. This results in β_j being underpenalised if the training data suggests that they are near the edges of this region. The horseshoe and Finnish horseshoe are superior in this regard as their prior distributions have thinner spikes. This diffuse base may be a result of our original specification for $p(\kappa_j)$. Although it has an asymptote at $\kappa_j = 1$, it is a $\log(\cdot)$ asmyptote, meaning that the slope as $\kappa_j \to 1$ is potentially not steep enough to execute great penalisation of insignificant coefficients.

5.6 Comparing Variable Selection Schemes

In the Bayesian paradigm, we lack the luxury afforded to frequentists of automatic variable selection, which occurs in many of frequentist penalisation techniques (such as the lasso). As such we must consider our own way to judge which variables should be included. In previous examples we selected an equal tailed credible interval, and if this interval contained zero we considered the variable insignificant. Recall, a $(1-\alpha)\%$ EQT credible interval for a random variable X implies that $\mathbb{P}(X \leq l) = (\alpha/2)\%$ and $\mathbb{P}(X \geq u) = (\alpha/2)\%$ implies that the $(1-\alpha)\%$ EQT interval for $\boldsymbol{X}$ is given by $[l, u]$. Our choice for the percentage, $(1-\alpha)$, of our equal tailed interval was arbitrary. In this brief section we determine a fair, standardised way to select variables using Bayesian regression. This is so that in our following simulations, we can compare the variable selection methods of the different penalising priors. We say that the optimal EQT interval minimises the distance criterion, which was proposed here [18] and used in the simulation study given here [20].

$$\text{The distance criterion} := \sqrt{(1 - \text{correct inclusion rate})^2 - (\text{false inclusion rate})^2}.$$

Once we find the optimal EQT percentage according to the distance criterion of our model given our data, we wish to have an empirical method of comparing the variable selection ability. For this task, we employ Matthew's correlation coefficient (MCC). MCC was proposed by B.W Matthews in [15].

$$\text{MCC} := \frac{TP \times TN - FP \times FN}{\sqrt{(TP + FP) \times (TP + FN) \times (TN + FP) \times (TN + FN)}}.$$

- TP refers to the sum of true positives. We have a true positive when our model correctly believes a coefficient to be significant. FP is the number of times our model believes an insignificant predictor to be significant.
- TN refers to the sum of true negatives, which refers to the number of times our model correctly identifies a coefficient as insignificant. FN is the number of times we falsely identify a significant predictor is insignificant.[10]

In the appendix we list the functions written to assess these values using RStan's posterior samples, (A.12).

[10]The distance criterion and MCC both require knowledge of the true values of $\boldsymbol{\beta}$. This makes them unusable on real datasets, however, with simulation studies, we control the value of $\boldsymbol{\beta}$ and so they are useful tools for assessing variable selection.

Chapter 6

Simulation Study

In this chapter we will compare the performance of different Bayesian shrinkage priors and frequentist methods. We will use the full-Bayes method while fitting models, and we will report statistics for comparison for each of them. We will be simulating each model on each of simulated dataset 20 times. For variable selection comparison we will employ algorithms which determine the EQT which minimises the distance criterion, and then report the MCC of the model using the optimal EQT. We determine a variable to be insignificant if zero lies within its optimal EQT. We shall consider 6 cases of data generation.

1. We consider $N_{train} = 300$, $N_{test} = 300$, $\boldsymbol{\beta} = (-5, 5)^T$, $\sigma^2 = 16$ and $\boldsymbol{X}_{ij} \overset{iid}{\sim} N(0, 1)$.

2. We consider $N_{train} = 50$, $N_{test} = 100$ and $\boldsymbol{\beta}$ is given by a sequence between -20 and -10 of length 25 and a sequence between 10 and 20 of length 25 (giving 50 predictors in total). We again choose $\sigma^2 = 16$ and $\boldsymbol{X}_{ij} \overset{iid}{\sim} N(0, 1)$.

3. We consider $N_{train} = 200$, $N_{test} = 200$ and $\boldsymbol{\beta}$ is a sequence between $-10 \to -5$ of length 10 and a sequence of $5 \to 10$ of length 10, with $\beta_j = 0$ for $j \in \{31, \dots, 50\}$. We will take $\sigma^2 = 9$ and $\boldsymbol{X}_{ij} \overset{iid}{\sim} N(0, 1)$.

4. We consider $N_{train} = 30$, $N_{test} = 200$ and $\boldsymbol{\beta}$ is a sequence between $-10 \to -5$ of length 10 and a sequence of $5 \to 10$ of length 10, with $\beta_j = 0$ for $j \in \{31, \dots, 50\}$. We will take $\sigma^2 = 9$ and $\boldsymbol{X}_{ij} \overset{iid}{\sim} N(0, 1)$

5. We consider $N_{train} = 30$, $N_{test} = 200$ and $\boldsymbol{\beta}$ is a sequence between $10 \to 15$ of length 10 and a sequence of $-15 \to -10$ of length 10, with $\beta_j = 0$ for $j \in \{31, \dots, 50\}$. We will take $\sigma^2 = 1$ and $\boldsymbol{X}_i \sim MVN(\boldsymbol{0}_{50}, \Sigma)$, where Σ is the covariance matrix describing $\mathbb{V}ar(\boldsymbol{X}_{ij}) = 9$ (for $i = j$) and covariances (with $j \neq k$) given by $\mathbb{C}ov(\boldsymbol{X}_{ij}, \boldsymbol{X}_{ik}) = 5$ for $j, k \in \{1, \dots, 5\}$, $\mathbb{C}ov(\boldsymbol{X}_{ij}, \boldsymbol{X}_{ik}) = 8$ for $j, k \in \{6, \dots, 15\}$, $\mathbb{C}ov(\boldsymbol{X}_{ij}, \boldsymbol{X}_{ik}) = 9$ (super-collinearity) for $j, k \in \{16, \dots, 20\}$ and $\mathbb{C}ov(\boldsymbol{X}_{ij}, \boldsymbol{X}_{ik}) = 0$ for $j, k \in \{21, \dots, 50\}$.

6. We consider $N_{train} = 30$, $N_{test} = 200$ and $\boldsymbol{\beta}$ is a sequence between $10 \to 15$ of length 10 and a sequence of $-15 \to -10$ of length 10. We will take $\sigma^2 = 1$ and $\boldsymbol{X}_i \sim MVN(\boldsymbol{0}_{50}, \Sigma)$, where Σ is the covariance matrix describing $\mathbb{V}ar(\boldsymbol{X}_{ij}) = 9$ (for $i = j$) and covariances (with $j \neq k$) given by $\mathbb{C}ov(\boldsymbol{X}_{ij}, \boldsymbol{X}_{ik}) = 5$ for $j, k \in \{1, \dots, 5\}$, $\mathbb{C}ov(\boldsymbol{X}_{ij}, \boldsymbol{X}_{ik}) = 8$ for $j, k \in \{6, \dots, 15\}$ and $\mathbb{C}ov(\boldsymbol{X}_{ij}, \boldsymbol{X}_{ik}) = 9$ (super-collinearity) for $j, k \in \{16, \dots, 20\}$

Case 1 serves as a way to compare our regression methods in an ideal setting in which all the assumptions of the linear model are met nicely with plenty of training data. In this case we may expect the OLS regression to be the superior method. Case 2 will show us how the different regressions interact with significant predictors, but in a low data

environment. Case 3 should give us information on how the relevant regression methods produce sparsity in a high data environment, while case 4 does the same in a low data environment. Case 5 also checks sparsity, with the difference of highly collinear significant predictors. Case 6 checks performance with highly collinear, but not sparse, data.

The simulation code can be found in the appendix, with the sourced functions here A.13 and the file which carries out the simulations for each case here A.14. We summarise the results by taking means of the 20 reported values for each cell in each of the one through to six cases respectively in the tables 6.1, table 6.2, table 6.3, table 6.5, table 6.6 and table 6.7. Results from each of the 20 simulations done for each regression method for each case can be found at this link simulation case data. For clarity, results in the table which use Bayesian methods have a 'b' prefix, and we give frequentist methods a 'f' prefix. We summarise our main findings for each case in the following list,

1. All Bayesian regressions offer near identical results with the exception of computation time, this is due to the influence of the likelihood on our posterior distribution being much greater than the influence of our prior, because of the large amount of data we have to train each model. Because there is plenty of data for our number of predictor variables we expect OLS regression to be optimum. From our simulations we find that the PMSE of OLS and the Bayesian techniques are identical. The frequentist penalisation techniques offer slightly higher PMSEs, potentially due to their bias in coefficient estimation, which does not disappear even when we observe lots of training data.

Regression	PMSE	ELPD	σ^2	Time/s
bRidge	16.4	-840	15.8	8.86
bLasso	16.4	-840	15.7	12.3
bNet	16.4	-840	15.8	12.6
bSpikeSlab	16.4	-840	15.7	9.11
bHorseshoe	16.4	-840	15.7	9.39
bFinnishHorse.	16.4	-840	15.8	11.0
bUninformative	16.4	-840	15.9	8.81
fRidge	17.9			0.07
fLasso	17.7			0.053
fNet	17.7			0.062
fOLS	16.4			0.002

Table 6.1: Table of results for case 1.

2. We see that OLS and the uninformative prior Bayesian model both exhibit huge PMSEs, indicating terrible predictive accuracy in small data training sets. Bayesian ridge regression offers the lowest average PMSE and ELPD values suggesting that this model is optimal for prediction. Also, notice that the "worst" Bayesian regression for this case (excluding the uninformative model) is that of spike-and-slab. This is due to the assumption of the data being generated either from the spike or the slab distribution is dubious, due to the lack of sparsity in this dataset. In spite of this, it offers better PMSE than the "best" frequentist method for this data, the elastic net. We also observe that the estimates of σ^2 are massively increased when compared to case 1, indicating that our Bayesian models greatly overestimate σ^2 in small training datasets. A potential improvement to our Bayesian models would use a prior to shrink the posterior estimates of σ^2, to combat the huge inflation of its estimate in low data scenarios.

Regression	PMSE	ELPD	σ^2	Time/s
bRidge	632	-199	70.3	39.8
bLasso	2150	-226	242	18.6
bNet	972	-206	292	1315
bSpikeSlab	1097	-219	470	246
bHorseshoe	4545	-262	1089	48.7
bFinnishHorse.	1052	-212	440	1449
bUninformative	1.04e8	-463	4.6e6	96.9
fRidge	9641			0.121
fLasso	5429			0.136
fNet	4817			0.127
fOLS	63636			0.001

Table 6.2: Table of results for case 2.

3. Case 3's large and sparse training datasets seem to provide the perfect environment for the spike-and-slab model, which has the best predictive performance for this case. All Bayesian regressions impressively exhibited flawless variable selection in all 20 simulations under this case. Furthermore, they all offered good estimates for σ^2. In contrast, the frequentist lasso and elastic net consistently identified some insignificant predictors as significant. This worse variable selection is captured by their lower MCC scores.

Regression	PMSE	ELPD	MCC	Cor. Inc.	False Inc.	Opt. EQT	σ^2	Time/s
bRidge	12.2	-535	1	1	0	0.985	9.36	9
bLasso	12.1	-533	1	1	0	0.983	9.21	11.8
bNet	12.1	-534	1	1	0	0.980	9.37	57.6
bSpikeSlab	10.5	-518	1	1	0	0.742	9.01	14.9
bHorseshoe	11.4	-526	1	1	0	0.958	8.92	25.4
bFinnishHorse.	11.4	-526	1	1	0	0.950	8.98	37.9
bUninformative	12.3	-535	1	1	0	0.983	9.43	9.22
fRidge	16.1			1	1			0.134
fLasso	12.3		0.74	1	0.253			0.081
fNet	12.4		0.674	1	0.330			0.0725
fOLS	12.3			1	1			0.0015

Table 6.3: Table of results for case 3

4. This case had sparse data with a small training dataset, which is a difficult combination for linear regression to deal with in general. Because of this, we see that the PMSEs are about $\times 40$ larger than in case 3. We find the σ^2 estimates to be gravely inflated as in case 2. We again notice that the "worst" Bayesian method (excluding the uninformative prior), the horseshoe, offers better predictive performance than the best frequentist method, the elastic net. The Bayesian ridge offered the lowest average PMSE, and the highest MCC, leading us to believe it is potentially optimal for this data environment. This could be due to the more diffuse normal prior distribution better representing our uncertainty in our beliefs about the coefficients with small training datasets. Meanwhile, priors with sharper peaks suppose we have more prior certainty about the value of the coefficients which can be a detrimental assumption. However, the ELPD suggests that the Bayesian lasso edges out the Bayesian ridge. We calculate using pseudo-Bayesian weights that the Bayesian lasso is the 'most likely' model, which we can interpret as the model which offers optimal Bayesian predictive performance. These weights are shown in table 6.4 below,

Regression (Case 4)	Pseudo-Weights
bRidge	0.258
bLasso	0.702
bNet	0.000
bSpikeSlab	0.035
bHorseshoe	0.005
bFinnishHorseshoe	0.000
bUninformative	0.000

Table 6.4: Pseudo-Bayesian model weights for case 4.

Regression	PMSE	ELPD	MCC	Cor. Inc.	False Inc.	Opt. EQT	σ^2	Time/s
bRidge	478	-108	0.611	0.77	0.16	0.65	31.7	48.7
bLasso	489	-107	0.579	0.79	0.21	0.52	37.2	24.9
bNet	543	-121	0.510	0.75	0.25	0.51	140	301
bSpikeSlab	473	-110	0.593	0.80	0.20	0.49	45.1	152
bHorseshoe	580	-112	0.504	0.73	0.23	0.42	81.8	92
bFinnishHorse.	507	-117	0.585	0.80	0.21	0.47	83.0	126
bUninformative	9.7e10	-274	0.090	0.57	0.483	0.29	2.6e6	140
fRidge	1148			1	1			0.127
fLasso	785			0.465	0.138			0.103
fNet	746			0.517	0.167			0.104
fOLS								

Table 6.5: Table of results for case 4.

5. Case 5 investigates model performance under a small training dataset with sparse predictors and collinear data. We notice that with this data the Bayesian ridge struggled significantly compared to the other cases, with a high PMSE. Interestingly, it has the best ELPD, and so cross-validation suggests that it is the best Bayesian regression method, which contradicts what the PMSE value is telling us. We also see that the Bayesian elastic net has very poor average performance, most likely as a result of divergent MCMC chains, although even when it did not diverge it offered a lowest PMSE of 6279, which is not very impressive. Divergence of MCMC chains can usually be diagnosed by inspecting the time taken of the MCMC sampler; in the case of divergence, it will be very slow. The spike-and-slab model offers the best variable selection for this dataset, in spite of a poor ELPD. σ^2 estimates are very inflated. The frequentist ridge suffered greatly, having an even higher PMSE than the divergent Bayesian elastic net. The other frequentist methods seem to have out performed the Bayesian ridge regression, but not performed as well as any of the other Bayesian methods[1]

[1]The Bayesian elastic net had divergent MCMC chains, resulting in poor performance. Some chains did not diverge though, and offered similar performance to the other Bayesian techniques.

Regression	PMSE	ELPD	MCC	Cor. Inc.	False Inc.	Opt. EQT	σ^2	Time/s
bRidge	4879	-125	0.410	0.60	0.20	0.68	103	126
bLasso	2980	-127	0.435	0.68	0.24	0.58	129	78.5
bNet*	36340	-58226	0.156	0.58	0.42	0.46	8732	569
bSpikeSlab	2313	-154	0.786	0.89	0.097	0.22	1938	240
bHorseshoe	2316	-128	0.281	0.58	0.31	0.47	171	188
bFinnishHorse.	2396	-130	0.304	0.66	0.36	0.41	171	276
bUninformative	1.6e5	-206	0.0567	0.542	0.487	0.342	4.6e4	131
fRidge	52903			1	1			0.158
fLasso	3575		0.419	0.50	0.12			0.127
fNet	3444		0.505	0.62	0.14			0.158
fOLS								

Table 6.6: Table of results for case 5.

6. This case offers the largest contrast between the frequentist and Bayesian models. The non-sparse data alongside a low true $\sigma^2 = 1$ has allowed the (non-divergent) Bayesian models to have an extremely small PMSE of around ≈ 2.5. We also see that the removal of sparsity from case 5 has lead to much better estimates of σ^2. The colossal difference in predictive performance of the Bayesian and frequentist models here provide great motivation for the study of Bayesian linear regression[2].

Regression	PMSE	ELPD	σ^2	Time/s
bRidge	2.52	-53.6	1.68	92.7
bLasso	2.45	-55.6	1.8	85.1
bNet*	9907	-108758	3527	261
bSpikeSlab*	710	-126	1271	134
bHorseshoe	2.41	-53	1.49	112
bFinnishHorse.	2.44	-52.5	1.45	145
bUninformative	2260	-159	2860	95.6
fRidge	1084			0.116
fLasso	616			0.113
fNet	694			0.136
fOLS				

Table 6.7: Table of results for case 6.

[2]The Bayesian elastic net and Bayesian spike-and-slab suffered from divergent MCMC chains, causing poor average performance for both models in this case.

Chapter 7

Conclusion

This project aimed to motivate and explain the use of frequentist penalisation techniques as well as Bayesian shrinkage priors. Our main findings were that the Bayesian shrinkage priors were often superior in terms of predictive performance, and very competitive with regards to inducing sparsity, even if true variable selection requires manual removal of predictors, when compared to their equivalent frequentist penalisation methods. We also saw that, in spite of having MAP estimates which coincided with frequentist loss function minimisation problems, it was hard to definitively relate the practical results given by Bayesian shrinkage priors with their frequentist namesakes. Furthermore, our simulation study provides a useful method of comparing the performance of different types of regression when provided with different datasets. Case 1 in our simulation study showed that the Bayesian techniques offer near identical results to OLS in the ideal case of linear regression where $n \gg p$ and all other assumptions were met, while the frequentist penalisation methods kept their theoretical biases even when the data provided conclusive evidence against it. The flexible Bayesian relationship between the posterior and its prior and likelihood result in powerful linear regressions which can be used effectively on many datasets. Adapting our prior choice depending on the data we are modelling can greatly improve predictive and inferential performance. For instance, concerning case 3 of our simulation study, the spike-and-slab prior roughly matched the situation in which the coefficients were selected, ie, from two distributions with one tightly centered about zero and one distribution more spread out from zero. This fact gave it superior performance, in spite of the large sample size which usually would give all Bayesian methods very similar results. However, it should be noted that if our $\boldsymbol{\beta}$ prior is at odds with the 'true' coefficient values, it could weaken the inferential and predictive power of the model, and so, in general it is good to use multiple models for real datasets. The frequentist methods tested also exhibit benefits and drawbacks. The automatic variable selection performed by the lasso and elastic net are valuable inferential tools. Although they generally did not perform as well as the Bayesian methods in our simulations, it must be acknowledged that the Bayesian methods were in an ideal (and rare) scenario in which we knew the true $\boldsymbol{\beta}$ values, which gave us the means to adapt and refine our inferences on dataset sparsity. Because of this, and their accessibility alongside their low computation times, they make for highly practical regression methods in realistic scenarios for making inferences about predictors and responses. An ideal dataset analysis would use a multitude of different models from both paradigms to obtain a fair and complete view of any given linear relationship between some data and a response.

Bibliography

[1] A. G. Aki Vehtari and J. Gabry. Practical bayesian model evaluation using leave-one-out cross-validation and waic. pages 1–5, 2017.

[2] Q. G. Angelika Stefab, Dimitris Katsimpokis and E.-J. Wagenmakers. Expert agreement in prior elicitation and its effects of bayesian inference. 2022.

[3] S. Banerjee. Bayesian linear model: Gory details. pages 2, 3, 8.

[4] J. Berger. A robust generalized bayes estimator and confidence region for a multivariate normal mean. pages 716–761, 1980.

[5] M. Betancourt. Bayes sparse regression. 2018.

[6] M. Betancourt. Sparsity blues. 2021.

[7] C. Carvalho and N. Polson. The horseshoe estimator for sparse signals. page 3, 2010.

[8] A. Gelfand. Gibbs sampling. pages 1–2, 2000.

[9] A. Gelman. Prior distributions for variance parameters in hierarchical models. page 522, 2006.

[10] A. E. Hoerl and R. W. Kennard. Ridge regression: Biased estimation for nonorthogonal problems. page 1, 1970.

[11] B. Jiang and Q. Sun. Bayesian high-dimensional linear regression with generic spike-and-slab priors. 2019.

[12] I. Jolliffe and J. Cadima. Principal component analysis: a review and recent developments. pages 1–4, 2016.

[13] K. M. I. Katharine M. Banner and T. J. Rodhouse. The use of bayesian priors in ecology: The good, the bad and the not great. 2019.

[14] T. Lewis and B. Carlin. Bayesian methods for data analysis. 2008.

[15] B. Matthews. Comparison of the predicted and observed secondary structure of t4 phage lysozyme. pages 442–451, 1975.

[16] M. G. Minjung Kyung, Jeff Gill and G. Casella. Penalized regression, standard errors, and bayesian lassos. page 31, 2010.

[17] T. Park and G. Casella. Bayesian lasso. *Journal of the American Statistical Association*, page 1, 2008.

[18] N. J. Perkins and E. F. Schisterman. The inconsistency of "optimal" cutpoints obtained using two criteria based on the receiver operating characteristic curve. *American Journal of Epidemiology*, pages 670–675, 2006.

[19] J. Piironen and A. Vehtari. Sparsity information and regularization in the horseshoe and other shrinkage priors. pages 1,4, 2017.

[20] J. M. Sara van Erp, Daniel L. Oberski. Shrinkage priors for bayesian penalized regression. *Journal of Mathematical Psychology*, page 35, 2019.

[21] T. H. Stephen Bates and R. Tibshirani. Cross-validation: what does it estimate and how well does it do it? 2022.

[22] S. D. Team. Rstan: the r interface to stan. 2024. URL https://mc-stan.org/.

[23] R. Tibshirani. Regression shrinkage and selection via the lasso. 1996.

[24] R. T. Trevor Hastie and J. Friedman. The elements of statistical learning. 2001.

[25] R. T. Trevor Hastie and B. Narasimhan. The relaxed lasso. 2023.

[26] A. Tsun. 7. statistical estimation, chapter 7.5, maximum a posteriori estimation. page 4.

[27] W. N. van Wieringen. Lecture notes on ridge regression. pages 5–12, 2023.

[28] J. R. Veerman. Estimating error and prior variance in a high-dimensional ridge regression models. page 11, 2018.

[29] H. Zou and T. Hastie. Regularization and variable selection via the elastic net. *Journal of the Royal Statistical Society Series B: Statistical Methodology*, page 1, 2005.

Appendices

Appendix A

Appendices

A.1 Example 2.2.2

(2.2.2)

```
## Example 2.2.2 Simulated Collinearity
library(MASS) ## Required for the mvrnorm function

set.seed(42)  ## For reproducibility
variance <- matrix(c(1, 0, 0,
                     0, 1, 0.9,
                     0, 0.9, 1), nrow = 3, byrow = TRUE)

## The variance matrix for our data generation.
## Notice the high covariance between the
## Second and Third parameters.

n <- 10 ## Number of rows of our design matrix.
mu <- c(0, 0, 0)  ## The mean values of our predictors
\boldsymbol{X} <- mvrnorm(n, mu, variance)

beta <- c(5, 5, 5)  ## The true values of our beta's with respect
                    ## to the predictor variables
Y <- \boldsymbol{X} %*% beta + rnorm(n, mean = 0, sd = 1)
summary(lm(Y ~ \boldsymbol{X}))
```

A.2 Example 3.2.3

3.2.3

```
## PMSE example 3.2.3
library(faraway) ## Imports interesting data sets
set.seed(42) ## For reproducibility
?worldcup ## Tells us about the data set we are interested in
data("worldcup")
head(worldcup)
# We wish to understand how Time Played, Saves and Tackles predicts
# number of shots taken by each player in the 2010 world cup.
## We want to compare two models and see which is most effective,
## will it be the ridge model or the least squares model?
training.rows <- sample(1:nrow(worldcup), floor(0.8*nrow(worldcup)))
```

```
### scale our covariates and separate training/testing rows
worldcupRAW <- worldcup[,c(3,4,6,7)]
worldcup <- scale(worldcupRAW)
training.set <- data.frame(worldcup[training.rows,])
testing.set <- data.frame(worldcup[-training.rows,])

ridge.modelMat.train <- model.matrix(Shots ~ Time + Tackles + Saves,
                                     data = training.set)
ridge.modelMat.test <- model.matrix(Shots ~ Time + Tackles + Saves,
                                    data = testing.set)
worldcup.ls <- lm(Shots ~ Time + Tackles + Saves,
                  data = training.set )
library(glmnet) ## Allows us to use ridge regression
## Picks an appropriate lambda value and fits a ridge regression to the data
worldcup.ridge <- cv.glmnet(ridge.modelMat.train, training.set$Shots, alpha = 0)
pred.ls <- predict(worldcup.ls, newdata = testing.set)
pred.ridge <- predict(worldcup.ridge, ridge.modelMat.test)
# PMSE = E[(y_hat - y)^2], where y and y_hat are obtained from the test sets.
# y_hat (= pred.ls, pred.ridge) represents the predicted response given the x.test predic

## put predictions back on original scale
pred.ls <- pred.ls*sd(worldcupRAW$Shots)+mean(worldcupRAW$Shots)
pred.ridge <- pred.ridge*sd(worldcupRAW$Shots)+mean(worldcupRAW$Shots)

PMSE.ls <- mean((pred.ls - testing.set$Shots)^2)
PMSE.ridge <- mean((pred.ridge - testing.set$Shots)^2)
```

A.3 Data Generation (parameters tuned for 4.3.1)

4.3.1

```
## Data Simulation for Model Testing

library(MASS)

## Parameter Definitions
set.seed(42)  ## for reproducibility
N <- 100  ## number of observations
total.p <- 3  ## number of predictor variables
real.p <- 2   ## number of non-zero predictor variables
x.sd <- 0
mu <- rnorm(total.p, 0, x.sd) ## means for values of generated data
data.cov <- 0  ##covariance between x-data obs when generated by mvrnorm
variance <- 1  ##variance along diagonal of generated x data
varmat <- matrix(rep(data.cov, total.p*total.p), nrow = total.p)
diag(varmat) <- variance
beta <- c(0.5, 2,
          rep(0, total.p - real.p)) ## true beta values
y.sd <- 3 ## sigma

## for loop eliminates covariance between real and fake predictors
for(i in (real.p+1):total.p){
  for(j in 1:total.p ){
    if(i !=j){
```

```
      varmat[i,j] <- 0
      varmat[j,i] <- 0
    }
  }
}
##
## Data Generation Functions
gen_x.norm <- function(N, total.p,
                       mu, varmat){
  return(mvrnorm(N, mu, varmat))
}
gen_y.data <- function(beta, x, sd){
  return(x%*%beta + rnorm(nrow(x), 0, sd))
}
##
x <- gen_x.norm(N, total.p, mu, varmat)
y <- gen_y.data(beta, x, y.sd)

beta # true beta values

## let's set up penalised regression:
## using 80% of simulations to train the model and 20% to test
training.rows <- sample(1:N, floor(0.8*N))

x.train <- x[training.rows, ]
y.train <- y[training.rows]

x.test <- x[-training.rows, ]
y.test <- y[-training.rows]
#####
```

A.4 Stan NIG prior

4.3.1

```
data {
  int<lower=0> N;
  int<lower=0> K;
  matrix[N, K] x;
  vector[N] y;
  real<lower=0>lambda;
}
parameters {
  vector[K] beta;
  real<lower=0> sigma2;
  // with alpha intercept (we include/exclude alpha in all our stan models depending on n
  // each specific problem)
  real alpha;
}
model {
  sigma2 ~ inv_gamma(0.5, 2); // a reasonable prior spec
  beta ~ normal(0, sqrt(sigma2/lambda));
  y ~ normal( x*beta + alpha, sqrt(sigma2));
```

```
}
```

A.5 LOO-CV in RStudio

5.3.3

```
###CROSS VALIDATION
## normal-inverse-gamma model, which reports the log likelihood
model <- stan_model('NIGLOO.stan')

## A function to calculate the PMSE (for comparison with LOO)
f.PMSE <- function(lambda){
  data = list(N = nrow(x.train),
              K = ncol(x.train),
              x = x.train,
              y = y.train,
              lambda = lambda,
              x_new = x.test,
              N_new = nrow(x.test))
  y.new <- extract(sampling(model,data,chains = 1), pars = 'y_new')$y_new
  return(PMSEcalc(colMeans(y.new), y.test))
}
f.loo <- function(lambda){
  data = list(N = nrow(x.train),
              K = ncol(x.train),
              x = x.train,
              y = y.train,
              lambda = lambda,
              x_new = x.test,
              N_new = nrow(x.test))
  ## optim seeks to minimise a function,
  ## hence we take the negative of the ELPD,
  ## as usually we prefer to maximise it
  return(-loo(sampling(model, data,
                       chains = 1),
              pars = 'log_lik')$ELPD)
}
optimal.lambda.loo <- optim(lambda, f.loo, method = 'L-BFGS-B', lower = 0.0001, upper = 1
optimal.lambda.PMSE <- optim(lambda, f.PMSE, method = 'L-BFGS-B', lower = 0.0001, upper =

data.loo <-list(N = nrow(x.train),
                K = ncol(x.train),
                x = x.train,
                y = y.train,
                lambda = optimal.lambda.loo$par,
                x_new = x.test,
                N_new = nrow(x.test))
data.PMSE <-list(N = nrow(x.train),
                 K = ncol(x.train),
                 x = x.train,
                 y = y.train,
                 lambda = optimal.lambda.PMSE$par,
```

```
                x_new = x.test,
                N_new = nrow(x.test))

fit.loo <- sampling(model, data.loo)
fit.PMSE <- sampling(model, data.PMSE)
```

A.6 Normal-Inverse-Gamma Prior Ready for LOO

```
    data {
    // 'NIGLOO.stan'
  int<lower=0> N;
  int<lower=0> K;
  matrix[N, K] x;
  vector[N] y;
  real<lower=0>lambda;
  int<lower=0> N_new; // prediction stuff
  matrix[N_new, K] x_new;
}
parameters {
  //real alpha;
  vector[K] beta;
  real<lower=0> sigma2;
  vector[N_new] y_new;

}
model {
 sigma2 ~ inv_gamma(0.5, 2);
 beta ~ normal(0, sigma2/lambda);
 y ~ normal( x*beta, sigma2);
 y_new ~ normal( x_new *beta, sigma2);
}

generated quantities {
  vector[N] log_lik;
  for (n in 1:N)
    log_lik[n] = normal_lpdf(y[n] |x[n,]*beta, sigma2);
}
```

A.7 Spike and Slab Stan

6.5.2

```
data {
  int<lower=0> N;
  int<lower=0> K;
  matrix[N, K] x;
  vector[N] y;
  //real<lower=0> lambda;
  int<lower=0> N_new; // prediction stuff
  matrix[N_new, K] x_new;
  real<lower=0> tau2; // variance of spike normal
}
parameters {
```

```
  //real alpha;
  vector[K] beta;
  real<lower=0,upper=1> theta; // our slab parameter
  real<lower=0> sigma2;
  vector[N_new] y_new;
  real<lower=0>lambda;

}
model {
 theta ~ beta(3,1);
 lambda ~ cauchy(0, 1);
 sigma2 ~ inv_gamma(0.5, 2);
 //beta ~ normal(0, sqrt(sigma2*z/lambda));
for (k in 1:K)
    target += log_mix(theta,
                      normal_lpdf(beta[k] | 0, sqrt(sigma2/lambda)),
                      normal_lpdf(beta[k] | 0, sqrt(tau2)));
  //y ~ normal( x*beta + alpha, sqrt(sigma2));
  y ~ normal( x*beta, sqrt(sigma2));
 // y_new ~ normal( x_new*beta + alpha, sqrt(sigma2));
 y_new ~ normal( x_new*beta, sqrt(sigma2));

}
generated quantities {
  vector[N] log_lik;
  for (n in 1:N)
    log_lik[n] = normal_lpdf(y[n] |x[n,]*beta, sqrt(sigma2));

}
```

A.8 Horseshoe Prior Stan

6.3.3

```
data {
  int<lower=0> N;
  int<lower=0> K;
  matrix[N, K] x;
  vector[N] y;

  //real<lower=0> lambda;
  //int<lower=0> N_new; // prediction stuff
  //matrix[N_new, K] x_new;
}
parameters {
  //real alpha;
  vector[K] beta;
  real<lower=0> sigma2;
 // vector[N_new] y_new;
  vector<lower=0>[K]lambda;
  real<lower=0> tau;
}
model {
```

```
 sigma2 ~ inv_gamma(0.5, 2);
 tau ~ cauchy(0, sigma2^(0.5));
 lambda ~ cauchy(0, tau);
 beta ~ normal(0, sqrt(lambda));
  //y ~ normal( x*beta + alpha, sqrt(sigma2));
  y ~ normal( x*beta, sqrt(sigma2));
 // y_new ~ normal( x_new*beta + alpha, sqrt(sigma2));
// y_new ~ normal( x_new*beta, sqrt(sigma2));
}
```

A.9 Strawderman-Berger Prior Stan

6.3.3

```
data {
  int<lower=0> N;
  int<lower=0> K;
  matrix[N, K] x;
  vector[N] y;

  //real<lower=0> lambda;
 // int<lower=0> N_new; // prediction stuff
 // matrix[N_new, K] x_new;
}
parameters {
  //real alpha;
  vector[K] beta;
  real<lower=0> sigma2;
//  vector[N_new] y_new;
  vector<lower=0, upper=1>[K]kappa;
}
model {
 sigma2 ~ inv_gamma(0.5, 2);
 kappa ~ beta(0.5,1);
 beta ~ normal(0, sqrt(kappa^(-1)-1));
  //y ~ normal( x*beta + alpha, sqrt(sigma2));
  y ~ normal( x*beta, sqrt(sigma2));
 // y_new ~ normal( x_new*beta + alpha, sqrt(sigma2));
// y_new ~ normal( x_new*beta, sqrt(sigma2));
}
```

A.10 Finnish Horseshoe Prior Stan

6.3.4

```
data{
  int<lower=0> N;
  int<lower=0> K;
  matrix[N, K] x;
  vector[N] y;
  // prediction data
  int<lower=0> N_new;
  matrix[N_new, K] x_new;
}
```

```
transformed data{
  // could be wise to move this into the input data section of the model
  // if you want to edit the values alot, as recompiling to change them is slow
  real m0 = 50; // prior belief of number of significant beta_j
  int nu = 1; // degrees of student-t dist obtained when marginalising c2, 1==cauchy
  real s2 =  56.25; // impacts tail size of c2 inverse gamma. large implies less certain
}
parameters{
  vector[K] beta;
  real<lower=0> tau;
  vector<lower=0>[K] lambda;
  real<lower=0> c2;
  real<lower=0> sigma2;
  vector[N_new] y_new;
}
// avoid using fractions as it can upset stan
transformed parameters{
  vector<lower=0>[K] lambda_tilde;
  {
    for (j in 1:K)
    {
     lambda_tilde[j] = sqrt((c2*square(lambda[j]))*(c2+square(tau*lambda[j]))^(-1));
    }
  }
}
model{
  sigma2 ~ inv_gamma(0.5, 2);
  c2 ~ inv_gamma(nu*0.5, nu*0.5*s2);
  lambda ~ cauchy(0, 1);
  tau ~ cauchy(0, m0*(K-m0)^(-1)*sqrt(sigma2)*sqrt(N)^(-1)); // shape = t0
  beta ~ normal(0, tau*lambda_tilde);
  y ~ normal(x*beta, sqrt(sigma2));
  y_new ~ normal( x_new*beta, sqrt(sigma2));
}
```

A.11 Hierarchical NIG Prior Stan

Hierarchical Comparison

```
data {
  int<lower=0> N;
  int<lower=0> K;
  matrix[N, K] x;
  vector[N] y;
  int<lower=0>nu;

  //real<lower=0> lambda;
  int<lower=0> N_new; // prediction stuff
  matrix[N_new, K] x_new;
}
parameters {
  //real alpha;
  vector[K] beta;
```

```
  vector<lower=0>[K] tau2;
  real<lower=0> sigma2;
  vector[N_new] y_new;
  real<lower=0>lambda;

}
model {
 lambda ~ cauchy(0, 1);
 sigma2 ~ inv_gamma(0.5, 2);
 for(j in 1:K){
 tau2[j] ~ inv_gamma(nu*0.5, nu*0.5*lambda^-1);
 beta[j] ~ normal(0, sqrt(sigma2*tau2[j]));
 }
 // tau2 ~ inv_gamma(nu*0.5, nu*0.5*lambda^-1);
 // beta ~ normal(0, sigma2*tau2);
 //
  //y ~ normal( x*beta + alpha, sigma2);
  y ~ normal( x*beta, sqrt(sigma2));
 // y_new ~ normal( x_new*beta + alpha, sigma2);
 y_new ~ normal( x_new*beta, sqrt(sigma2));

}
generated quantities {
  // generate values of the log-likelihood for the purpose of using LOO.
  vector[N] log_lik;
  for (n in 1:N)
    log_lik[n] = normal_lpdf(y[n] |x[n,]*beta, sqrt(sigma2));
}
```

A.12 Variable Selection Functions

Variable Selection Functions

```
library(rstan)
## we wish to make a function which takes a given quantile and checks whether
## each coefficient has 0 within that quantile, if it does we can
## compare our new idea of beta with the true beta to assess variable
## selection accuracy.
coeff.selector <- function(beta.samps, EQTpercent){
  ## beta.samps = extract(modelSamps, pars = 'beta')£beta
  ## EQTpercent = size of eqt credible interval we want to select variables
  ##              based upon, 95% is large, 50% is quite reasonable
  if (EQTpercent>1){
    ## if EQTpercentage is above 1, make it a percentage by dividing
    ## by a hundred (error proofing)
    EQTpercent <- EQTpercent/100
  }
  ## a vector with beta.included_i = 1 implying beta_i is included
  ## and           beta.included_i = 0 implying beta_i is not included
  beta.included <- rep(1, ncol(beta.samps))
  for(j in 1:ncol(beta.samps)){
    ## for each beta_j we estimate the desired percentile eqt from
    ## our beta_j samples.
```

```
    percentiles <- quantile(beta.samps[,j],
                            probs = c((1-EQTpercent)/2,(1+EQTpercent)/2))
    if(percentiles[1]<=0 & 0<=percentiles[2]){
      ## check whether zero lies within the percentile
      ## and indicate that it shouldn't be included if it does
      beta.included[j] <- 0
    }
  }
  return(beta.included)
}
## now we want a function which compares our selected coefficient vector
## returned by coeff.selector with the true beta coefficients
## true beta coefficients are stored in 'beta'
select.comparison <- function(beta.included, beta){
  ## correct inclusion rate = sum(both!=0)/sum(beta==1)
  ## false inclusion rate = sum(beta.inc==1 & beta==0)/sum(beta==0).
  ## we catch potential singularities using these if statements
  if(sum(beta==0)==0){
    return(list(correct.inclu.rate =
                  sum((beta.included!=0)&(beta!=0))/sum(beta!=0),
                false.inclu.rate =
                  sum((beta.included==1)&(beta==0))/1))}
    if(sum(beta!=0)==0){
      return(list(correct.inclu.rate =
                    sum((beta.included!=0)&(beta!=0))/1,
                  false.inclu.rate =
                    sum((beta.included==1)&(beta==0))/sum(beta==0)))
  }
  return(list(correct.inclu.rate =
                sum((beta.included!=0)&(beta!=0))/sum(beta!=0),
              false.inclu.rate =
                sum((beta.included==1)&(beta==0))/sum(beta==0)))
}

## and finally, we want a function which uses the above two to iteratively
## test different EQT percentiles, to minimise the distance criterion
## Distance Criterion = sqrt((1-correct.inc.rate)^2-false.inc.rate^2)
distance.criterion <- function(beta.samps, beta, EQTpercent){
  return(sqrt((1-select.comparison(coeff.selecter(beta.samps,EQTpercent),
                                   beta)$correct.inclu.rate)^2 +
                select.comparison(coeff.selecter(beta.samps,EQTpercent),
                                  beta)$false.inclu.rate^2))
}

optimal.EQT <- function(beta.samps, beta){
  ## want to check 50,55,60,65,70,75,80,85,90,95 EQTpercentiles
  # first value of distance is the current distance associated with
  # the EQTpercent, and second value is the EQTpercent which gave
  # rise to that distance
  distance <- c(1,0)
  for(i in 1:20){
    EQTpercent <- (5*i)/100
    current.distance <- distance.criterion(beta.samps,beta,EQTpercent)
```

```
    if(current.distance<distance[1]){
      distance <- c(current.distance, EQTpercent)
    }
  }
  return(distance[2])} ## returns optimal EQT percentage

MCC.calc <- function(optimalSelection, beta){
  # optimal selection is the selection of variables leading to
  # lowest possible distance criterion
  TN <- sum((optimalSelection==0)&(beta==0))   ## true zeroes
  TP <- sum((optimalSelection!=0)&(beta!=0))   ## true significant coefficients
  FN <- sum((optimalSelection==0)&(beta!=0))   ## falsely insignificant
  FP <- sum((optimalSelection!=0)&(beta==0))   ## falsely significant
  return((TN*TP-FN*FP)/sqrt((TP+FP)*(TP+FN)*(TN+FP)*(TN+FN)))
}

## example of use
data <- list(N=nrow(x.train),
             K=ncol(x.train),
             x=x.train,
             y=y.train,
             x_new=x.test,
             N_new=nrow(x.test),
             tau2 = 0.01,
             m0 = 5)
spikeslabModel <- stan_model('Spiked+Slabbed.stan') ## A stan model
spikeSamps <- sampling(spikeslabModel, data)
spikeBetaSamps <- extract(spikeSamps, pars = 'beta')$'beta'
optim.Selection <- coeff.selector(spikeBetaSamps,
                                  optimal.EQT(spikeBetaSamps, beta))
## MCC varies between -1 and +1, +1 is perfect selection
##                             , -1 is worst selection
MCC.calc(optim.Selection, beta)
```

A.13 Simulation Master (functions)

Back to text

```
      PMSEcalc <- function(predict, y.test){
  return(mean((predict-y.test)^2))}
gen_x.norm <- function(N, total.p,
                       mu, varmat){
  return(mvrnorm(N, mu, varmat))
}
gen_y.data <- function(beta, x, sd){
  return(x%*%beta + rnorm(nrow(x), 0, sd))
}
instant_data <- function(Ntrain, Ntest, total.p, mu, varmat, beta, y.sd){
  x <- gen_x.norm(Ntrain+Ntest, total.p, mu, varmat)
  y <- gen_y.data(beta, x, y.sd)
  ## for some low observation data we may want to increase sample size N,
  ## but decrease training.rows, so we still have a fair amount of testing
  ## data
```

```
  training.rows <- sample(1:(Ntrain+Ntest), floor(Ntrain))

  x.train <- x[training.rows, ]
  y.train <- y[training.rows]

  x.test <- x[-training.rows, ]
  y.test <- y[-training.rows]
  return(list(x.train = x.train,
              y.train = y.train,
              x.test = x.test,
              y.test = y.test))
}
data.list <- function(x.train, x.test, y.train, y.test, tau2 = 0.01,
                      m0 = floor(ncol(x.train)/2)){
  return(list(N = nrow(x.train),
              K = ncol(x.train),
              x = x.train,
              y = y.train,
              x_new = x.test,
              N_new = nrow(x.test),
              tau2 = tau2,
              m0 = m0,
              y.test = y.test))
}

##BAYESIAN:
bayesSim <- function(model, data, y.test, beta, i = 1){
  # sampling + time taken to sample
  tic('sleeping')
  samps <- sampling(model, data, seed = pi * i,
                    show_message = FALSE, verbose = FALSE, refresh = 0)
  time <- toc(log = TRUE)
  time.taken <- as.numeric(time$toc-time$tic)
  # predictive performance
  PMSE <- PMSEcalc(colMeans(extract(samps, pars = 'y_new')$y_new), y.test)
  ELPD <- loo(samps)$estimates[1,1]
  sigma2 <- mean(extract(samps, pars = 'sigma2')$'sigma2')
  # variable selection
  beta.samps <- extract(samps, pars = 'beta')$'beta'
  optEQT <- optimal.EQT(beta.samps, beta)
  opt.selection <- coeff.selector(beta.samps, optEQT)
  correct.inc.rate <- select.comparison(opt.selection,
                                        beta)$correct.inclu.rate
  false.inc.rate <- select.comparison(opt.selection,
                                      beta)$false.inclu.rate
  MCC <- MCC.calc(opt.selection, beta)
  return(list(PMSE = PMSE,
              ELPD = ELPD,
              MCC = MCC,
              correct.inc.rate = correct.inc.rate,
              false.inc.rate = false.inc.rate,
              OptEQT = optEQT,
              sigma2 = sigma2,
```

```
              time = time.taken))
}
#example use
## data <- data.list(x.train, x.test, y.train, y.test)
## bRidgeObs1 <- bayesSim(bRidge, data, y.test, beta)
## results£BaysRidge£PMSE[1] <- bRidgeObs1£PMSE

## bayesian list updater function:
b.results.update <- function(results, j, obs, i ){
  # j varies through the different models,
  # j = 1 implies bRidge, j = 2 implies bLasso, ... ,
  # j = 7 implies bUninformative
  results[[j]]$PMSE[i] <- obs$PMSE
  results[[j]]$ELPD[i] <- obs$ELPD
  results[[j]]$correct.inc.rate[i] <- obs$correct.inc.rate
  results[[j]]$false.inc.rate[i] <- obs$false.inc.rate
  results[[j]]$OptEQT[i] <- obs$OptEQT
  results[[j]]$MCC[i] <- obs$MCC
  results[[j]]$sigma2[i] <- obs$sigma2
  results[[j]]$time[i] <- obs$time
  return(results)
}

## FREQUENTIST:
freqSim <- function(alpha.sel, data.gen, beta, i = 1){
  set.seed( pi * i)
  ## ridge regression:
  tic('sleeping')
  # fReg == frequentist regression method
  fReg <- cv.glmnet(data.gen$x.train, data.gen$y.train, alpha = alpha.sel,
                    type = 'mse', intercept = FALSE)
  time <- toc(log = TRUE)
  time.taken <- as.numeric(time$toc-time$tic)
  beta.hat <- coef(fReg)[-1]
  PMSE <-PMSEcalc(predict(fReg, newx = data.gen$x.test),data.gen$y.test)
  opt.selection <- as.numeric(beta.hat!=0)
  correct.inc.rate <- select.comparison(opt.selection,
                                        beta)$correct.inclu.rate
  false.inc.rate <- select.comparison(opt.selection,
                                      beta)$false.inclu.rate
  MCC <- MCC.calc(opt.selection, beta)
  return(list(PMSE = PMSE,
              MCC = MCC,
              correct.inc.rate = correct.inc.rate,
              false.inc.rate = false.inc.rate,
              time = time.taken))

}
f.results.update <- function(results, j, obs, i ){
  # j varies through the different models,
  # j = 8 implies fRidge, j = 9 implies fLasso,
  # j = 10 implies fElasticNetAlpha=0.5
  results[[j]]$PMSE[i] <- obs$PMSE
```

```
  results[[j]]$correct.inc.rate[i] <- obs$correct.inc.rate
  results[[j]]$false.inc.rate[i] <- obs$false.inc.rate
  results[[j]]$false.inc.rate[i] <- obs$false.inc.rate
  results[[j]]$MCC[i] <- obs$MCC
  results[[j]]$time[i] <- obs$time
  return(results)
}
coeff.selector <- function(beta.samps, EQTpercent){
  ## beta.samps = extract(modelSamps, pars = 'beta')£beta
  ## EQTpercent = size of eqt credible interval we want to select variables
  ##              based upon, 95% is large, 50% is quite reasonable
  if (EQTpercent>1){
    ## if EQTpercentage is above 1, make it a percentage by dividing
    ## by a hundred (error proofing)
    EQTpercent <- EQTpercent/100
  }
  ## a vector with beta.included_i = 1 implying beta_i is included
  ## and           beta.included_i = 0 implying beta_i is not included
  beta.included <- rep(1, ncol(beta.samps))
  for(j in 1:ncol(beta.samps)){
    ## for each beta_j we estimate the desired percentile eqt from
    ## our beta_j samples.
    percentiles <- quantile(beta.samps[,j],
                            probs = c((1-EQTpercent)/2,(1+EQTpercent)/2))
    if(percentiles[1]<=0 & 0<=percentiles[2]){
      ## check whether zero lies within the percentile
      ## and indicate that it shouldn't be included if it does
      beta.included[j] <- 0
    }
  }
  return(beta.included)
}
OLSsim <- function(data.gen){
  tic('sleeping')
  fit.lm <- lm(data.gen$y.train ~ data.gen$x.train+0)
  time <- toc(log = TRUE)
  time.taken <- as.numeric(time$toc-time$tic)
  beta.hat <- coef(fit.lm)
  PMSE <-PMSEcalc(data.gen$x.test%*%beta.hat,data.gen$y.test)
  opt.selection <- as.numeric(beta.hat!=0)
  correct.inc.rate <- select.comparison(opt.selection,
                                        beta)$correct.inclu.rate
  false.inc.rate <- select.comparison(opt.selection,
                                      beta)$false.inclu.rate
  MCC <- MCC.calc(opt.selection, beta)
  return(list(PMSE = PMSE,
              MCC = MCC,
              correct.inc.rate = correct.inc.rate,
              false.inc.rate = false.inc.rate,
              time = time.taken))
}

## VARIABLE SELECTION
```

```
select.comparison <- function(beta.included, beta){
  ## correct inclusion rate = sum(both!=0)/sum(beta==1)
  ## false inclusion rate = sum(beta.inc==1 & beta==0)/sum(beta==0).
  ## we catch potential singularities using these if statements
  if(sum(beta==0)==0){
    return(list(correct.inclu.rate =
                  sum((beta.included!=0)&(beta!=0))/sum(beta!=0),
                false.inclu.rate =
                  sum((beta.included==1)&(beta==0))/1))}
    if(sum(beta!=0)==0){
      return(list(correct.inclu.rate =
                    sum((beta.included!=0)&(beta!=0))/1,
                  false.inclu.rate =
                    sum((beta.included==1)&(beta==0))/sum(beta==0)))
  }
  return(list(correct.inclu.rate =
                sum((beta.included!=0)&(beta!=0))/sum(beta!=0),
              false.inclu.rate =
                sum((beta.included==1)&(beta==0))/sum(beta==0)))
}


distance.criterion <- function(beta.samps, beta, EQTpercent){
  return(sqrt((1-select.comparison(coeff.selector(beta.samps,EQTpercent),
                                   beta)$correct.inclu.rate)^2 +
                select.comparison(coeff.selector(beta.samps,EQTpercent),
                                  beta)$false.inclu.rate^2))
}

optimal.EQT <- function(beta.samps, beta){
  ## want to check 50,55,60,65,70,75,80,85,90,95 EQTpercentiles
  # first value of distance is the current distance associated with
  # the EQTpercent, and second value is the EQTpercent which gave
  # rise to that distance
  distance <- c(1,0)
  for(i in 1:20){
    EQTpercent <- (5*i)/100
    current.distance <- distance.criterion(beta.samps,beta,EQTpercent)
    if(current.distance<distance[1]){
      distance <- c(current.distance, EQTpercent)
    }
  }
  return(distance[2])} ## returns optimal EQT percentage

MCC.calc <- function(optimalSelection, beta){
  # optimal selection is the selection of variables leading to
  # lowest possible distance criterion
  TN <- sum((optimalSelection==0)&(beta==0))   ## true zeroes
  TP <- sum((optimalSelection!=0)&(beta!=0))   ## true significant coefficients
  FN <- sum((optimalSelection==0)&(beta!=0))   ## falsely insignificant
  FP <- sum((optimalSelection!=0)&(beta==0))   ## falsely significant
  return((TN*TP-FN*FP)/sqrt((TP+FP)*(TP+FN)*(TN+FP)*(TN+FN)))
}
```

A.14 Simulation Study (Running)

Study

```
    ### SIMULATION STUDY
# Remember to set seeds for reproducibility during iterations
## to be used with 'variableSelectionDistanceFunctions.R'
## data conditions generated by 'data-gen.R'
## simulations generated by 'SimGenerator.R'
source('simulationMaster.R')  ## stores all functions we need
# compile the Bayesian Models we will use
library(MASS)
library(rstan)
library(glmnet) # for fitting penalised frequentist models
library(tictoc) # for measuring run time
total.sims <- 20
results <- list('BaysRidge'= list(PMSE = rep(0,total.sims),
                                  ELPD = rep(0,total.sims),
                                  MCC = rep(0,total.sims),
                                  correct.inc.rate = rep(0,total.sims),
                                  false.inc.rate = rep(0,total.sims),
                                  OptEQT = rep(0,total.sims),
                                  sigma2 = rep(0,total.sims),
                                  time = rep(0,total.sims)),
                        'BaysLasso' = list(PMSE = rep(0,total.sims),
                                   ELPD = rep(0,total.sims),
                                   MCC = rep(0,total.sims),
                                   correct.inc.rate = rep(0,total.sims),
                                   false.inc.rate = rep(0,total.sims),
                                   OptEQT = rep(0,total.sims),
                                   sigma2 = rep(0,total.sims),
                                   time = rep(0,total.sims)),
                        'BaysNet' = list(PMSE = rep(0,total.sims),
                                     ELPD = rep(0,total.sims),
                                     MCC = rep(0,total.sims),
                                     correct.inc.rate = rep(0,total.sims),
                                     false.inc.rate = rep(0,total.sims),
                                     OptEQT = rep(0,total.sims),
                                     sigma2 = rep(0,total.sims),
                                     time = rep(0,total.sims)),
                        'SpikeSlab' = list(PMSE = rep(0,total.sims),
                                     ELPD = rep(0,total.sims),
                                     MCC = rep(0,total.sims),
                                     correct.inc.rate = rep(0,total.sims),
                                     false.inc.rate = rep(0,total.sims),
                                     OptEQT = rep(0,total.sims),
                                     sigma2 = rep(0,total.sims),
                                     time = rep(0,total.sims)),
                        'Horseshoe' = list(PMSE = rep(0,total.sims),
                                       ELPD = rep(0,total.sims),
                                       MCC = rep(0,total.sims),
                                       correct.inc.rate = rep(0,total.sims),
                                       false.inc.rate = rep(0,total.sims),
                                       OptEQT = rep(0,total.sims),
```

```
                                        sigma2 = rep(0,total.sims),
                                        time = rep(0,total.sims)),
                          'Finnish Horseshoe' = list(PMSE = rep(0,total.sims),
                                                     ELPD = rep(0,total.sims),
                                                     MCC = rep(0,total.sims),
                                                     correct.inc.rate = rep(0,total.sims),
                                                     false.inc.rate = rep(0,total.sims),
                                                     OptEQT = rep(0,total.sims),
                                                     sigma2 = rep(0,total.sims),
                                                     time = rep(0,total.sims)),
                          'Uninformative' = list(PMSE = rep(0,total.sims),
                                                 ELPD = rep(0,total.sims),
                                                 MCC = rep(0,total.sims),
                                                 correct.inc.rate = rep(0,total.sims),
                                                 false.inc.rate = rep(0,total.sims),
                                                 OptEQT = rep(0,total.sims),
                                                 sigma2 = rep(0,total.sims),
                                                 time = rep(0,total.sims)),
                          'FreqRidge' = list(PMSE = rep(0,total.sims),
                                             MCC = rep(0,total.sims),
                                             correct.inc.rate = rep(0,total.sims),
                                             false.inc.rate = rep(0,total.sims),
                                             time = rep(0,total.sims)),
                          'FreqLasso' = list(PMSE = rep(0,total.sims),
                                             MCC = rep(0,total.sims),
                                             correct.inc.rate = rep(0,total.sims),
                                             false.inc.rate = rep(0,total.sims),
                                             time = rep(0,total.sims)),
                            'FreqENalpha0.5' = list(PMSE = rep(0,total.sims),
                                                    MCC = rep(0,total.sims),
                                                    correct.inc.rate = rep(0,total.sims),
                                                    false.inc.rate = rep(0,total.sims),
                                                    time = rep(0,total.sims)),
                                   'OLS' = list(PMSE = rep(0,total.sims),
                                                MCC = rep(0,total.sims),
                                                correct.inc.rate = rep(0,total.sims),
                                                false.inc.rate = rep(0,total.sims),
                                                time = rep(0,total.sims)))
## COMPILE BAYESIAN MODELS
# 'b' prefix implies Bayesian

bRidge <- stan_model('NIGpredict.stan')
bLasso <- stan_model('lassoModel.stan')
bElasticNet <- stan_model('ElasticNet.stan')
bSpikeSlab <- stan_model('Spiked+Slabbed.stan') ## requires tau2 spec in data
bHorseshoe <- stan_model('HorseShoe.stan')
bFinnishHorseshoe <- stan_model('MyfinnishHorse.stan')  ## requires m0 spec in data
bUninformative <- stan_model('WeaklyInformativeLM.stan')
for(i in 1:total.sims){
  set.seed(pi * i)
  data.gen <- instant_data(N.train, N.test , total.p, mu, varmat, beta, y.sd)
  ## (data == the data to be fed into stan models)
  data <- data.list(data.gen$x.train, data.gen$x.test, data.gen$y.train, data.gen$y.test,
```

```
                    m0 = max(sum(beta==0),1))
  ## unfortunately rstan does not allow lists of models, so we
  ## must manually run through each model sampler to obtain results
  ## b.results.update function stored in "SimGenerator.R"
  ##### BAYESIAN MODEL UPDATE ###################################
  results <- b.results.update(results, 1,
                              bayesSim(bRidge, data, data.gen$y.test
                                       , beta, i), i)
  print(paste(i - 6/7, "iterations complete"))
  results <- b.results.update(results, 2,
                              bayesSim(bLasso, data, data.gen$y.test
                                       , beta, i), i)
  print(paste(i - 5/7, "iterations complete"))
  results <- b.results.update(results, 3,
                              bayesSim(bElasticNet, data, data.gen$y.test
                                       , beta, i), i)
  print(paste(i - 4/7, "iterations complete"))
  results <- b.results.update(results, 4,
                              bayesSim(bSpikeSlab, data, data.gen$y.test
                                       , beta, i), i)
  print(paste(i - 3/7, "iterations complete"))
  results <- b.results.update(results, 5,
                              bayesSim(bHorseshoe, data, data.gen$y.test
                                       , beta, i), i)
  print(paste(i - 2/7, "iterations complete"))
  results <- b.results.update(results, 6,
                              bayesSim(bFinnishHorseshoe, data, data.gen$y.test
                                       , beta, i), i)
  print(paste(i - 1/7, "iterations complete"))
  results <- b.results.update(results, 7,
                              bayesSim(bUninformative, data, data.gen$y.test
                                       , beta, i), i)
  print("iteration = ", i)

  ####### END OF BAYESIAN MODEL UPDATE ###################################
  ####### FREQUENTIST MODEL UPDATE     ###################################
  results <- f.results.update(results, 8,
                              freqSim(0, data.gen, beta, i),
                              i)
  results <- f.results.update(results, 9,
                              freqSim(1, data.gen, beta, i),
                              i)
  results <- f.results.update(results, 10,
                              freqSim(0.5, data.gen, beta, i),
                              i)
  results <- f.results.update(results, 11,
                              OLSsim(data.gen), i)
  ####### END OF FREQUENTIST MODEL UPDATE ################################
}
## permanently save the results:
saveRDS(results, file="ResultsCase6.RData")
## read the results:
results6 <- readRDS("ResultsCase6.RData")
```

```
View(results)
## Different Cases:
########CASE1############
## Easy Regression ##
N.train <- 300
N.test <- 300
beta <- c(-5, 5)
y.sd <- 4
## \boldsymbol{X}_ij ~ N(0,1)
mu <- c(0, 0)
varmat <- matrix(c(1,0,0,1), nrow = 2)

########CASE2#########
## Very Significant Predictors and Low Sample Size
N.train <- 50
N.test <- 100
beta <- c(seq(-20, -10, length.out = 25), seq(10, 20, length.out = 25))
y.sd <- 4
## \boldsymbol{X}_ij ~ N(0,1)
mu <- rep(0, length(beta))
varmat <- matrix(rep(0, length(beta)^2), nrow = length(beta))
diag(varmat) <- 1

########CASE3########
## Sparse High Sample Size
N.train <- 200
N.test <- 200
beta <- c(seq(5, 10, length.out = 10), seq(-5, -10, length.out = 10),
          rep(0, 30))
y.sd <- 3
## \boldsymbol{X}_ij ~ N(0, 1)
mu <- rep(0, length(beta))
varmat <- matrix(rep(0, length(beta)^2), nrow = length(beta))
diag(varmat) <- 1

#######CASE4########
## Sparse Low Sample Size
N.train <- 30
N.test <- 200
beta <- c(seq(5, 10, length.out = 10), seq(-5, -10, length.out = 10),
          rep(0, 30))
y.sd <- 3
## \boldsymbol{X}_ij ~ N(0, 1)
mu <- rep(0, length(beta))
varmat <- matrix(rep(0, length(beta)^2), nrow = length(beta))
diag(varmat) <- 1

#######CASE5#####
## Sparse Low Sample Size and High Correlation in data
N.train <- 30
N.test <- 200
beta <- c(seq(10, 15, length.out = 10), seq(-10, -15, length.out = 10),rep(0, 30))
y.sd <- 1
```

```
mu <- rep(10, length(beta))
varmat <- matrix(rep(0, length(beta)^2), nrow = length(beta))
### insignifcant predictors cant have a covariance with significant predictors
## 1:5 covariance of 5 with each other
## 6:15 covariance of 8 with each other
## 16:20 covariance of 9 with each other
## All data has a variance of 9
varmat[1:5, 1:5] <- 5
varmat[6:15, 6:15] <- 8
varmat[16:20, 16:20] <- 9
diag(varmat) <- 9

#######CASE6#######
## Non-sparse Collinear data
N.train <- 30
N.test <- 200
beta <- c(seq(10, 15, length.out = 10), seq(-10, -15, length.out = 10))
y.sd <- 1
mu <- rep(10, length(beta))
varmat <- matrix(rep(0, length(beta)^2), nrow = length(beta))
## 1:5 covariance of 5 with each other
## 6:15 covariance of 8 with each other
## 16:20 covariance of 9 with each other
## All data has a variance of 9
varmat[1:5, 1:5] <- 5
varmat[6:15, 6:15] <- 8
varmat[16:20, 16:20] <- 9
diag(varmat) <- 9
```

A.15 Figures

Click to go back to text

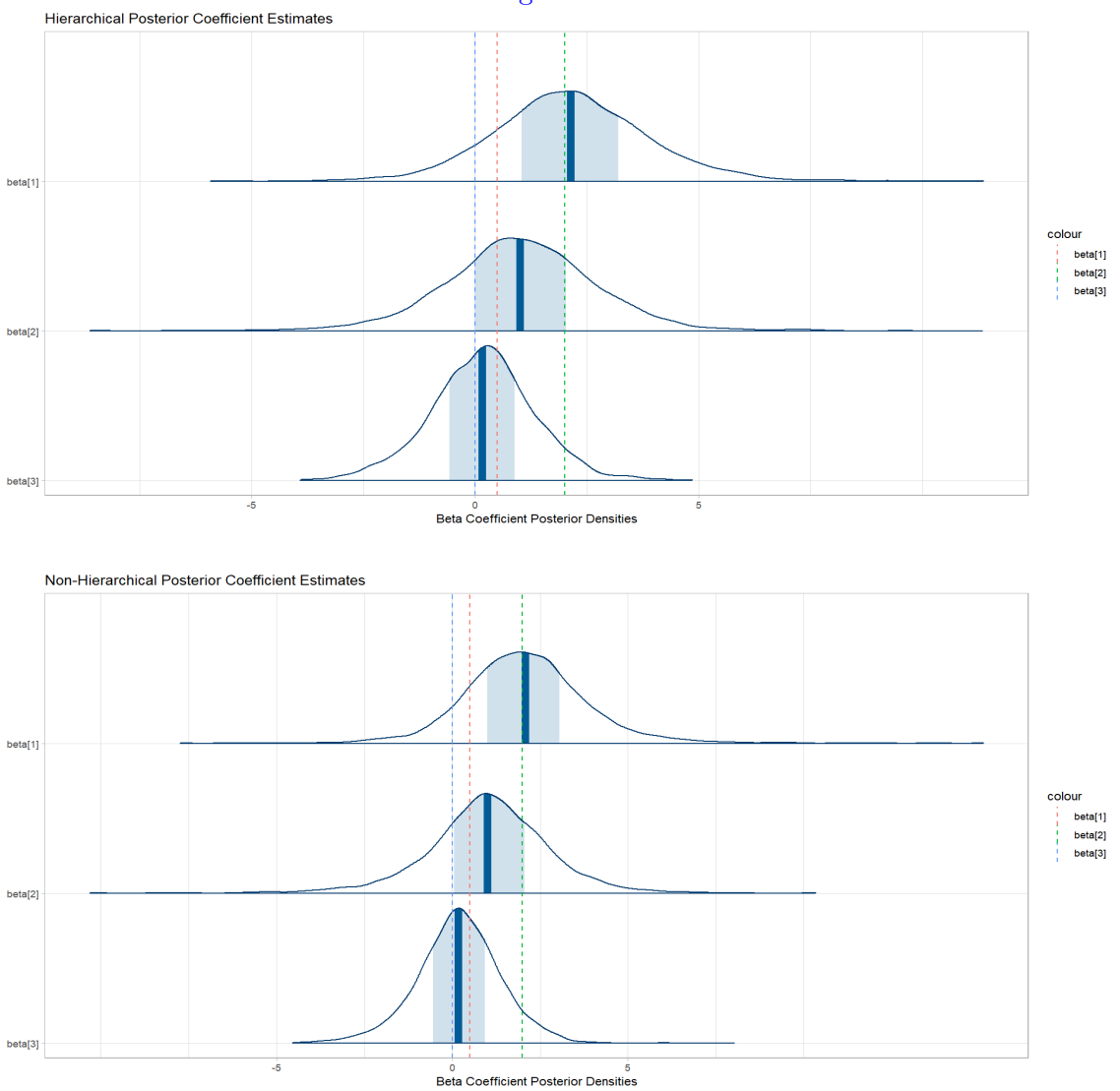

Figure A.1: Plots of posterior coefficient densities. The hierarchical model relates to the student-t prior and the non-hierarchical model relates to the ridge prior.

A.16 Example 5.2.2.

Grouped data sims

```
## GROUPED DATA STUDY 5.2.2.
library(glmnet)
library(rstan)
## x has 50 columns
## 200 training rows
## 100 testing rows
n <- 100
x <- matrix(rep(0,n*50), nrow = n)
set.seed(42*pi)
```

```
for(i in 1:n){
  z1 <- rnorm(1)
  z2 <- rnorm(1)
  z3 <- rnorm(1)
  z4 <- rnorm(1)
  for(j in 1:10){
    x[i,j] <- z1 + rnorm(1, 0, 0.1)
  }
  for(j in 11:20){
    x[i,j] <- z2 + rnorm(1,0 , 0.1)
  }
  for(j in 21:30){
    x[i,j] <- z3 + rnorm(1, 0, 0.1)
  }
  for(j in 31:40){
    x[i,j] <- z4 + rnorm(1, 0, 0.1)
  }
  for(j in 41:50){
    x[i,j] <- rnorm(1)
  }
}
beta <- c(rep(10, 20), c(rep(-10, 20)), rep(0, 10))
epsilon <- rnorm(n, 0, 10)
y <- x%*%beta +epsilon

NIGpredict <- stan_model('NIGpredict.stan')
Blasso <- stan_model('lassoModel.stan')
Benet <- stan_model('ElasticNet.stan')

NIGsamps <- sampling(NIGpredict, data = list(N = nrow(x),
                                             K = ncol(x),
                                             x = x,
                                             y = as.vector(y)))
BlassoSamps <- sampling(Blasso, data = list(N = nrow(x),
                                             K = ncol(x),
                                             x = x,
                                             y = as.vector(y)))
BenetSamps <- sampling(Benet, data = list(N = nrow(x),
                                             K = ncol(x),
                                             x = x,
                                             y = as.vector(y)))

fLasso <- cv.glmnet(x, y, alpha = 1, type.measure = 'mse')
fRidge <- cv.glmnet(x, y, alpha = 0, type.measure = 'mse')
fEnet <- cv.glmnet(x, y, alpha = 0.5, type.measure ='mse')
c.fLasso <- coef(fLasso)
c.fRidge <- coef(fRidge)
c.fEnet <- coef(fEnet)
c.NIG <- colMeans(extract(NIGsamps, pars = 'beta')$'beta')
c.Blasso <- colMeans(extract(BlassoSamps, pars = 'beta')$'beta')
c.Benet <- colMeans(extract(BenetSamps, pars = 'beta')$'beta')

mean(abs(c.NIG- beta))
```

```
mean(abs(c.Blasso - beta))
mean(abs(c.Benet - beta))
mean(abs(c.fRidge[-1] - beta))
mean(abs(c.fLasso[-1] - beta))
mean(abs(c.fEnet[-1] - beta))
```

A.17 Bayesian Lasso Stan

```
    data {
  int<lower=0> N;
  int<lower=0> K;
  matrix[N, K] x;
  vector[N] y;

  //int<lower=0> N_new; // prediction stuff
  //matrix[N_new, K] x_new;
}
parameters {
  real alpha;
  vector[K] beta;
  real<lower=0> sigma2;
  //vector[N_new] y_new;
  real<lower=0>lambda;

}
model {
  lambda ~ cauchy(0, 1);
  sigma2 ~ inv_gamma(0.5, 2);
  beta ~ double_exponential(0, sqrt(sigma2)/lambda);
  y ~ normal( x*beta + alpha, sqrt(sigma2));
  //y_new ~ normal( x_new*beta + alpha, sigma2);

}
generated quantities {
  vector[N] log_lik;
  for (n in 1:N)
    log_lik[n] = normal_lpdf(y[n] |x[n,]*beta, sqrt(sigma2));
}
```

A.18 Elastic Net Stan

```
    data {
  int<lower=0> N;
  int<lower=0> K;
  matrix[N, K] x;
  vector[N] y;
  //int<lower=0> N_new; // prediction stuff
  //matrix[N_new, K] x_new;
}
parameters {
  real alpha;
  vector[K] beta;
  real<lower=0> sigma2;
```

```
  //vector[N_new] y_new;
  real<lower=0>lambda1;
  real<lower=0>lambda2;
  vector<lower=1>[K]tau;

}
model {
  lambda1 ~ cauchy(0, 1);
  lambda2 ~ cauchy(0, 1);
  target += -log(sigma2);
  for (j in 1:K){
    tau[j] ~ gamma(0.5, 8*lambda2*sigma2*lambda1^(-2)) T[1,];
    beta[j] ~ normal(0, sqrt(sigma2*(tau[j]-1)*(tau[j]*lambda2)^(-1)));
    }

  y ~ normal( x*beta + alpha, sqrt(sigma2));
//  y_new ~ normal( x_new*beta, sqrt(sigma2));

}
generated quantities {
  vector[N] log_lik;
  for (n in 1:N)
    log_lik[n] = normal_lpdf(y[n] |x[n,]*beta, sqrt(sigma2));
}
```

A.19 Burn (custom) Prior Stan

```
    data {
  int<lower=0> N;
  int<lower=0> K;
  matrix[N, K] x;
  vector[N] y;

  //real<lower=0> lambda;
  int<lower=0> N_new; // prediction stuff
  matrix[N_new, K] x_new;

  // special global scale parameters
  int<lower=0, upper=(K-1)> m0;
}

parameters {
  //real alpha;
  vector[K] beta;
  real<lower=0> sigma2;
  vector[N_new] y_new;
  vector<lower=0, upper = 1>[K] kappa;
  real<lower=0> tau;
}
transformed parameters{
  vector<lower = 0>[K] lambda;{
    for (j in 1:K){
```

```
    lambda[j] = sqrt((1-kappa[j])*kappa[j]^(-1));
  }
}
}
model {
  // custom "altBurn" prior for lambda
  for (j in 1:K){
 target += log(-(1-kappa[j])*log(kappa[j]));
  };
 sigma2 ~ inv_gamma(0.5, 2);
 tau ~ cauchy(0, m0*(K-m0)^(-1)*sqrt(sigma2)*N^(-0.5));
 beta ~ double_exponential(0, tau * lambda);
  //y ~ normal( x*beta + alpha, sigma2);
  y ~ normal( x*beta, sqrt(sigma2));
 // y_new ~ normal( x_new*beta + alpha, sigma2);
 y_new ~ normal( x_new*beta, sqrt(sigma2));

}

generated quantities {
  vector[N] log_lik;
  for (n in 1:N)
    log_lik[n] = normal_lpdf(y[n] |x[n,]*beta, sqrt(sigma2));
}
```