University of
St Andrews

School of Mathematics and Statistics
MT5099 Dissertation for M.Sc. Programme/s

Variable Selection in High-Dimensional Data within the Bayesian Framework

Author:
ERNA KUGINYTĖ

15th August 2023

Abstract

This study examines challenges inherent in variable selection for linear models within the Bayesian framework, particularly in high-dimensional settings using R packages. 'High-dimensionality' refers to contexts involving many predictors that are fewer, equal to, or greater than the number of data points. Frequentist penalised regression Lasso and elastic-net methods and the machine learning XGBoost method are initially used to establish a benchmark. Subsequent Bayesian approaches encompass Bayesian Lasso, spike-and-slab prior, spike-and-slab Lasso, horseshoe priors, and the simplified shotgun stochastic search with screening. Packages that share methodologies are juxtaposed for user-friendliness and implementation nuances. Simulated data scenarios reveal that Bayesian methods, which compute coefficient credible intervals, often outperform frequentist counterparts by adeptly integrating model uncertainty into feature selection. However, they may miss weak signals when predictors significantly outnumber data points. Methods based on point estimates tend to retain irrelevant variables, while the machine learning method underperforms in all settings. Analysing socio-economic crime data largely aligns with the synthetic data study outcomes.

Acknowledgements

I am profoundly grateful to my advisor, Dr Michail Papathomas, whose guidance was instrumental throughout my dissertation journey. His insights and constructive feedback on my draft greatly contributed to my work. My sincere appreciation goes to the Computer Science laboratory for providing a conducive environment that promotes privacy and a cheerful atmosphere, free from the constraints of societal norms. A special mention goes to the new friends I have made over the last year - their support has been invaluable, and they will forever hold a special place in my heart. Finally, I would like to express my gratitude to the beaches of St Andrews, whose breathtaking beauty was a constant source of inspiration and tranquillity during this period.

Software and Code

All analyses presented in this thesis were conducted using R software, version 2022.12.0+353. The complete R code can be found in the Appendix Code Section 10.3 for replication and further exploration. To execute the analysis, initiate the 'main.R' file. Ensure that other associated files are sourced correctly within the 'main.R' for seamless execution.

Contents

List of Figures

List of Tables

1 Introduction

The personal motivation to delve into the intricacies of the Bayesian statistical framework was kindled by Nobel laureate Daniel Kahneman's seminal and highly digestible book 'Thinking Fast and Slow' (Kahneman, 2011). Kahneman's dissection of human cognitive biases and flawed decision-making, especially in ambiguous situations, characterises System 1 thinking as heuristic-driven, low-cost, and low-effort, often resulting in pitfalls of emotionally induced biases. It is posited that the Bayesian methodology parallels System 2 thinking, where prior beliefs are updated with new evidence, considering the strength of the information. The Bayesian approach provides an explicit mathematical framework for handling uncertainty, counterbalancing overconfidence and confirmation bias by quantifying uncertainties in light of new evidence.

Bayesian statistics employs probability distributions, constructed using probability theory, to describe the 'degree of belief'. Its strengths and weaknesses simultaneously lie in the flexible incorporation of background information. Arguably, it presents a more organic approach to scientific reasoning than frequentist methods, updating beliefs with new data. Relying only on data likelihood under specific hypotheses, without prior beliefs, results in a rigid inference process. Priors become especially valuable when data is limited; sometimes, they might be the only available information. Even with a non-informative prior, Bayesian methodology yields a distribution in contrast to the classical approach's point estimate, making it, although subjectively, more intuitively understandable.

Bayesian approaches elegantly circumvent challenges encountered by classical methods. They sidestep issues of functional maximisation, which eliminates problems with algorithm convergence and the delicate task of choosing starting values close to the maximum (Train, 2012). Further, Bayesian methods alleviate the dilemma between local and global maxima, as convergence does not inherently imply the attainment of a global maximum. Moreover, they allow for desirable consistency and efficiency under more forgiving conditions: consistency is achievable with a fixed number of simulation draws, and any increase in draw numbers with sample size ensures efficiency.

This thesis begins with an overview and reflection on personal interests in Chapter 1. Chapter 2 details the motivation for high-dimensional variable selection, while Chapter 3 delves into Bayesian inference theory. Chapter 4 overviews linear regression and model selection across frequentist and Bayesian paradigms. Chapter 5 expands on variable selection techniques, encompassing frequentist, Bayesian, and machine learning approaches. Chapter 6 showcases simulations and their outcomes, leading into Chapter 7's deep dive into a crime dataset analysis. Finally, Chapter 8 provides the conclusions of the research.

2 Motivation

In data-driven decision-making, emphasis is on the abundance of data, adhering to the principle that observations should outnumber explanatory variables to prevent model overfitting and boost predictive power. Technological advancements have accelerated scientific progress, leading to vast data sets where variables often outnumber data points. A key challenge in such high-dimensional settings is achieving sparsity, ensuring model simplicity and interpretability without compromising predictive accuracy, thus addressing the variance and overfitting issues inherent in large datasets (Hastie et al., 2017). For instance, while astronomy and image processing may have thousands of noisy pixel observations, only a small subset is essential to identify key objects (Johnstone & Silverman, 2004). Meanwhile, data can be limited in medical research on rare diseases or novel treatments. Here, the Central Limit Theorem (CLT) might be invoked to assume data distribution, even if sample sizes do not fully support the theorem's accuracy.

Variable selection seeks the optimal subset of predictors and coefficients to drive the most fitting model for the data. However, it is essential to remember that the 'best' model does not claim to uncover the absolute truth about the underlying natural processes, which are far too intricate to be fully captured in mathematical terms (Steyerberg, 2019). With unseen or undiscovered predictors, and potential effects too minuscule to empirically detect, statistical models remain valuable approximations, drawing from the limited palette of known predictors to paint a feasible picture of the complex reality. When the vector of regression coefficients β is large and sparse, that is, most elements are zero or negligible, identifying the significant elements of β becomes particularly important (Moran et al., 2019). The goal is to identify a sparse subset of predictors that adequately capture the true signals within the data, allowing for the construction of parsimonious, interpretable models that effectively mitigate overfitting (J. Fan & Lv, 2008).

Within classical statistics, specifically in high-dimensional contexts, the non-uniqueness of solutions frequently arises. When predictors p outnumber observations n, the matrix $X^T X$ often becomes singular or nearly so (Hastie et al., 2017). This results in ordinary least squares estimates for regression coefficients that are unstable or undefined. One of the primary factors exacerbating this issue is collinearity. Even when predictors are not causally related, their high correlations can obfuscate the interpretation of their individual effects on the response variable. Such collinearity poses a significant challenge in high-dimensional model selection (Jianqing et al., 2010). Moreover, it can be misleading in high-dimensional geometry, potentially guiding statisticians towards an inaccurate model selection (J. Fan & Lv, 2008). This environment further amplifies the risk of overfitting.

In high-dimensional model building, addressing noise is crucial. Noisy data, characterised by corruption or a low signal-to-noise ratio, can skew results if not properly managed, leading to misleading conclusions. While assuming a parametric form streamlines estimation, it might not always capture the true underlying structure, risking inaccurate estimates. Furthermore, highly

flexible models, though adaptable, are prone to overfitting, becoming more attuned to noise than to genuine patterns (Hastie et al., 2017). Such noise can compromise both the reliability and interpretability of a model.

This thesis explores penalised regression methods within the frequentist framework, addressing high dimensionality, collinearity, and noise challenges. Penalised regression techniques impose penalties on regression coefficients, promoting stability amidst collinearity and high dimensionality. In contrast, Bayesian methods, while incorporating priors for structured uncertainty modelling, stand out due to their non-reliance on asymptotic results. As the sample size increases, Bayesian inference capitalises on the asymptotic normality of the posterior distribution, ensuring the parameter vector's posterior distribution converges to multivariate normality for consistent and efficient estimates (Gelman et al., 2020). Machine learning technique Extreme Gradient Boosting (XGBoost) utilise grid search, cross-validation, and tree number limitation to optimise parameters, minimise prediction errors, and guard against overfitting, noise, and spurious correlations. Whilst frequentist and Bayesian methods often set their primary goal as drawing inference, machine learning is often cited for its aim to predict. These methods are defined in detail in subsequent chapters.

This analysis investigates high-dimensional linear regression variable selection using both simulated data, where the number of predictors is smaller, equal, greater or even much greater than the number of data points, and the real data with many predictors and data points. The study employs a mix of Bayesian methods, classical penalised regression, and a gradient-boosted decision tree, all executed in R software. It should be noted that some methodologies deployed in this study are not the original versions but extensions found within the implemented packages. This approach is intended to simplify the reader's narrative and illuminate the adaptations required to overcome computational limitations, enhance methodological efficiency and improve performance outcomes. In doing so, this study provides insights into which adjustments have proved most effective. Furthermore, comparative analyses of different packages for some methodologies are undertaken to compare user-friendliness and consistency.

The aims of this thesis extend beyond applying and comparing various variable selection methods in linear regression problems. Equally important is the personal journey into the depth of the Bayesian statistical framework, as it remained unexplored during my studies. Before my Master's in Applied Statistics and Data Mining, my academic foundation was rooted in a creative discipline. Hence, this exploration of statistical frameworks is a scholarly endeavour and a pivotal chapter in my academic transition and growth. This document delivers definitions of the methods, blending theory with application to foster a deep understanding of the methodology and its practical testing.

3 Bayesian Framework: Building Blocks

This thesis focuses on parametric models, characterised by a finite number of parameters independent of the sample size, belonging to the parameterised family of distributions. The number of parameters reflects model complexity.

The steps involved in Bayesian inference methodology are outlined to facilitate familiarity with the concepts:

1. *Development of a full Probability Model*: A joint probability distribution that encompasses all observable and latent variables is formulated. Ensuring the model is consistent with the prevailing understanding of the scientific problem and the data collection procedure is crucial.
2. *Conditioning on Observed Data*: The posterior distribution is computed and analysed. The most common approach to posterior distribution computation is Markov Chain Monte Carlo (MCMC) and Gibbs Sampling. Given the observed data, this distribution represents the conditional probability of the latent variables of interest.
3. *Assessment of Model Fit and Posterior Distribution Implications*: The model is evaluated, as are the plausibility of the substantive conclusions derived from the posterior distribution. The assessment also includes checking the conclusions' robustness and the results' sensitivity to the initial modelling assumptions. If necessary, the model is modified or expanded, and the process is iterated.

3.1 Foundations of Bayesian Inference

This section relies on 'Bayesian Analysis' lecture material by Fan, Stanford University (2016).

In the Bayesian framework, parameter θ is modelled as a random variable. This paradigm absorb prior beliefs about an unknown parameter θ and revises these beliefs upon observing new data. Prior to observing any data, the unknown parameter is represented as a random variable Θ with a probability distribution $f_\Theta(\theta)$, known as the prior distribution (later denoted as $p(\theta)$). This distribution embodies the initial beliefs about the parameter's value.

Conditional on $\Theta = \theta$, the observed data X is assumed to follow the distribution $f_{X|\Theta}(x|\theta)$. This distribution characterises a parametric model parameterised by θ. The joint distribution of Θ and X is expressed as:

$$f_{X,\Theta}(x, \theta) = f_{X|\Theta}(x|\theta) f_\Theta(\theta) \tag{1}$$

For continuous cases, the marginal distribution of X is given by:

$$f_X(x) = \int f_{X,\Theta}(x,\theta)d\theta = \int f_{X|\Theta}(x|\theta)f_\Theta(\theta)d\theta \tag{2}$$

The conditional distribution of Θ given $X = x$ is defined as:

$$f_{\Theta|X}(\theta|x) = \frac{f_{X|\Theta}(x|\theta)f_\Theta(\theta)}{\int f_{X|\Theta}(x|\theta')f_\Theta(\theta')d\theta'} \tag{3}$$

This equation is derived using Bayes' theorem, which relates the posterior distribution, the likelihood, and the prior distribution. Specifically, the numerator represents the product of the likelihood and the prior, while the denominator ensures normalisation. $f_{\Theta|X}(\theta|x)$ (later presented as $\pi(\theta|x)$) is termed the posterior distribution of Θ, representing our updated understanding of the parameter Θ after observing data X. Concisely, it can be articulated as:

$$\begin{gathered} f_{\Theta|X}(\theta|x) \propto f_{X|\Theta}(x|\theta)f_\Theta(\theta) \\ \text{Posterior density} \propto \text{Likelihood} \times \text{Prior density} \end{gathered} \tag{4}$$

where the posterior density is proportional to the product of the likelihood and the prior density. The symbol $\propto$ masks the proportionality factor $f_X(x) = \int f_{X|\Theta}(x|\theta')f_\Theta(\theta')d\theta'$, which is independent of θ.

The *prior distribution* represents our knowledge or beliefs about a parameter before observing any data. Priors can be categorised as either 'informative' or 'uninformative'. 'Informative' priors are derived from sources such as additional data, insights from experts, or elicitation methods (Dias et al., 2017). On the other hand, uninformative priors are typically broad or non-committal distributions, serving as placeholders in the absence of strong prior beliefs (examples include uniform or Jeffreys' priors) (Price & Manson, 2002). Prior selection will play a crucial role for variable selection in this thesis.

The likelihood function $f_{X|\Theta}(x|\theta)$, later presented as $p(x|\theta)$, evaluates how probable the observed data x is under various parameter values θ. Unlike the prior distribution, the likelihood function depends solely on the data and quantifies its support for various parameter values. The likelihood function is not a probability distribution over θ, that is, it does not provide probabilities for different parameter values but rather gives a measure of how well each parameter value θ is supported by the data.

The *posterior distribution* synthesises all available information regarding the parameter of interest. However, deriving analytical summaries, such as the posterior distribution's mean, variance, credible intervals, or inclusion probabilities often requires evaluating complex integrals. The evaluation can be incredibly challenging for high-dimensional posterior distributions. Monte Carlo integration, a simulation technique, offers an effective solution for estimating these integrals. Within the Monte

Carlo framework, the Markov Chain Monte Carlo (MCMC) methodology is a powerful tool for approximating all these posterior summary statistics. The application of this methodology is defined in the following section.

3.2 MCMC Algorithm

The following description lays the foundation of *Markov Chain Monte Carlo (MCMC)* algorithms, of which three are particularly prominent: Gibbs sampling, Metropolis-Hastings, and Importance/Rejection sampling.

MCMC involves generating samples of θ from approximate distributions and iteratively refining these samples to converge to the desired posterior distribution, $\pi(\theta|x)$. The essence of *MCMC* is not the Markov property per se but the progressive improvement of the approximation towards the target distribution with each iteration. A Markov chain is defined as a stochastic sequence $\{\theta^0, \theta^1, \theta^2, \ldots, \theta^n\}$, where each state θ^n depends only on its immediate predecessor θ^{n-1}, where the initial state θ^0 is set to an arbitrary value. The Markov chain evolves according to a transition kernel P, dependent only on θ^n:

$$\theta^{n+1} \sim P(\theta^n, \theta^{n+1}) \, (\equiv P(\theta^{n+1}|\theta^n)). \tag{5}$$

Given the conditions of aperiodicity and irreducibility, a Markov chain will converge to a stationary distribution that does not depend on its initial values. In practical applications, determining the exact point of convergence is challenging. The initial states of the chain, which might not accurately represent the posterior distribution, are typically discarded in a process known as 'burn-in' (Brooks & Gelman, 1998). This ensures that the chain has sufficiently converged to a stable mean. Various techniques, including trace plots (Plummer et al., 2005) and the Brooks-Gelman-Rubin (BGR) method (Brooks & Gelman, 1998), help ascertain the burn-in length. BGR method uses an analysis of variance to assess the similarity of estimates from different starting points.

Once the burn-in phase is complete, it is crucial to have an adequate number of iterations to ensure accurate summary statistics and minimal Monte Carlo errors. A common approach to estimating the Monte Carlo error involves batching. The chain is divided into m batches, each of length T, so that $n = mT$. Let $\theta_1, \ldots, \theta_m$ be the sample means for each batch, and θ denote the mean overall n samples. The batch means estimate of σ^2 is then,

$$\hat{\sigma}^2 = \frac{T}{m-1} \sum_{i=1}^{m} (\bar{\theta}_i - \bar{\theta})^2. \tag{6}$$

An estimate of the Monte Carlo error is $\sqrt{\frac{\hat{\sigma}^2}{n}}$. The efficiency of the Markov chain in exploring the parameter space can be evaluated using the autocorrelation function (ACF). The ACF is the

correlation of a parameter's value in the Markov chain with itself at a lag j, defined as $\text{cor}(\theta^t, \theta^{t+j})$. Efficient chains show a fast decrease in ACF as the lag increases, indicating low dependency between chain values within a few iterations.

An alternative estimate of the Monte Carlo error uses the effective sample size M, defined as

$$M = \frac{n}{1 + 2\sum_{k=1}^{\infty} \rho_k}, \tag{7}$$

where ρ_k is the autocorrelation at lag k. Practically, M is estimated through an alternative method accounting for autocorrelations. The Monte Carlo error can be estimated as $\sqrt{\frac{\hat{\sigma}^2}{\hat{M}}}$.

Finally, 'thinning' in the context of *MCMC* refers to selecting every k_{th} realisation from the chain, discarding the rest. This process helps reduce the autocorrelation within the sampled sequence. While thinning can be beneficial, especially in memory-constrained situations, it should be used judiciously. By discarding samples, valuable information might be lost.

Section 4.5 will introduce the Reversible Jump MCMC method, an advanced extension of *MCMC* that involves Gibbs sampling, which is the most relevant to this paper.

4 Model Selection Methodology

4.1 The Setting: Linear Regression Model

The thesis explores the multiple linear regression model within the frequentist statistical framework, described as

$$\mathbf{Y} = \beta_0 + \mathbf{X}\boldsymbol{\beta} + \boldsymbol{\epsilon}, \quad \boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \sigma^2 \mathbf{I}_n). \tag{8}$$

Here, $\mathbf{Y} \in \mathbb{R}^n$ is the response vector, β_0 is the intercept, $\mathbf{X} \in \mathbb{R}^{n \times p}$ is the design matrix comprising p potential predictors. The vector $\boldsymbol{\beta} = (\beta_1, ..., \beta_p)^T \in \mathbb{R}^p$ represents the set of regression coefficients that will be estimated. The noise vector $\boldsymbol{\epsilon} \in \mathbb{R}^n$ is constituted by independently distributed normal random variables, each sharing a common but unknown variance σ^2. The term $\mathbf{I}_n$ denotes the $n \times n$ identity matrix, ensuring that the noise components are uncorrelated and have constant variance.

Transitioning to the Bayesian framework, the likelihood becomes:

$$\mathbf{Y}|\alpha, \boldsymbol{\beta}, \sigma^2 \sim \mathcal{N}_n(\mathbf{1}_n\alpha + \mathbf{X}\boldsymbol{\beta}, \sigma^2 \mathbf{I}_n), \tag{9}$$

where the term $\mathbf{1}_n\alpha$ creates a vector of length n where every element is the intercept α.

In Bayesian regression, parameters α, $\boldsymbol{\beta}$, and σ^2 are treated as random variables with respective prior distributions:

$$\begin{aligned} \boldsymbol{\beta} &\sim \pi(\boldsymbol{\beta}), \\ \alpha &\sim \pi(\alpha), \\ \sigma^2 &\sim \pi(\sigma^2). \end{aligned} \tag{10}$$

This thesis discusses methods for assessing the relative quality of statistical models, including the Akaike information criterion (AIC) and Watanabe–Akaike information criterion (WAIC). While the analysis focuses on variable selection, the task of model selection is closely related, and some of the applied R packages use these criteria (for example, *'spikeslab'* calculates AIC criteria, *'bayesreg'* - WAIC), warranting their brief explanation for methodological completeness and personal understanding.

4.2 AIC

The following three sections draw from 'Bayesian Data Analysis' by Gelman et al. (2020).

In classical statistics, parameter θ is typically inferred using a point estimate, denoted as $\hat{\theta}$, rather than the full posterior distribution. The maximum likelihood estimate (MLE) is often used as this

point estimate. MLE determines the parameter values of a statistical model that maximise the likelihood of observing the given data. A common approach for calculating out-of-sample predictive accuracy involves using the log posterior density of the observed data x given the point estimate, $\log p(x|\hat{\theta})$, and correcting overfitting bias. When k represents the number of estimated parameters, the bias penalisation is performed by subtracting k from the log predictive density based on the MLE, according to the formula:

$$\widehat{elpd}_{\text{AIC}} = \log p(x|\hat{\theta}_{\text{mle}}) - k. \tag{11}$$

AIC is then defined as twice the negative of this quantity:

$$\text{AIC} = -2 \log p(x|\hat{\theta}_{\text{mle}}) + 2k. \tag{12}$$

AIC favours models that predict well and penalises models with excessive parameters to discourage overfitting. Though *AIC's* bias correction, under certain assumptions, is applicable in normal linear models with known variance and uniform priors, it is inadequate in Bayesian models. In such cases, the penalty of k does not accurately represent the effective number of parameters. Hence, other criteria were introduced.

4.3 WAIC

WAIC requires Monte Carlo estimation and is complex to implement but offers a fully Bayesian approach to estimate out-of-sample expectations (Spiegelhalter et al., 2014). This method calculates the log pointwise posterior predictive density and incorporates a correction for the effective number of parameters to prevent overfitting.

Here, the robust WAIC modification is explored. The effective number of parameters, p_{WAIC2}, is derived from the variance of individual terms in the log predictive density across all data points n as

$$p_{WAIC2} = \sum_{i=1}^{n} \text{var}_{\text{post}}(\log p(x_i|\theta)). \tag{13}$$

Using the sample variance definition, the total variance across all data points is computed to obtain the effective number of parameters. Then to correct for bias:

$$\widehat{\text{elppd}}_{\text{WAIC}} = \text{lppd} - \text{p}_{\text{WAIC}}. \tag{14}$$

Finally, similar to AIC and DIC, the WAIC is determined by:

$$WAIC = -2\text{lppd} + 2\text{p}_{WAIC2}, \tag{15}$$

where *lppd* is the computed log pointwise predictive density, derived from the average of the predictive densities across all data points.

Unlike *AIC*, which gauges the plug-in predictive density's performance, *WAIC* assesses predictions for new data in a Bayesian context by averaging over the posterior distribution. While *AIC*, and *WAIC* aim to estimate the expected out-of-sample deviance of a model, akin to versions of cross-validation, Bayesian Information Criterion (BIC) focuses on approximating the marginal probability density of the data under the model, pertinent in the context of discrete model comparison (2012).

4.4 Bayes Factor and Reversible Jump Monte Carlo

The *Bayes factor*, introduced by Jeffrey (1935), is a key tool in traditional Bayesian model comparison, widely discussed in Bayesian literature. It quantifies the relative evidence for two models, M_i and M_j, given data x,

$$B_{ij} = \frac{p(x|M_i)}{p(x|M_j)} = \frac{\int p(x|\theta_i, M_i)p(\theta_i|M_i)d\theta_i}{\int p(x|\theta_j, M_j)p(\theta_j|M_j)d\theta_j}, \tag{16}$$

where $p(x|\theta_k, M_k)$ denotes the likelihood under model k, and $p(\theta_k|M_k)$ represents the prior distribution of θ_k. The variable M denotes the model, taking values from a finite set of K models. *Bayes factors* can be derived from posterior model probabilities with known prior model probabilities (Kass & Raftery, 1995), enabling Bayesian model averaging, which accounts for model uncertainty in posterior estimates (Hoeting et al., 1999).

Evaluating *Bayes factors* is challenging due to the difficulty computing marginal likelihoods for each model. MCMC methods offer a solution but necessitate careful handling of varying parameter numbers to ensure ergodicity in the Markov chain. Green's (1995) *Reversible Jump MCMC (RJMCMC)* employs auxiliary variables to manage model dimension changes. Defining pseudo-priors or auxiliary variables remains a challenging area of ongoing research. Barker & Link (2013) introduced a simplified *RJMCMC* approach, extending the Metropolis-Hastings algorithm to allow the Markov chain to traverse varying dimensions. This methodology, further developed by Gelling, Schofield, and Barker (2019), is encapsulated in the *'rjmcmc'* package, facilitating Bayes factor and posterior model probability calculations using *RJMCMC* outputs.

Given data x with models indexed $1, ..., K$, and a set of model-specific parameters θ_k for each model k, along with prior model probabilities $p(M_k)$, the posterior model probabilities are related to Bayes factors as:

$$\frac{p(M_i|x)}{p(M_j|x)} = B_{ij} \times \frac{p(M_i)}{p(M_j)} \tag{17}$$

RJMCMC enables sampling across models by considering the model indicator as a latent variable sampled using MCMC. The transition between models i and j necessitates that: both models have an equal number of parameters, and a bijective mapping exists between the parameters of the two models. Auxiliary variables u_i are introduced to ensure the dimensions match, that is, $\dim(\theta_i, u_i) = \dim(\theta_j, u_j)$. With the freedom to transition between any pair of models, $K(K-1)/2$ bijections must be defined. The choice of auxiliary variables and bijections does not alter the posterior distribution but affects computational efficiency. Limiting transitions between models can reduce the number of required bijections. During the t iteration of the Markov chain, a proposed model M_j^* is introduced, while the current value is denoted as $M_i^{(t-1)}$. For model M_j^*, the proposed parameter values are determined using the bijection $f_{ij}(\cdot)$ as

$$(\theta_j^*, u_j^*) = f_{ij}(\theta_i^{(t-1)}, u_i^{(t-1)}). \tag{18}$$

The joint proposal is accepted or rejected using a Metropolis step. The selection of the bijection is crucial, as it affects the efficiency and convergence rate of the chain. The restricted version of *RJMCMC* involves introducing a universal parameter vector ψ, whose dimension is at least the maximum dimension of the model-specific parameters, that is,

$$\dim(\psi) \geq \max_k \{\dim(\theta_k)\}. \tag{19}$$

Model-specific parameters θ_i and auxiliary variables u_i are derived from ψ using a bijection $g_i(\cdot)$:

$$(\theta_i, u_i) = g_i(\psi), \tag{20}$$

$$\psi = g_i^{-1}(\theta_i, u_i). \tag{21}$$

Model parameters in model i are mapped to model j through the universal parameter vector ψ,

$$\begin{aligned}(\theta_j, u_j) &= g_j(\psi) \\ &= g_j(g_i^{-1}(\theta_i, u_i)).\end{aligned} \tag{22}$$

This method necessitates K bijections to move among K models. The joint distribution is expressed as:

$$p(y, \psi, M_k) = p(y|\psi, M_k)p(\psi|M_k)p(M_k), \tag{23}$$

where $p(y|\psi, M_k) = p(y|\theta_k, M_k)$ is the joint probability density for the data under model k, $p(\psi|M_k)$ is the prior for ψ given model k, and $p(M_k)$ is the prior model probability for model k.

Since priors are typically in the form $p(\theta_k|M_k)$, $p(\psi|M_k)$ is found as:

$$p(\psi|M_k) = p(g_k(\psi)|M_k) \left| \frac{\partial g_k(\psi)}{\partial \psi} \right|, \tag{24}$$

where $\left|\frac{\partial g_k(\psi)}{\partial \psi}\right|$is the determinant of the Jacobian for the bijection g_k, later denoted as $|J_k|$. The algorithm employs a Gibbs sampler that alternates between updating M and ψ. The full-conditional distribution for M is categorical, with probabilities:

$$p(M_k|\cdot) = \frac{p(y, \psi, M_k)}{\sum_j p(y, \psi, M_j)}. \tag{25}$$

A sample from the full-conditional for ψ is obtained by drawing θ_k and u_k from their respective distributions and computing $\psi = g_k^{-1}(\theta_k, u_k)$. Barker and Link (2013) also detailed a Rao-Blackwellized estimator for posterior model probabilities based on estimating a transition matrix whose (i, j) entry represents the probability of transitioning from model M_i to M_j. The posterior model probabilities are derived by normalising the left eigenvector of this estimated transition matrix. An essential feature of the *'rjmcmc'* package is the automatic computation of $|J_k|$ through automatic differentiation, which greatly simplifies implementation, especially when dealing with many parameters.

The *'rjmcmc'* package facilitates precise estimation of Bayes factors and posterior model probabilities for a predefined set of models. While the original Green's *RJMCMC* (1995) is a versatile algorithm, allowing parameter changes in MCMC simulations, the *'rjmcmc'* package is tailored to refine posterior distributions and facilitate model comparison. Notably, it only permits transitions between models with equal parameters. Comprehensive domain knowledge of the socio-economic data analysed in later chapters is necessary, including the ability to pre-select variables, a requirement that could not be met. Prioritising established methodologies ensured reliability in the findings of this dissertation.

5 Variable Selection Methodology

5.1 Frequentist Penalised Regression

In penalised regression methods, a penalty term is added to the log-likelihood function to enforce a trade-off between bias and variance in regression coefficients, consequently optimising prediction error.

Least Absolute Shrinkage and Selection Operator (Lasso) incorporates the L1-norm, originally proposed by Tibshirani (1996), of regression coefficients (excluding the intercept) as the penalty term:

$$-\log L + \lambda \sum_{j=1}^{p} |\beta_j|, \quad \lambda > 0. \tag{26}$$

This penalisation shrinks coefficients toward zero and sets those with a negligible predictive contribution to zero, serving as an embedded feature selection method. *Lasso* does not group predictors, often arbitrarily selecting one from a group of highly correlated predictors.

Ridge regression employs the L2-norm of regression coefficients (excluding the intercept) as the penalty:

$$-\log L + \lambda \sum_{j=1}^{p} \beta_j^2, \quad \lambda > 0. \tag{27}$$

Ridge regression shrinks coefficients towards zero but retains all predictors in the model. When $n > p$ and there is high multicollinearity, *Ridge regression* often offers superior predictions. Since the penalty term is the sum of squared coefficients, shrinkage would be unfair across predictors with different scales. Hence, they need to be standardised.

Elastic-Net combines the L1-norm and L2-norm penalties, the method first proposed by Zou and Hastie (2005):

$$-\log L + \lambda_1 \sum_{j=1}^{p} |\beta_j| + \lambda_2 \sum_{j=1}^{p} \beta_j^2, \quad \lambda_1, \lambda_2 > 0. \tag{28}$$

The combination of penalties can also be expressed as:

$$\lambda \left(\alpha \sum_{j=1}^{p} |\beta_j| + \frac{1-\alpha}{2} \sum_{j=1}^{p} \beta_j^2 \right), \quad \lambda > 0,\ 0 \leq \alpha \leq 1. \tag{29}$$

Here, α controls the mixing of *Lasso* and *Ridge* penalties, and λ regulates the overall strength of regularisation, see Figure 1.

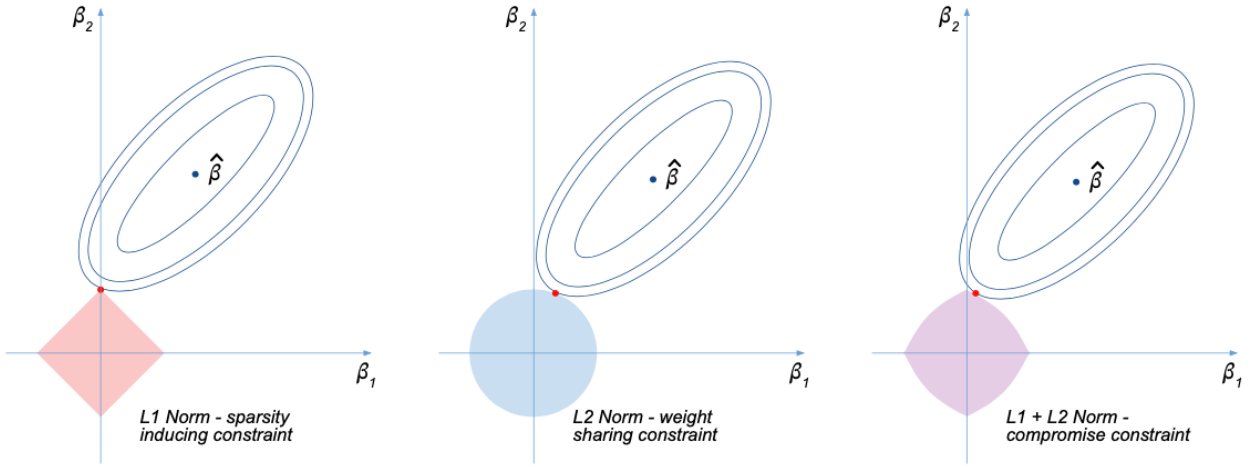

Figure 1: Contours of the Residual Sum of Squares (RSS) and Constraint Functions for Penalised Regression Techniques. Lasso (left), Ridge (centre), and Elastic-Net (right). Note: The shaded regions represent the constraint boundaries, blue ellipses depict the RSS contours.

Despite its proven efficacy, the original *Lasso* method has certain constraints. Tibshirani (1996) noted that ridge regression surpasses *Lasso* when dealing with multicollinearity among predictors. In situations with more predictors, p, than observations, n, Lasso's convex optimisation limits it to selecting no more than n variables. Moreover, it disregards meaningful feature ordering and struggles to effectively select highly correlated grouped variables, tending to choose individual ones instead. Later, Meier, Van De Geer and Bühlmann (Meier et al., 2008) introduced algorithms designed for extremely high-dimensional problems to solve convex optimisation issues. They demonstrated that the group *Lasso* estimator for logistic regression remains statistically consistent with a sparse true underlying structure, even when the number of predictors significantly outnumbers the observations.

The *'glmnet'* package in R provides generalised linear model fitting, allowing for Lasso or elastic-net regularisation with a spectrum of lambda values, and includes capabilities for prediction, plotting, and cross-validation, even for sparse datasets. Ten k-fold cross-validations are used to determine λ, which controls the overall strength of the penalty in the *'glmnet'* package under the Lasso ($\alpha = 1$) or the elastic-net ($alpha = 0.5$) penalties.

5.2 Machine Learning

Exploring variable selection methods and comparing frequentist and Bayesian inference approaches with a renowned machine learning method, *Extreme Gradient Boosting (XGBoost)*, would offer valuable insights. *XGBoost* introduced by Chen and Guestrin (2016) is often cited for outstanding performance in Kaggle competitions. The method incorporates a feature importance mechanism, which, in simple terms, quantifies the contribution of individual attributes to the construction of

decision trees within the ensemble (Chen & Guestrin, 2016).

XGBoost boasts exceptional scalability, enabling rapid processing on single machines and adept scaling to billions of examples in memory-constrained environments. As a comprehensive tree-boosting system, it introduces innovations such as a sparsity-aware algorithm for sparse data and a theoretically grounded weighted quantile sketch for handling instance weights, effectively streamlining resource employment in processing large datasets (Chen & Guestrin, 2016).

The following methodology is based on Wang, Xu, Zhao, Peng and Wang's definition (2019).

The *XGBoost* model:

$$\hat{y}_i = \sum_{k=1}^{K} f_k(x_i), \quad f_k \in \mathcal{F} \tag{30}$$

where $\hat{y}_i$ is the predicted value for the i-th instance, K denotes the number of trees, $\mathcal{F}$ denotes the set of all possible regression trees, and f_k represents a specific regression tree.

The goal of *XGBoost* is to build a K regression tree such that the predictions of the tree group are as close as possible to the true values while ensuring the greatest generalisation ability. The prediction process is achieved by minimising an objective function, given by:

$$\text{obj}(\theta) = \sum_{i}^{n} l(y_i, \hat{y}_i) + \sum_{k=1}^{K} \Omega(f_k), \tag{31}$$

where the first component, $\sum_{i=1}^{n} l(y_i, \hat{y}i)$, is a loss function that measures the deviation of the predicted values from the true values; the second part, $\sum_{k=1}^{K} \Omega(f_k)$, acts as a regularisation term that controls the complexity of the model, as:

$$\Omega(f_k) = \gamma T + \frac{1}{2}\lambda \|w\|^2, \tag{32}$$

where T represents the number of leave nodes in the tree, and $\|w\|^2$ is the weight of the corresponding leaf nodes.

During the t-th iteration of training, the objective function is defined as:

$$\text{obj}^{(t)} = \sum_{i=1}^{n} l\left(y_i, \hat{y}_i^{(t-1)} + f_t(x_i)\right) + \Omega(f_k) + \sum_{i=1}^{t-1} \Omega(f_i) \tag{33}$$

This formulation encapsulates each tree's training error and complexity, steering the algorithm toward building an ensemble of trees that effectively balances accuracy and generalisation.

Figure 2 illustrates the tree-boosting process based on Guo et al. (2020).

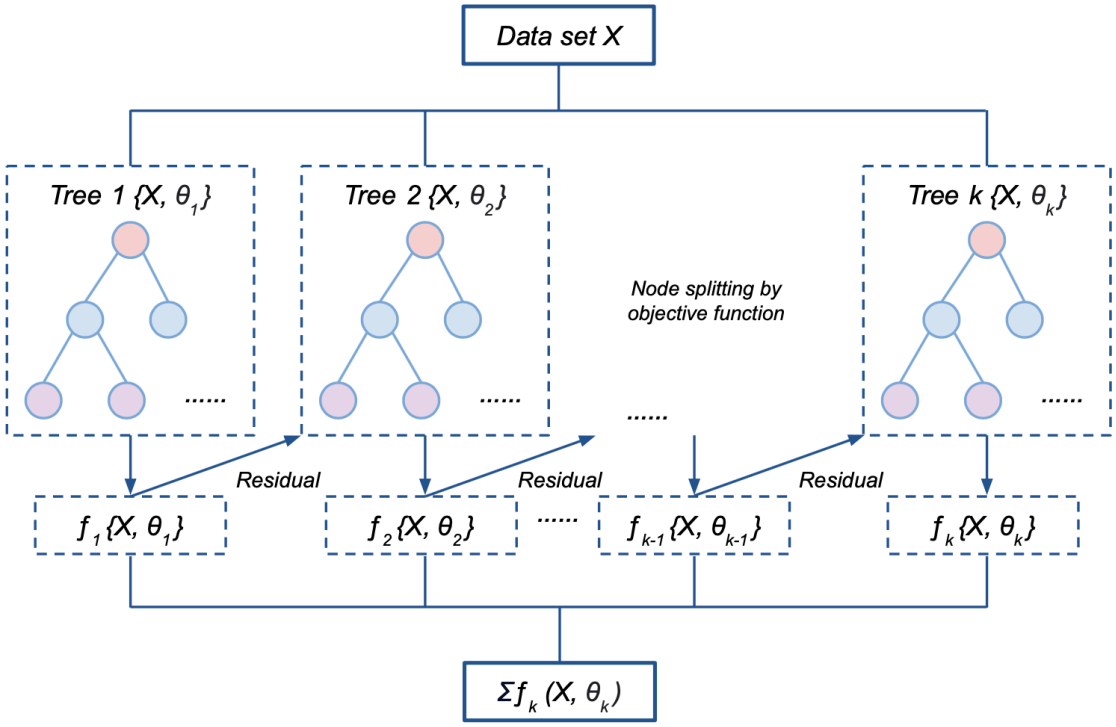

Figure 2: XGBoost Algorithm. Iterative Tree Building Process

The *XGBoost* model, known for its accuracy and resistance to overfitting (Wang et al., 2019), supports weak classification and regression models. Although its efficacy in classification is recognised, its use in high-dimensional settings is debated (Li, 2020). In this study, the real dataset being investigated comprises a scenario where the number of data points exceeds the number of features, albeit with a large feature space. It is essential to incorporate an additional augmented simulated dataset configuration defined in Section 6.1. Additionally, Adebisi (2020) highlighted potential difficulties faced by *XGBoost* when tasked with linear regression with a continuous target. Notably, the majority of studies employing this method focus on classification rather than linear regression problems. This topic is delved into further in the Conclusions chapter. To note, it is sensitive to outliers, too (Dairu & Shilong, 2021).

This thesis employs the *XGBoost* model to calculate feature importance, ranking variables by their significance in building boosted trees. Features with non-zero importance scores are selected, typically encompassing most covariates. The top variables are identified by aggregating the gain, focusing on features contributing to 80% of the total gain, in line with the Pareto principle.

In the analysis, a grid search optimises hyperparameters, including boosting rounds (50, 100, 150), tree depth levels (3, 5, 7, 9), learning rates (0.01, 0.1, 0.3), and minimum loss reduction values (0, 0.1, 1). Column subsample ratios during tree construction are set at 0.6, 0.8, and 1, with minimum instance weight in a child node at 1, 3, and 5. Training instance subsample ratios are 0.8 and 1.

A 5-fold cross-validation, repeated thrice, is used. The model focuses on linear regression, with performance assessed by the Root Mean Square Error (RMSE) during tuning.

5.3 Bayesian Framework

5.3.1 Bayesian Lasso

The Lasso, a frequentist penalised regression approach, aims to minimise the Residual Sum of Squares (RSS) subject to the non-differentiable constraint on the coefficients using the L1-norm, expressed as:

$$\min_{\beta} (\tilde{y} - X\beta)^T (\tilde{y} - X\beta) + \lambda \sum_{j=1}^{p} |\beta_j| \tag{34}$$

where $\tilde{y} = y - \bar{y}1_n$ for some $\lambda \geq 0$.

Tibshirani (1996) interpreted the Lasso estimates as Bayes posterior mode under individual Laplace priors for each predictor. The Laplace distribution's ability to manifest as a scale mixture of normal distributions with independently exponentially distributed variances offers advantages. It prompted several researchers to adopt Laplace priors within a hierarchical Bayesian framework. This thesis discusses the one the *'monomvn'* package implements. Park and Casella (2008) proposed Gibbs sampling for the Lasso, incorporating a Laplace prior within the hierarchical model. They considered a fully Bayesian analysis using a conditional Laplace prior as:

$$\pi(\beta|\sigma^2) = \prod_{j=1}^{p} \frac{\lambda}{2\sigma} e^{-\lambda|\beta_j|/\sigma} \tag{35}$$

where the noninformative, scale-invariant marginal prior is $\pi(\sigma^2) = \frac{1}{\sigma^2}$. The conditioning on σ^2 asserts that it secures a unimodal full posterior. A lack of unimodality can hinder the convergence of the Gibbs sampler, thus rendering point estimates less reliable.

Park and Casella (2008) argue that the *Bayesian Lasso* offers a middle ground between Lasso and Ridge regression by providing smooth paths similar to Ridge but with a tendency to push less significant parameters towards zero faster, akin to Lasso. This behaviour suggests an edge of the Laplace prior over Gaussian or Student-t priors in rapidly diminishing weakly related parameters. However, this becomes an issue when the method is tasked with variable selection, as the weakly related predictors would become not precisely zero but only relatively small. The Bayesian Lasso employs a double-exponential (Laplace) prior on the regression coefficients, leading to the coefficients' shrinkage towards zero, refer to Figure 3.

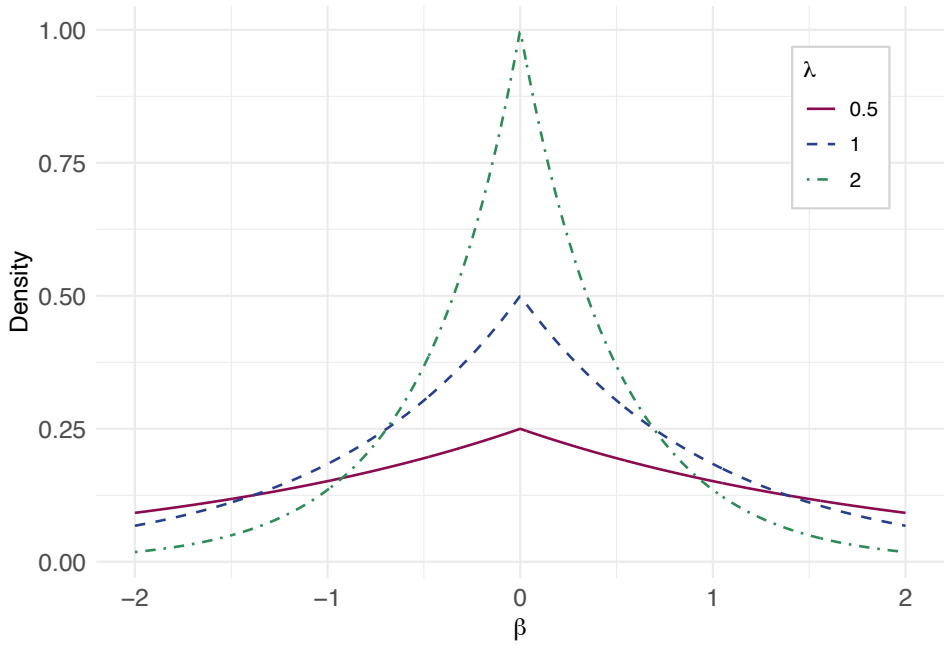

Figure 3: Density Plots of Laplace Prior Distributions for Bayesian Lasso: Impact of tau

The *'monomvn'* package employs the Bayesian Lasso model via the Gibbs Sampling algorithm from Park and Casella (2008). It introduced a feature to use a Rao-Blackwellized sample of σ^2, with β integrated out, to improve the mixing of the sampling algorithm. A unique part of this package is the inclusion of RJMCMC (default) for Bayesian model selection and averaging, which allows users to determine the best model based on the columns of the design matrix and their corresponding β parameters. Unlike Hans (2009) and Geweke (1996) methods, which require a specific prior on each β_i and individual conditional sampling, this implementation maintains joint sampling from the full β vector of non-zero entries, thus facilitating better Markov chain mixing. It also allows RJ proposals to alter the count of non-zero entries on a component-wise basis, with high acceptance rates due to marginalised between-model moves.

To apply the *Bayesian Lasso* model, users set the initial Lasso penalty parameter square, λ_2; default is 1. For a conservative approach, a burn-in of 1,000 samples is used, with inference drawn from the subsequent 5,000 samples. Variable normalisation is disabled; data is manually standardised as outlined in Section 6.2. Other hyperparameters are left as default. Variables with an inclusion probability over 0.5 are considered significant.

5.3.2 Spike-and Slab Prior

The *spike-and-slab* approach was initially pioneered by Lempers (1971), Mitchell, and Beauchamp (Mitchell & Beauchamp, 1988). The term 'spike-and slab' refers to a two-component mixture prior used for β. This prior was set such that the β_k elements were mutually independent, consisting of a flat uniform distribution (the slab) and a zero degenerate distribution (the spike). See Figure 4

illustrating two samples drawn from normal distributions, representing the 'spike' and the 'slab'. The 'spike' represents a sample from a distribution with a small standard deviation, mirroring the zero degenerate distribution in the *spike-and-slab* prior. The 'slab' is a sample drawn from a distribution with a large standard deviation, representing the flat, uniform distribution in the prior.

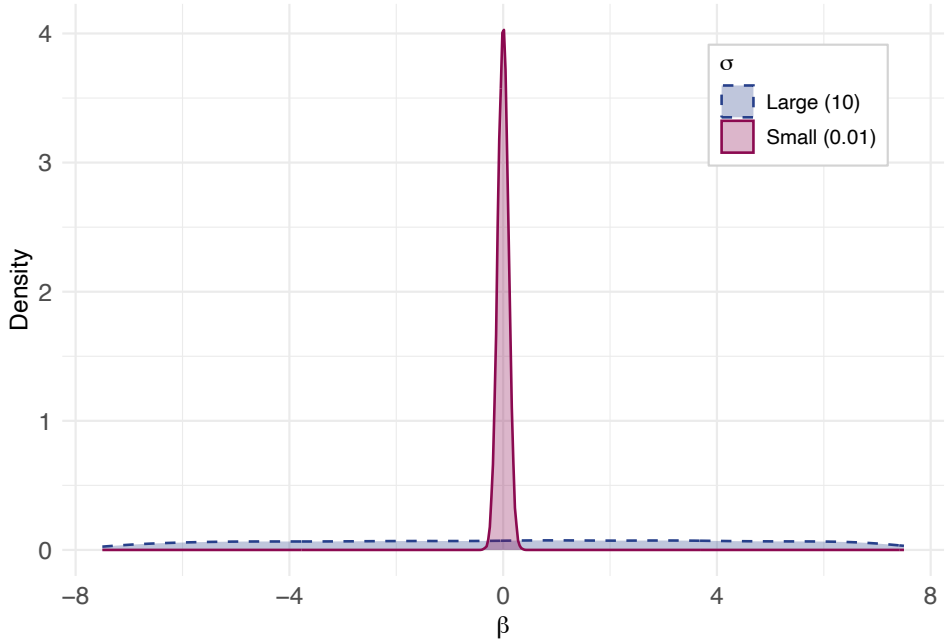

Figure 4: Density Plots of Three Spike-and-Slab Variants

Ishwaran and Rao (2005) proposed a departure from this design. Instead of a two-component mixture, they posited a multivariate normal scale mixture distribution for β, dictated by the prior π for the hypervariance γ. Despite the divergence in distribution choice, the core principle paralleled the original methodology, aiming to shrink truly zero β_k coefficients via small posterior mean values. The hypervariances played a vital role in this, with smaller values driving coefficient shrinkage and larger ones inflating coefficients for final model selection.

In further developing the *spike-and-slab* model, Ishwaran and Rao (2005) introduced a continuous bimodal prior in a rescaled model variant. The use of such a flexible prior helps to alleviate calibration challenges. To prevent the diminishing influence of the prior on the posterior as the sample size increases, they proposed a modification: a sample size invariant or 'universal' rescaling of the *spike-and-slab* model. This modification entails transforming the original Y_i values by a factor of $\sqrt{n}$ and incorporating a variance inflation factor to compensate for the altered variance of the transformed data. The chosen inflation factor can be viewed as a penalisation shrinkage effect of the posterior mean. They demonstrated that selecting n as the inflation factor ensures that the prior continues to influence the posterior, avoiding a vanishing effect. Coupled with a suitably chosen prior for γ, this provides a robust model selection procedure based on the posterior mean, yielding superior performance over ordinary least squares (OLS) methods (N. Polson & Sun, 2017).

The rescaled *spike-and-slab* model is defined by a Bayesian hierarchical structure as follows (Ishwaran & Rao, 2005):

$$\begin{aligned}(Y_i^*|x_i,\boldsymbol{\beta},\sigma^2) &\overset{ind}{\sim} \mathcal{N}(x_i^t\boldsymbol{\beta},\sigma^2\lambda_n), \quad i=1,\ldots,n,\\ (\boldsymbol{\beta}|\boldsymbol{\gamma}) &\sim \mathcal{N}(\mathbf{0},\boldsymbol{\Gamma}), \quad \boldsymbol{\Gamma}=\mathrm{diag}(\gamma_1,\ldots,\gamma_K),\\ \boldsymbol{\gamma} &\sim \pi(d\boldsymbol{\gamma}),\\ \sigma^2 &\sim \mu(d\sigma^2),\end{aligned} \tag{36}$$

where the values of $Y_i^* = \hat{\sigma}_n^{-1} n^{1/2} Y_i$ are scaled versions of the original response Y_i, where $\hat{\sigma}_n^2 = ||Y - X\hat{\beta}_n^\circ||^2/(n-K)$ serves as an unbiased estimate for σ_0^2 based on the full model, and $\hat{\beta}_n^\circ = (X^tX)^{-1}X^tY$ is the ordinary least squares estimate for β_0.

Here, λ_n is a variance inflation factor introduced to account for the scaling of the Y_i's. While a natural choice for λ_n might be n, to match the $\sqrt{n}$-scaling, λ_n also plays a critical role in controlling the increase in the variance of the data. In this context, $\lambda_n = n$ represents the penalisation necessary to guarantee a significant shrinkage effect in the limit.

The *'spikeslab'* package, developed by Ishwaran, Kogalur and Rao (2010), implements a rescaled three-step *spike-and-slab* algorithm using the generalised elastic-net (gnet) and Bayesian model averages (BMA) estimator. The BMA estimator adeptly handles correlation issues common in high-dimensional datasets. It leverages the strengths of weighted generalised ridge regression (WGRR), showcasing a key advantage of the Bayesian approach. On the other hand, the gnet estimator applies the principle of soft-thresholding, a potent frequentist regularisation concept, to achieve sparse variable selection in complex, high-dimensional data settings.

The algorithm that underlines the package involves three main steps: First, variables are filtered down to the top nF, where n is the sample size and $F > 0$ is the user-specified fraction, and ordered based on absolute posterior mean coefficient value, computed via Gibbs sampling applied to an approximate rescaled *spike-and-slab* posterior. For cases where $p \geq n$ users can apply this filtering step. Further, a rescaled *spike-and-slab* model is fitted to unfiltered variables from Step 1 using a Gibbs sampler, employing a blocking technique for computational efficiency, and the posterior mean of the regression coefficients BMA is computed and returned as an estimator for the regression coefficients.

Lastly, the gnet estimator is computed with its L2-regularisation parameters fixed, determined from the restricted BMA of Step 2, and its solution path for L1-regularisation is obtained using the *'lars'* package, selecting the model that minimises the AIC criterion, negating the need for cross-validation. Unlike the elastic-net approach, this method simplifies optimisation and reduces computational time, especially in high-dimensional problems (Ishwaran et al., 2010).

To fit the model using the *spike-and-slab* approach, default hyperparameters are employed unless specified otherwise: A burn-in of 1,000 Gibbs samples is initiated, followed by the collection of 5,000

Gibbs samples for inference. The options 'big p small n', 'big p small n factor', and 'screen' for filtering are activated only in scenarios where $p \geq n$. Variables possessing non-zero gnet coefficient estimates are typically chosen.

5.3.3 Spike-and-Slab Prior Meets Lasso

In frequentist statistics, sparse recovery for β is often achieved through the Lasso, whereas in the Bayesian domain, *spike-and-slab priors* are favoured for sparse modelling of β. In the Bayesian framework, the *spike-and-slab Lasso (SSL)*, introduced by Ročková & George (2018), bridges penalised likelihood Lasso method with the traditional *spike-and-slab prior* approach, capitalising on the strengths of both while mitigating their drawbacks.

The package *'SSLASSO'* specifically implements *SSL*, which uses a dynamic penalty that adjusts based on the sparsity level and performs selective shrinkage. It also supports fast algorithms for finding the most probable estimates, ensuring efficiency and scalability. Lastly, debiasing the posterior modal estimate or applying effective posterior sampling techniques can quantify uncertainty for the *SSL*. The package primarily focuses on settings where $p > n$.

The methodology definition is based on Tadesse and Vannucci (2022). The spike-and-slab Lasso prior is given by:

$$\begin{aligned} \pi(\beta|\gamma) &= \prod_{i=1}^{p}[(1-\gamma_i)\psi(\beta_i|\lambda_0) + \gamma_i\psi(\beta_i|\lambda_1)], \\ \pi(\gamma|\theta) &= \prod_{i=1}^{p}[\theta^{\gamma_i}(1-\theta)^{1-\gamma_i}], \\ \theta &\sim Beta(a,b) \end{aligned} \tag{37}$$

where $\psi(\beta|\lambda) = (\lambda/2)e^{-\lambda|\beta|}$ denotes the Laplace density with scale parameter λ. As depicted in Figure 5, larger values of λ result in a density peaked around zero (the 'spike'), while smaller λ values lead to a diffuse density (the 'slab').

The original model assumed a known variance $\sigma^2 = 1$. Later works extended this to handle unknown variance, placing an independent Jeffreys prior on σ^2 where $p(\sigma^2) \propto \sigma^{-2}$.

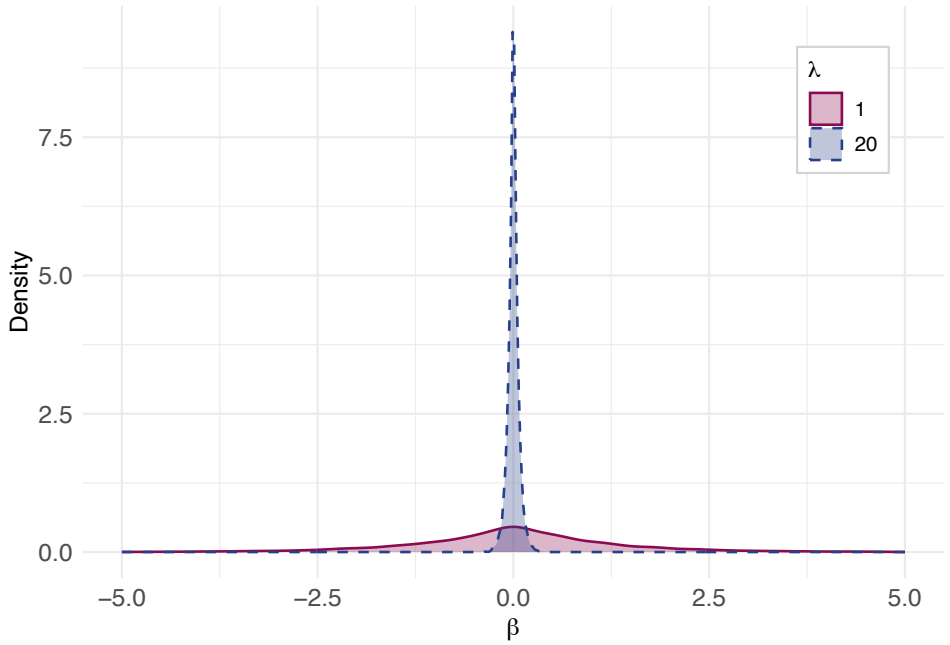

Figure 5: Density Plots of Laplace Distributions: Impact of Scale Parameter

Setting $\lambda_1 = \lambda_0$ results in the L1 penalty used in the Lasso. As $\lambda_0 \to \infty$, the spike-and-slab Lasso approaches the ideal point-mass model. Therefore, the SSL prior allows for a non-concave continuum between penalised likelihood and point-mass constructs.

The *SSL* prior, a mixture of two Laplace distributions, can be viewed as a two-group refinement of the L1 penalty in Lasso, leading to exactly sparse posterior modes for $p(\beta|y)$, enabling simultaneous variable selection and parameter estimation. This offers an advantage over traditional spike-and-slab formulations, which often require post hoc thresholding. Although the original Lasso is known to suffer from estimation bias, *SSL* offers two key advantages, as demonstrated by Tadesse and Vannucci (2022). First, it adaptively mixes two Lasso 'bias' terms, applying either high shrinkage for small $|\beta_i|$ or low shrinkage for large $|\beta_i|$. Unlike the adaptive Lasso, which assigns fixed penalties, *SSL* adjusts the coefficient-specific penalties to extremes. Second, the prior on θ introduces dependency in the marginal prior $p(\beta)$ and non-separability in the *SSL* penalty, enabling *SSL* to borrow information across coordinates and adapt to sparsity information. The *'SSLASSO'* package fits a set of models, each distinguished by the regularisation parameter λ_0, using a coordinate descent algorithm. This algorithm utilises screening rules to exclude irrelevant predictors, adopting a similar approach proposed by Breheny and Huang (2011). This thesis uses default settings to fit the *SSL* model.

5.3.4 Horseshoe Priors

The *horseshoe* prior was introduced by Carvalho, Polson, and Scott (2010), who characterised it as multivariate-normal scale mixtures. They modified the prior specification to set λ_i to be conditionally independent as further defined (Carvalho et al., 2010). In the context of a p-dimensional vector

$y|\theta \sim N(\theta, \sigma^2 I)$, when sparsity is assumed for β, the Bayesian *horseshoe* prior, denoted as π_{HS}, is applied. The assumption here is that each β_i is conditionally independent, each having a density of $\pi_{HS}(\beta_i|\tau)$. The *horseshoe* prior is then formulated as follows:

$$\begin{aligned} \beta_i|\lambda_i &\sim \mathcal{N}(0, \lambda_i^2), \quad \text{for } i = 1, \ldots, p, \\ \lambda_i|\tau &\sim \mathcal{C}^+(0, \tau), \\ \tau|\sigma &\sim \mathcal{C}^+(0, \sigma), \end{aligned} \tag{38}$$

where $\mathcal{N}$ is the normal distribution and $\mathcal{C}^+$ is the half-Cauchy distribution, specifically over the positive reals with a scale parameter denoted by a. It is vital to note that each β_i is a mixture of its own λ_i and that all λ_i elements have a half-Cauchy prior with a common scale, τ. The λ_i is referred to as the local shrinkage parameter and τ as the global shrinkage parameter. Additionally, Jeffreys' prior is employed for the variance, denoted by $p(\sigma^2) \propto 1/\sigma^2$, which is non-informative. Similarly, the prior for τ also follows Jeffreys' treatment, scaled by σ, the standard deviation of the error model.

The *horseshoe* prior enforces sparsity on the regression coefficients $\boldsymbol{\beta}$. Specifically, the posterior mean of each coefficient β_j can be expressed as a linear function of the corresponding observation y_i:

$$E(\beta_i|y_i) = y_i(1 - E(k_i|y_i)), \tag{39}$$

where k_i represents the shrinkage coefficient. The half-Cauchy prior on λ_i induces a $\text{Be}(\frac{1}{2}, \frac{1}{2})$ distribution for k_i, which has a *horseshoe* shape. For $k_i \approx 0$, there is negligible shrinkage representing signals, while for $k_i \approx 1$, there is substantial shrinkage representing noise.

The *horseshoe* prior has the property that it tends to shrink the majority of the coefficients β_j towards zero, enforcing sparsity, while allowing some coefficients to remain large if they are indeed associated with the response variable: its flat, Cauchy-like tails, ensures that significant signals maintain their magnitude, resulting in minimal post-hoc shrinkage. Simultaneously, its infinitely tall spike centred at the origin facilitates intense shrinkage for elements of β that are zero, effectively emphasising the sparse nature of the solution (Carvalho et al., 2010). A notable advantage of the *horseshoe* prior is that it does not require user-specified hyperparameters as the priors for λ_i, τ, and σ are fully defined. Figure 6 showcases the horseshoe prior with three different magnitudes of the global shrinkage parameter. As the plot shows, a smaller τ value tends to concentrate more mass around zero, leading to an amplified global shrinkage effect.

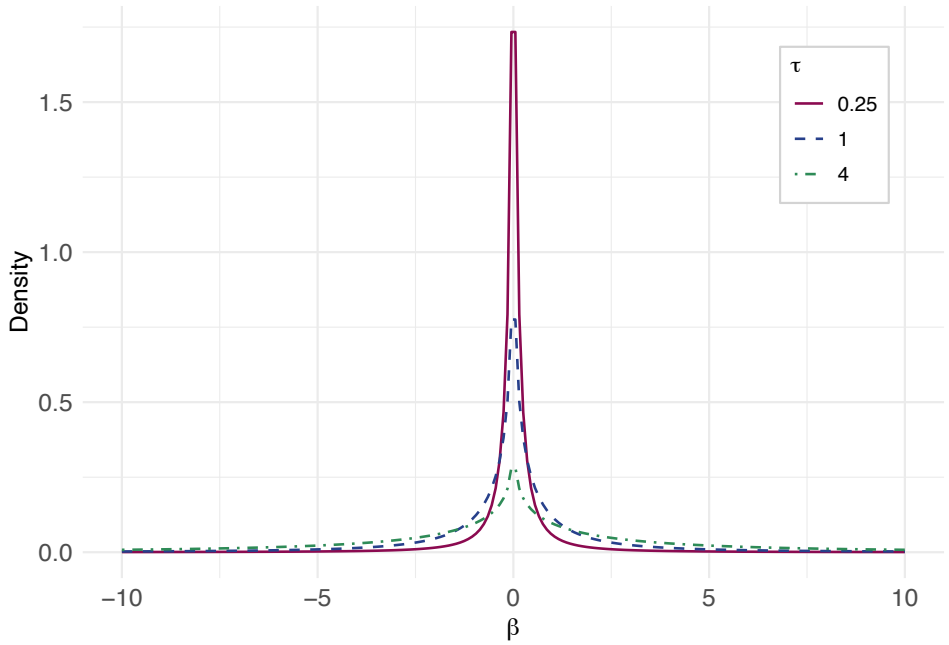

Figure 6: Density Plots of Horseshoe Distributions: Impact of tau

The *horseshoe+* estimator, an extension of the horseshoe estimator, excels in ultra-sparse problems (Bhadra et al., 2016). In contrast to the horseshoe estimator, the *horseshoe+* estimator has a lower posterior mean squared error and faster posterior concentration rates in terms of the Kullback–Leibler divergence metric. The prior distribution π_{HS+} for local shrinkage hyperparameters $(\lambda_1, ..., \lambda_p)$ retains the zero-mean half-Cauchy form and an additional layer of hyperparameters $(\eta_1, ..., \eta_p)$ is applied. Each η_i relates to the prior variance of the corresponding hyperparameter λ_i, creating an expanded hierarchy, as per Makalic and Schmidt (2016):

$$\begin{aligned} \beta_i|\lambda_i, \eta_i, \tau &\sim \mathcal{N}(0, \lambda_i^2), \\ \lambda_i|\eta_i, \tau &\sim \mathcal{C}^+(0, \tau\eta_i), \\ \eta_i &\sim \mathcal{C}^+(0, 1). \end{aligned} \tag{40}$$

In both horseshoe and *horseshoe+* models, the local shrinkage random effects λ_i are not marginally independent after the global shrinkage parameter τ is considered. The *horseshoe+* model further develops the concept by introducing an additional level of local shrinkage parameters η_i alongside τ, yielding conditionally independent λ_i. Integrating over η_i yields λ_i's density:

$$p(\lambda_i|\tau) = \frac{4\log(\lambda_i/\tau)}{\pi^2\tau(\lambda_i/\tau)^2 - 1}. \tag{41}$$

The introduction of the additional $\log(\lambda_i/\tau)$ term in the numerator leads to unique properties for the proposed estimator. Global shrinkage parameter τ can be handled in various ways. A full Bayesian

approach might involve assigning a standard half-Cauchy or Uniform(0,1) prior on τ. An alternative approach could appeal to an asymptotic argument, suggesting τ's empirical Bayes estimator be set to $\hat{\tau} = p_n/n$, where p_n is the count of non-zero entries in θ.

Figure 7 illustrates the density plots of *horseshoe* and *horseshoe+* prior distributions.

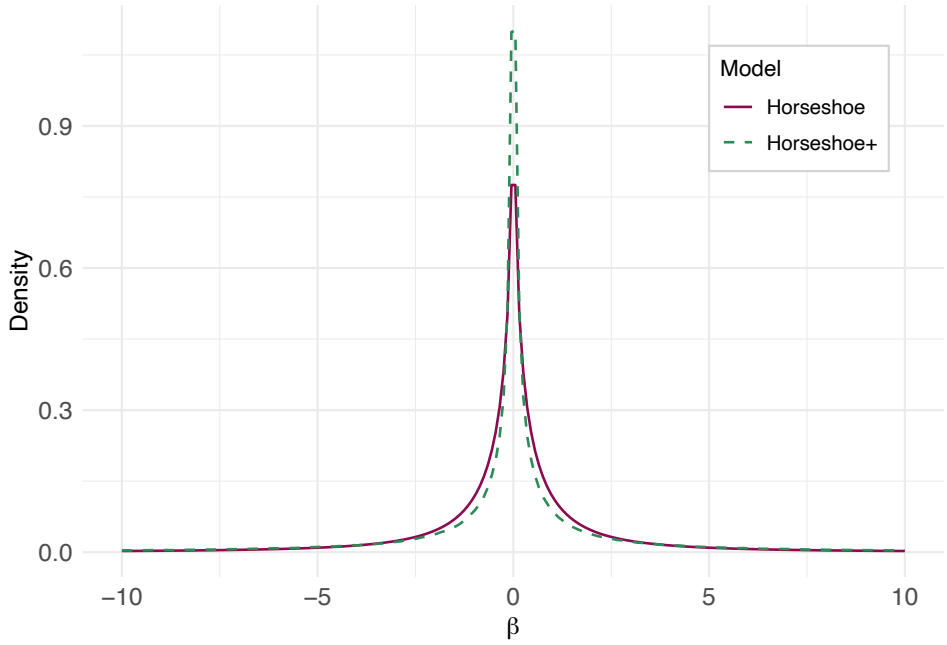

Figure 7: Density Plots of Horseshoe and Horseshoe+ Prior Distributions

This thesis explores *'bayesreg'* and *' horseshoe'* packages, both rooted in the work of Bhattacharya, Chakraborty, and Mallick (2016), each offering unique features.

The *'bayesreg'* package is tailored for linear or generalised linear regression models, leveraging Bayesian global-local shrinkage prior hierarchies as described by Polson and Scott (N. G. Polson & Scott, 2010). This thesis narrows its focus specifically on the *horseshoe* and *horseshoe+*, which are adept at handling high-dimensional datasets. A unique feature is its automatic grouping of factor variables, applying an added shrinkage to the corresponding dummy variables, thereby controlling model complexity. Additionally, it provides variable ranking, credible intervals, diagnostics, and the previously defined WAIC for prior selection. For efficient regression coefficient sampling, the package employs Gibbs sampling, choosing between Rue's algorithm when $p/n < 2$ (Rue, 2001) and Bhattacharya et al.'s method otherwise (Shin et al., 2017).

The *'horseshoe'* package allows for conducting sparse linear regression using the horseshoe prior, providing results such as posterior means and credible intervals. Depending on the predictor-to-observation ratio, it either adopts Bhattacharya et al.'s method for $p > n$ or Rue's approach otherwise (Rue, 2001). While it does not offer the horseshoe+ prior, it provides flexibility in handling the τ and error variance parameters. This thesis explores the two 'truncated Cauchy' and 'half Cauchy'

options for τ. Regarding the error variance σ^2 is set as 'Jeffreys'.

A conservative approach is adopted for both packages: an initial burn-in of 1,000 samples is discarded, followed by subsequent 5,000 samples. Other hyperparameters are maintained at their default settings.

5.3.5 Simplified Shotgun Stochastic Search Algorithm with Screening

The Shotgun Stochastic Search (SSS) algorithm, introduced by Hans et al. (2007), represents an attempt to navigate through extensive model spaces efficiently and pinpoint global maxima. Despite its effectiveness in exploring regions of high posterior model probability, the SSS algorithm is computationally intensive because it evaluates marginal probabilities at each interaction. Refer to Hans et al. (2007) for a detailed description. In pursuing a more contemporary variable selection method within the Bayesian framework, the *Simplified Shotgun Stochastic Search with Screening* (*S5*) developed by Shin, Bhattacharya and Johnson (2017) is presented here. *S5* is a more recently adapted version of the SSS algorithm and aims to address the computational challenges of its predecessor.

A specific scenario is considered with a response and p candidate predictors in the previously defined linear regression model context. A particular model is denoted by $k = k_1, ..., k_{|k|}$, with $1 \leq k_1 < ... < k_{|k|} \leq p$. For each model k, the design matrix and the corresponding regression coefficient are represented as X_k and β_k, respectively. The true model, denoted by t, is assumed to be fixed but unknown. The regression model under each k is given by $y = X_k\beta_k + \epsilon$, where $\epsilon \sim \mathcal{N}_n(0, \sigma^2 I_n)$.

In this thesis, almost all default settings of the package *'BayesS5'* are maintained, which include the nonlocal product inverse-moment (piMoM) prior. Given a model k, the prior density for the vector of regression coefficients β_k is defined as:

$$\pi(\beta_k|\sigma^2, \tau, k) = C^{*-|k|} \prod_{j=1}^{|k|} [(\beta_{k,j})^{-2r} \exp(-\tau/\beta_{k,j}^2)], \tag{42}$$

where $C^* = \tau^{(-r+1/2)}\Gamma(r - 1/2)$ for $r > 1/2$, and $\Gamma(\cdot)$ is the gamma function.

As illustrated in Figure 8, the piMoM prior densities are characterised as nonlocal, meaning the density value at the origin is exactly zero. A defining feature of nonlocal priors is that in the above formula, $\pi(\beta_k) = 0$ when $\beta_k = 0$.

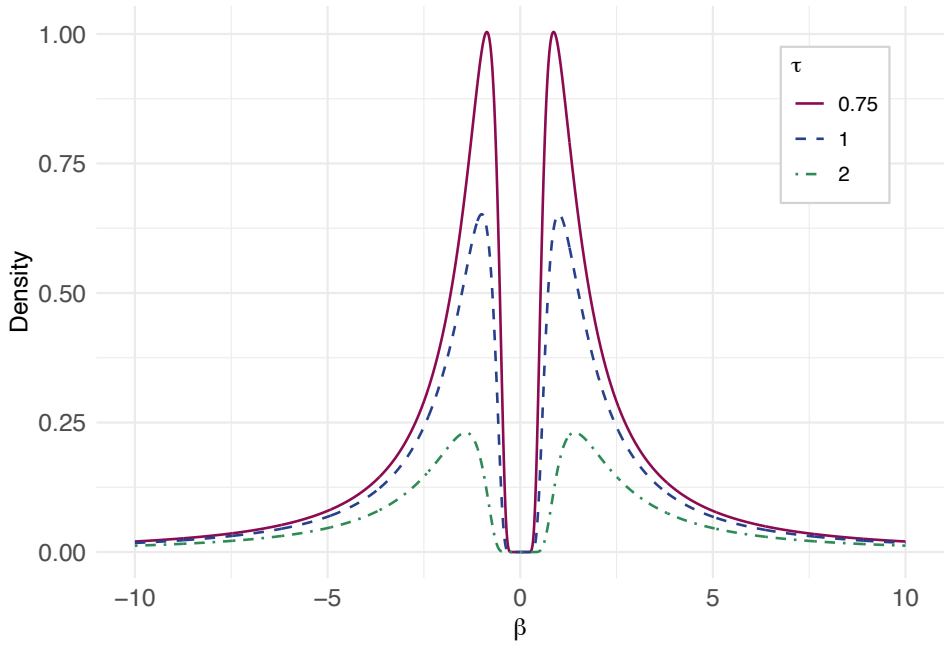

Figure 8: Density Plot of piMoM Prior for Different tau Values

The piMoM prior densities have inverse polynomial tails. A prior is also imposed on the space of models and the priors on the regression parameters given a model to complete the prior specification. A uniform prior is considered on the model space restricted to models having a size less than or equal to q_n, with $q_n < n$, that is, $\pi(k) \propto I(|k| \leq q_n)$, where $I(\cdot)$ denotes the indicator function. With a slight abuse of notation, the prior on the space of models is also denoted by π.

Keeping in mind that in $p \gg n$ settings, full posterior sampling using MCMC algorithms is highly inefficient, Shin et al. (2017) proposed a scalable stochastic search algorithm aimed at swiftly identifying regions of high posterior probability and finding the maximum a posterior (MAP) model.

The MAP model, denoted $\hat{k}$, is formally defined as:

$$\hat{k} = \underset{k \in \Gamma^*}{\arg\max} \{\pi(k|y)\}, \tag{43}$$

where Γ^* represents the set of models that are assigned non-zero prior probability. Shin et al. (2017) introduced enhancements to the efficiency of the SSS algorithm. Yet, this increased efficiency may make the algorithm less likely to explore 'interesting' regions of high posterior model probability, thereby increasing the likelihood of getting trapped in local maxima. To mitigate this issue, the algorithm incorporates sparsity-inducing priors to promote parsimony and employs temperature control, similar to the approach used in global optimisation algorithms such as simulated annealing (Kirkpatrick et al., 1983). To further alleviate the computational load, the algorithm adopts strategies from Iterative Sure Independence Screening (J. Fan & Lv, 2008), focusing only on variables that exhibit a strong correlation with the residuals of the current model.

Shin et al. (2017) established that the *S5* algorithm outperforms the SSS algorithm regarding speed when identifying the MAP model, requiring fewer model evaluations. Furthermore, they demonstrated that model selection procedures employing the piMoM prior attain robust model selection consistency in $p \gg n$ scenarios. However, as the ratio of predictors to data entries rises, method performance diminishes with a tendency to overlook the true signals, disadvantage observed by Xu (2021).

In the analysis presented in this thesis, default settings are predominantly retained. The model prior is set to 'Bernoulli uniform', and the log-marginal likelihood function is based on pre-specified priors for the regression coefficients, with 'piMoM' being the default choice. While the S5 algorithm is typically run twice by default, it is executed five times for this study. Additionally, the variance selection remains at its standard setting of 'fixed'.

6 Simulated Data Study

6.1 Simulation Overview

Four data types are simulated to create a versatile, controlled environment to test the defined methods. Each *Type 1*, *Type 2*, *Type 3*, and *Type 4* datasets are designed to be adaptable across various dimensionality settings. Importantly, the creation of true signals, the predetermined magnitude of error variance, and the inclusion of interaction terms, polynomials, and other features are strategically chosen to present varying levels of complexity for the tested models, thereby examining their robustness and adaptability under distinct circumstances.

The *Type 1* datasets consist of uncorrelated continuous covariates with a moderate noise level. The covariates are simulated from a multivariate normal distribution:

$$\mathbf{X} \sim \text{MVN}(\mathbf{u}_x, \sigma_x^2 \mathbf{\Sigma}_x), \tag{44}$$

where $\mathbf{u}_x$ is a $1 \times p$ mean vector, $\mathbf{\Sigma}_x$ is a $p \times p$ correlation matrix, and σ_x^2 is the common variance of all covariates. Each x_i is normally distributed with a mean of 5, that is, $\mathbf{u}_x = (5, 5, \ldots, 5)$. $\mathbf{\Sigma}_x$ is a $p \times p$ identity matrix and $\sigma_x^2 = 1$. The response variable y is generated as:

$$\mathbf{y} = \mathbf{X}\boldsymbol{\beta} + \boldsymbol{\epsilon}, \tag{45}$$

where $\boldsymbol{\epsilon_i} \sim N(0, \sigma_e^2)$ for $i = 1, 2, \ldots, n$. $\boldsymbol{\beta}$ is a $p \times 1$ vector of true coefficients, $\boldsymbol{\epsilon}$ is an $n \times 1$ vector of error terms, σ_e^2 is the error variance, and $\mathbf{I_n}$ is an $n \times n$ identity matrix. The regression coefficients are enforced as $\beta_1, \ldots, \beta_{10} = 3$; $\beta_{11}, \ldots, \beta_{20} = 5$, and $\beta_{p-20} = 0$. The intercept term is zero, and the errors (unexplained variability in the response variable) are normally distributed with $\sigma_e^2 = 15$.

The *Type 2* dataset comprises continuous covariates with temporal correlation and moderate noise. The generation of the *Type 2* dataset parallels the method utilised for the Type 1 dataset with certain distinctions. Specifically, the mean vector for the covariates $\mathbf{u}_x$ is constructed such that the first 30 variables each have a mean of 3, and the rest have a mean of 7; that is, $\mathbf{u}_x = (3, 3, \ldots, 3, 7, 7, \ldots, 7)$. The covariance matrix of the covariates $\mathbf{\Sigma}_x$ adheres to an autoregressive order 1 (AR(1)) structure, with the correlation coefficient $\rho = 0.8$, σ_x^2 is held constant at 1. For the response variable, 20 covariates are true signals. The vector of true regression coefficients, $\boldsymbol{\beta}$, is defined with non-zero values for the first 20 entries, while the remaining entries are set to zero;

specifically, $\boldsymbol{\beta} = (1, 2, 3, \ldots, 20, 0, \ldots, 0)^T$. The error term variance $\sigma_e^2 = 10$.

The *Type 3* dataset is a blend of continuous and categorical covariates, including interaction terms and polynomial features, with moderate noise. In this setup, the mean vector for continuous covariates $\mathbf{u}_x$ is segmented into three groups: the first 20 variables have a mean of 2, the next 30 have a mean of 5, and the remaining variables have a mean of 8, resulting in $\mathbf{u}_x = (2, 2, \ldots, 2, 5, 5, \ldots, 5, 8, 8, \ldots, 8)$. The covariance matrix $\boldsymbol{\Sigma}_x$ adheres to an AR(1) structure with a $\rho = 0.6$ correlation coefficient, while σ_x^2 remains constant at 1. The vector of true regression coefficients for continuous predictors, $\boldsymbol{\beta}$, takes on values such as $\boldsymbol{\beta} = (6, 6, 6, 6, 6, 4, 4, 4, 4, 4, 3, 3, 3, 3, 3, 0, 0, \ldots, 0)^T$.

The first categorical variable is binary (akin to having or not having an illness) and is set to $\beta = 4$. The second categorical variable is treated as ordinal, with categories 1 through 5 (serves as ordered category, for example, progressive education levels from middle school to a higher degree) and is set to have $\beta = 0$. Interaction terms are created by multiplying selected pairs of covariates: $X1$ and $X2$ continuous, $X3$ and $X4$ continuous, $X11$ and $X22$ continuous, *binarycategorical* and $X22$ continuous. Polynomial features are generated by elevating $X5$ and $X23$ covariates to the power of 2, and $X6$, $X23$ to the power of 3. From interaction terms and polynomials, only $X1 : X2$ is enforced to have $\beta = 3$ and $X23^2$ to have $\beta = 6$, the rest have $\boldsymbol{\beta} = (0, \ldots, 0)$. Lastly, the intercept has been set to a value of 2; the error term variance is $\sigma_e^2 = 12$.

The *Type 4* dataset is characterised by grouping structures among continuous covariates, where covariates within each group are highly correlated, while covariates between different groups are independent. The mean vector for continuous covariates $\mathbf{u}_x$ is segmented into five groups, that is $\mathbf{u}_x = (2, \ldots, 2, 4, \ldots, 4, 6, \ldots, 6, 8 \ldots, 8, 10, \ldots, 10)$. The vector of true regression coefficients corresponding to true signals within each group $\boldsymbol{\beta} = (6, 6, 6, 6, 6, 4, 4, 4, 4, 4, 3, 3, 3, 3, 3, 0, \ldots, 0)^T$, where the number of groups $T = 5$.

The covariance matrix $\boldsymbol{\Sigma}_x$ is constructed as a block-diagonal matrix, where each block corresponds to a group, and within each block, the elements are highly correlated. The diagonal blocks can have a specific structure, the AR(1), with correlation coefficients $\rho = 0.6$, while the off-diagonal blocks are matrices of zeros, indicating independence between groups.

The first categorical variable is binary and is set to have $\beta = 4$. The second categorical variable is treated as ordinal, with categories 1 through 5 and set to $\beta = 0$. Interaction terms are created by multiplying selected covariates: the continuous $X1$, $X2$ and $X3$, $X3$ and $X4$, $X16$ and $X17$. $X1 : X2 : X3$ is enforced to have $\beta = 3$, $X4 : X5$ and $X16 : X17$ to have $\beta = 0$. The error term variance is set at $\sigma_e^2 = 10$.

Four different dimensionality settings are considered under all four data types: $p(50) < n(200)$, reflecting a traditional setting with more observations than variables; $p(100) = n(100)$, representing a balanced case; $p(200) > n(150)$, indicative of high-dimensional scenarios such as in genomics; and $p(200) \gg n(50)$, where the number of variables substantially surpasses the number of observations. An additional configuration is proposed to suit XGBoost's strengths better and establish a baseline performance capacity. While retaining the data generation methodology outlined earlier, a subsequent dataset is simulated with a modified proportion of data points to features, namely $p = 50, n = 500$. These settings enable comparisons within data types and offer insights into practical challenges regarding data availability and computation. The choice of p and n values is guided by computation time and an approximation to real-world data, where obtaining data can be costly. This structure facilitates a comprehensive analysis of various scenarios. For reference, for all data generation, the reproducibility seed is set to 42.

6.2 Standardisation

The handling and standardisation of data can vary depending on the statistical packages used in the analysis. These packages often require data to be presented in specific formats, and the treatment of categorical variables can differ substantially. For instance, packages *'glmnet'*, *'caret'*, *'spikeslab'*, *'horseshoe'*, *'monomvn'*, and *'BayesS5'* require data to be presented in a matrix form. In this matrix, each column signifies a variable from the dataset, while each row represents an observation. If a dataset only comprises continuous variables, it can be directly converted into a matrix. However, when categorical variables are present, they must be transformed. The resulting matrix includes continuous and dummy variables, each representing different levels of the categorical variables. The intercept absorbs level one of each categorical variable to prevent the dummy variable trap, which could lead to multicollinearity. On the other hand, the *'bayesreg'* package favours data in a data frame format. In this case, categorical variables are kept in their original state as factors to integrate into the model adequately.

Standardisation is often adopted to mitigate multicollinearity and enhance coefficient interpretability (Olvera Astivia & Kroc, 2018). It is applied to continuous predictors in the simulated data. However, package-specific guidelines are crucial to prevent the inadvertent standardisation of categorical variables. Most of the packages used in the thesis, except *'SSLASSO'*, either exclude standardisation by default or allow its disablement; this ensures proper treatment of categorical variables. Therefore, the *SSL* method from *'SSLASSO'* package is not used on datasets containing categorical variables. The *'BayesS5'* package could not be used on data with categorical variables either, as it does not include the use of an intercept; hence, the categorical variables could not be encoded. While the *XGBoost* method typically exhibits scale insensitivity, standardisation of all continuous data is maintained for model consistency.

6.3 Results

Three metrics are used to evaluate the performance of the introduced methodologies for the variable selection task: total true signals (TS), false positives (FP), and false negatives (FN). TS is the count of covariates chosen by the method, FP quantifies the noise covariates wrongly identified as signals, while FN represents the missed true signal covariates.

6.3.1 Type 1

The penalised regression methods perform well in selecting TS except in scenarios where $p \gg n$. However, a significant number of FP are identified across all datasets. In the $p < n$ setting, FP coefficients are less than 0.6 in value, diminishing from 2.6 for TS. FP coefficients remain high for $p = n$ and $p > n$, comparable to the TS coefficients. For the $p = n$ setting, *Lasso's* FP coefficients range approximately from 0.01 to 1.2, while TS coefficients range from 2 to 6. In the same dimensionality, *elastic-net's* FP coefficients range approximately from 0.01 to 0.5, with TS coefficients ranging from around 2 to 5.4. In the $p > n$ setting, *Lasso's* FP coefficients range approximately from 0.01 to 0.52, with TS coefficients ranging from 2 to 5.4. In this setting, *elastic-net* exhibits FP coefficients ranging from 0.002 to 0.5, with TS coefficients ranging from 1.9 to 5.3. In the $p \gg n$ setting, both methods display a mix of coefficient values, ranging from 0.05 to just below 5, indicating a more complex variable selection scenario. With the *spike-and-slab* prior, all TP are identified for $p < n$ and $p > n$. Notably, $p < n$ yields many FP, while $p \gg n$ selects just 1 TP.

The evaluation of *SSL* performance presents a challenge due to the wide range of selected coefficients, from supposedly negligible values of 10^{-10} to 20 in *Type 1* data. The method performs excellently if these negligible values are excluded from the variable selection. However, when included, almost all variables are selected, indicating little variable choice. Consequently, this thesis presents two sets of results: (1) variables with coefficients more significant than 10^{-10}, and (2) all selected variables, distinguished in Table 1 by $'/'$. The variable selection is executed flawlessly for *Type 1* data in all but $p \gg n$ dimension settings. In the $p \gg n$ scenario, all coefficients are close to 0. Even when these negligible contributions are considered, numerous FP and FN are included.

Both *'horseshoe'* package methods with 'Truncated Cauchy' and 'Half Cauchy' priors perform well in all but $p \gg n$ settings, with only 1 FP variable identified in the $p = n$ setting. For $p \gg n$, hs TC identifies only 1 TS, while hs HC selects no variables. Both *'bayesreg'* package methods with *horseshoe* and *horseshoe+* priors identify all TS, except for the $p \gg n$ setting. Specifically, the *horseshoe* method identifies all TS, with only one FP in the $p = n$ setting, and selects only 2 TS in the $p \gg n$ scenario. The *horseshoe+* method identifies all TS in all settings except for the $p \gg n$ scenario, which selects only 3 TS.

For the *S5* method, several C0 values, the number of repetitions of S5 algorithms, were tested, with no improvement observed for values above 5. The method performs exceptionally well in all but $p \gg n$ situations. In the $p \gg n$ situation, only 1 TS is selected. *Bayesian Lasso* performs well

detecting TS but includes FP in all but $p < n$ dimensions. In the $p = n$ and $p > n$ dimensions, the TS inclusion probabilities are around 1, while FP probabilities decrease. In the $p \gg n$ dimension, only 6 TS are selected with varying probabilities for TS and FP. See Table 1 for results.

Table 1: Summary of Type 1 Simulated Data Results

		p < n			p = n			p > n			p » n		
Package	**Method**	**TS**	**FP**	**FN**	**TS**	**FP**	**FN**	**TS**	**FP**	**FN**	**TS**	**FP**	**FN**
Frequentist Methods													
glmnet	Lasso	20	19	0	20	37	0	20	41	0	9	23	11
glmnet	Elastic-Net	20	19	0	20	52	0	20	43	0	12	29	8
Bayesian Methods													
spikeslab	Spike-and-Slab Prior	20	19	0	13	1	7	20	2	0	1	0	19
SSLASSO	Spike-and-Slab Lasso	20/20	0/18	0/0	20/20	0/62	0/0	20/20	0/144	0/0	0/12	0/30	20/8
horseshoe	Horseshoe Prior, TC	20	0	0	20	1	0	20	0	0	1	0	19
horseshoe	Horseshoe Prior, HC	20	0	0	20	1	0	20	0	0	0	0	20
bayesreg	Horseshoe Prior	20	0	0	20	1	0	20	0	0	2	0	18
bayesreg	Horseshoe+ Prior	20	0	0	20	0	0	20	0	0	3	0	17
BayesS5	S5 Method	20	0	0	20	0	0	20	0	0	1	0	19
monomvn	Bayesian Lasso	20	0	0	20	7	0	20	20	0	6	2	14

[a] TS=True Signal, FP=False Positive, FN=False Negative

[b] p(50) < n(200), p(100) = n(100), p(200) > n(150), p(200) » n(50)

[c] Spike-and-Slab Lasso is presented with two sets of results separated by '/'

6.3.2 Type 2

For the $p < n$ setting, both *Lasso* and *elastic-net's* TS coefficients approximately range from 0.1 to 19.3. When $p = n$, the coefficients for both methods hover around 1.8 to 18.9. In scenarios where $p > n$ or $p \gg n$, the coefficients generally span from 0.3 to 19.6. These estimates roughly align with the enforced β coefficients.

Spike-and-slab prior method performs reasonably well in all but $p \gg n$ settings, where only 3 TP are identified. *SSL* does not identify any FP. However, including coefficients with negligible values would improve the selection of TS. Both *'horseshoe'* package methods perform relatively well, with only 1 FP identified in the $p < n$ and $p = n$ settings. In the $p \gg n$ setting, the 'truncated Cauchy' prior method identifies only 3 TS, while the 'half Cauchy' prior method identifies 13 signals. Both *horseshoe* and *horseshoe+* prior methods perform well, with *horseshoe* identifying 1 FP in the $p = n$ setting and 1 FN in the $p = n$ and $p > n$ settings. However, only 5 TS are selected in the $p \gg n$ setting. The *horseshoe+* method identifies 2 FN in the $p = n$ and $p > n$ settings and only 7 TS in the $p \gg n$ setting. *S5* performs relatively well, misidentifying 3, 5, and 6 signals as FN in the respective dimensions. In the $p \gg n$ setting, only 3 TS are selected. In all but $p \gg n$ settings, *Bayesian Lasso* approximates the inclusion probabilities for TS at around 1 and then slightly lower for FP. In the $p \gg n$ setting, TS probabilities vary from 0.5 to 1. Refer to Table 2 for results.

Table 2: Summary of Type 2 Simulated Data Results

		p < n			p = n			p > n			p » n		
Package	**Method**	**TS**	**FP**	**FN**	**TS**	**FP**	**FN**	**TS**	**FP**	**FN**	**TS**	**FP**	**FN**
Frequentist Methods													
glmnet	Lasso	20	0	0	19	0	1	20	0	0	20	0	0
glmnet	Elastic-Net	20	0	0	19	0	1	20	0	0	20	0	0
Bayesian Methods													
spikeslab	Spike-and-Slab Prior	20	4	0	18	0	2	18	0	2	3	1	17
SSLASSO	Spike-and-Slab Lasso	10/19	0/0	10/1	7/15	0/0	13/5	5/15	0/0	15/5	0/6	0/0	20/14
horseshoe	Horseshoe Prior, TC	20	1	0	19	1	1	18	0	2	3	0	17
horseshoe	Horseshoe Prior, HC	20	1	0	19	1	1	18	0	2	13	0	7
bayesreg	Horseshoe Prior	20	0	0	19	1	1	19	0	1	5	0	15
bayesreg	Horseshoe+ Prior	20	0	0	18	0	2	18	0	2	7	0	13
BayesS5	S5 Method	17	0	3	15	0	5	14	0	6	3	0	17
monomvn	Bayesian Lasso	20	0	0	20	3	0	20	3	0	18	0	2

[a] TS=True Signal, FP=False Positive, FN=False Negative

[b] p(50) < n(200), p(100) = n(100), p(200) > n(150), p(200) » n(50)

[c] Spike-and-Slab Lasso is presented with two sets of results separated by '/'

6.3.3 Type 3

In the $p < n$ setting, *Lasso's* TS coefficients range from 2.4 to 57.9, with FP ranging from 0.02 to 0.6. *Elastic-net's* coefficients for TS and FP intertwine in this setting, indicating some FP are not sufficiently penalised. This intertwining of TS and FP coefficients persists for both methods in the $p = n$ and $p > n$ settings. In the $p > n$ setting, *Lasso's* TS coefficients range from 2 to 51.6, with FP ranging from 0.1 to 1.9. In the $p \gg n$ setting, both methods fail to pick up the binary categorical variable signal.

Spike-and-slab prior method performance deteriorates, identifying quite a few FP, binary variable is not selected in all but $p < n$. Both *'horseshoe'* package methods with 'Truncated Cauchy' and 'Half Cauchy' prior methods yield identical results, with perfect TS selection in all but $p \gg n$ settings, where only 6 TS are selected. Both *'bayesreg'* package method with *horseshoe* prior performs well with 1 FN in the $p = n$ setting and only 5 TS in the $p \gg n$ setting, along with 1 FP. The *horseshoe+* prior method performs flawlessly in all but $p \gg n$ dimensions, where only 5 TS are selected. All methods utilising *horseshoe* prior place great importance on interaction and polynomial terms in $p \gg n$. *Bayesian Lasso* for all but $p \gg n$ settings computes TS signals' probabilities of around 1, with FP probabilities dipping lower. Significant probabilities are incorrectly assigned to polynomial and interaction terms. The method performs poorly in the $p \gg n$ situation, including 4 FP and 10 FN. A summary of the results can be found in Table 3.

Table 3: Summary of Type 3 Simulated Data Results

		p < n			p = n			p > n			p » n		
Package	**Method**	**TS**	**FP**	**FN**	**TS**	**FP**	**FN**	**TS**	**FP**	**FN**	**TS**	**FP**	**FN**
Frequentist Methods													
glmnet	Lasso	18	13	0	17	9	1	18	10	0	16	14	2
glmnet	Elastic-Net	17	12	1	17	12	1	18	9	0	16	16	2
Bayesian Methods													
spikeslab	Spike-and-Slab Prior	18	18	0	11	4	7	16	5	2	4	3	14
horseshoe	Horseshoe Prior, TC	18	0	0	18	0	0	18	0	0	6	0	12
horseshoe	Horseshoe Prior, HC	18	0	0	18	0	0	18	0	0	6	0	12
bayesreg	Horseshoe Prior	18	0	0	17	0	1	18	0	0	5	1	13
bayesreg	Horseshoe+ Prior	18	0	0	18	0	0	18	0	0	5	0	13
monomvn	Bayesian Lasso	18	0	0	18	4	0	18	8	0	8	4	10

[a] TS=True Signal, FP=False Positive, FN=False Negative

[b] p(50) < n(200), p(100) = n(100), p(200) > n(150), p(200) » n(50)

6.3.4 Type 4

In the $p < n$ setting, *Lasso's* TS coefficients range from 2.2 to 40.1, with FP ranging from 0.004 to 0.4. In this dimensionality, *Elastic-net's* TS coefficients range from 2.4 to 39.6, with FP ranging from 0.04 to 0.8. For both methods, the coefficients for TS and FP intertwine in the $p = n$ setting, indicating that some FP are not sufficiently penalised. In the $p > n$ setting, *Lasso's* TS coefficients range from 2.3 to 45, with FP ranging from 0.06 to 0.6. Again, *Elastic-net's* coefficients for TS and FP intertwine in this setting. In the $p \gg n$ setting, both methods fail to pick up the binary categorical variable signal.

Spike-and-slab method identifies quite a few FP. The binary variable is incorrectly identified as FN in all but $p < n$. Both *'horseshoe'* and *'bayesreg'* package methods ideally and yield identical results, with perfect TS selection in all but $p \gg n$ settings, where only 2 TS are selected with great importance placed on the interaction term, namely $X1 : X2 : X3$. In all but $p \gg n$ settings, the *Bayesian Lasso* approximates the TS probabilities to around 1, with FP probabilities dipping lower. In the $p > n$ setting, 11 FP are identified. The method performs poorly in the $p \gg n$ setting, with 10 TS and 3 FP. In contrast to Type 3 data analysis, the categorical and interaction terms with enforced $\beta = 0$ are correctly identified as unimportant. Table 4 provides a summary of the results.

Table 4: Summary of Type 4 Simulated Data Results

		p < n			p = n			p > n			p » n		
Package	**Method**	**TS**	**FP**	**FN**	**TS**	**FP**	**FN**	**TS**	**FP**	**FN**	**TS**	**FP**	**FN**
Frequentist Methods													
glmnet	Lasso	17	9	0	17	3	0	16	7	1	16	9	1
glmnet	Elastic-Net	17	10	0	17	2	0	16	12	1	15	13	2
Bayesian Methods													
spikeslab	Spike-and-Slab Prior	17	12	0	13	3	4	15	4	2	15	13	2
horseshoe	Horseshoe Prior, TC	17	0	0	17	0	0	17	0	0	2	0	15
horseshoe	Horseshoe Prior, HC	17	0	0	17	0	0	17	0	0	2	0	15
bayesreg	Horseshoe Prior	17	0	0	17	0	0	17	0	0	2	0	15
bayesreg	Horseshoe+ Prior	17	0	0	17	0	0	17	0	0	2	0	15
monomvn	Bayesian Lasso	17	0	0	17	2	0	17	11	0	10	3	7

[a] TS=True Signal, FP=False Positive, FN=False Negative

[b] p(50) < n(200), p(100) = n(100), p(200) > n(150), p(200) » n(50)

6.3.5 XGBoost

Without an 80% aggregation threshold for total gain, the *XGBoost* method often identifies most or all variables as significant contributors. Table 5 presents the *XGBoost* outcomes for all simulated data. The results exhibit inconsistent TS, FP, and FN detection across different dimensionalities. The results were closest to satisfactory when analysing data with only continuous covariates, excluding categorical, polynomial, and interaction variables, and without enforced covariance. The results do not show marked improvement even in the ostensibly favourable $p \ll n$ context.

Table 5: Summary of XGBoost Results for All Simulated Data

	p « n			p < n			p = n			p > n			p » n		
Data	**TS**	**FP**	**FN**	**TS**	**FP**	**FN**	**TS**	**FP**	**FN**	**TS**	**FP**	**FN**	**TS**	**FP**	**FN**
Type 1	15	3	5	11	20	9	18	23	2	8	14	12	15	0	5
Type 2	9	0	11	8	0	12	9	0	11	7	0	13	10	0	10
Type 3	3	2	11	10	8	8	2	2	16	3	2	15	2	7	16
Type 4	1	0	16	1	0	16	1	1	16	1	0	16	3	2	14

[a] TS=True Signal, FP=False Positive, FN=False Negative

[b] p(50) « n(500), p(50) < n(200), p(100) = n(100), p(200) > n(150), p(200) » n(50)

6.4 Discussion

The *Lasso* and *elastic-net* rely on mean point estimates, the *spike-and-slab* prior method relies on point estimates of generalised elastic-net (gnet), and variables are only included when the resulting coefficient is not precisely zero. *SSL* again hinges on point estimates; this thesis delineates two distinct approaches: an automatic variable selection and a manual selection that omits negligible coefficients. In uncorrelated data, disregarding negligible coefficients yields favourable *SSL* performance, except in $p \gg n$ cases. The performance of *SSL* is much poorer in Type 2 than Type 1 data, indicating that it struggles when temporal correlation is present. Only when the automatically selected negligible coefficients are included does it come close to satisfactory performance in Type 2 data.

The frequentist methods excel in the Type 2 setting, characterised by continuous variables with AR(1) correlation structure. These methods retain their robustness even when beta coefficients are enforced with varying magnitudes (1 to 20) in $p \gg n$ contexts. Yet, they tend to accrue numerous FP in alternative settings. This inclination towards FP is likely attributed to coefficients nearing, but not equating to, zero. Employing confidence or credible interval estimates might counteract this trend by excluding the corresponding covariate. Notably, the *spike-and-slab* formulation predominantly reduces the gnet coefficients to absolute values below 1, contrasting with the considerably elevated TS values. When faced with polynomials and interaction terms, their gnet coefficients are substantially larger than those of continuous variables. Even when these terms are false positives, they still indicate a pronounced relationship in the final model. At the same time, the binary variable is mostly overlooked even when it should be included. *Spike-and-slab's* performance in $p \gg n$ scenarios is notably subpar - a limitation echoed across all methods. Methods with point estimates might be more sensitive to small changes in the data, which can lead to variability in the selected predictors.

XGBoost's performance in linear regression tasks is suboptimal, marked by numerous FNs and FPs, aligning with expectations outlined in the methodology section; this is more extensively discussed in the Conclusions chapter.

Methods from the *'bayesreg'* and *'horseshoe'* packages rely on credible intervals to evaluate features, providing a measure of uncertainty around parameter estimates. Given the continuous nature of their sparsity-inducing priors, they effectively assign a posterior probability of one to the full model, bypassing the conventional Bayesian model inclusion probabilities. They demonstrated superior performance in most scenarios except for the extreme high-dimensional case $p \gg n$. This observation aligns with Carvalho, Polson, and Scott (2010), who highlighted the horseshoe prior's edge over the classical *Lasso* approach, suggesting it as a reliable default choice for drawing inference. The consistent performance of these methods across varied data situations underscores their adaptability and resilience to different modelling challenges. However, the method sometimes excludes the weaker signals; here, over-shrinkage in Bayesian credible sets arises when data-driven sparsity estimates produce intervals too narrow and near zero (Pas et al., 2017). This can stem from overly large bandwidth choices, excessive smoothness, or a posterior concentrated too close to zero, compromising

the coverage of the true parameter. In higher dimensions, adjusting the regularisation strength of the horseshoe prior might be beneficial, especially given that most results were great except in the $p \gg n$ scenarios, where true signals were overlooked.

The *Bayesian Lasso* and *S5* methodologies adopt a fully Bayesian approach for their computations, wherein the dimensionality of the model is treated as a random variable. This approach inherently weaves model uncertainty into the feature selection process, offering a probabilistic perspective on the significance of each predictor. Specifically, they provide an estimated posterior probability that individual components of the regression coefficients are non-zero. However, a notable limitation emerges as the number of predictors increases: the performance of these methods tends to decline. Within the *Bayesian Lasso* framework, FP that are included typically manifests with slightly diminished inclusion probabilities, especially when compared to TS, where the probability consistently approaches or is precisely one. This method performs as expected, weak signals are not drawn to precisely zero. *S5* excels by avoiding FP altogether but missing weaker signals, as anticipated. While the TP are identified with probabilities nearing or equal to 1, the method adeptly identifies false positives with probabilities below 0.07. However, its performance diminishes in scenarios where $p \gg n$.

Given the medium to high noise levels introduced, it is understandable that all methods face challenges in accurately selecting variables when confronted with the extreme $p \gg n$ setting.

7 Crime Data Study

Analysing simulated data provides foundational insights into methodologies, but examining a real dataset is vital as it presents with more complex challenges.

The dataset under study amalgamates socio-economic data from the 1990 Census, law enforcement data from the 1990 Law Enforcement Management and Admin Stats survey, and crime data from the 1995 Federal Bureau of Investigation Uniform Crime Reports (FBI UCR) for various communities in the U.S., that is, each in the data row represents a respective community. Comprising 2215 instances and 147 attributes, the dataset includes factors related to community characteristics, law enforcement, and crime rates, focusing on 125 predictive attributes and 18 potential target variables (crime attributes). It is essential to recognise that the dataset has limitations due to discrepancies in population values, the omission of some communities, and the absence of certain data. The FBI cautions against using this dataset as the sole criterion for evaluating communities, as it does not encompass all relevant factors. The dataset was donated to the UC Irvine Machine Learning Repository, for a more comprehensive understanding, the reader is referred to the repository (Redmond, 2011).

The non-predictive variables are disregarded, such as community names. Additionally, the analysis disregards variables with over 80% missing values to maintain data integrity and reliable outputs, communities with missing data are also eliminated. The wrangled data consist of 101 variables: 100 predictors and the target variable 'Violent Crimes per 100k Population' with 1992 instances. For a comprehensive list of available and included variables, refer to Appendix Table 13.

7.1 Ethical Considerations

Algorithmic decision-making mechanisms permeate many sectors of modern life, from email spam classification to credit scoring and employment candidate assessment. However, concerns have emerged regarding transparency, accountability, and fairness, specifically when these systems predicate their decisions on historical data. There exists a risk of perpetuating biases against certain demographic groups identified by 'sensitive attributes', such as gender, age, race, or religion, should these groups have been historically correlated with higher risk factors (Komiyama et al., 2018). Such variables refer to data that could be used to predict attributes protected by anti-discrimination laws, where the prejudiced actions are directed towards individuals based on their membership in certain groups rather than assessing them on their individual merits. The caution around discriminatory impacts can manifest in two significant forms: disparate treatment and disparate impact (Zafar et al., 2017). The former describes intentional discrimination against groups with evidence of explicit reference to group membership. The latter examines the unintentional yet potentially harmful consequences that decision-making processes can have on specific groups, and despite it being facially neutral, it can still contribute to accidental discrimination.

Decision-making entities such as banks, consultancies, and universities must strive to build classifiers free from discrimination, even if their historical data might inherently contain discriminatory elements. Žliobaitė, Kamiran and Calders (2011) highlight a legal case where a leading consultancy firm faced allegations of indirect racial discrimination. They used existing criminal records for pre-employment screening, inadvertently creating bias because of the data's historical correlation between race and criminality. The firm unintentionally discriminated using criminal records, leading to racial bias. This highlights that even indirect discrimination without explicit sensitive data is legally unacceptable.

Likewise, Komiyama, Takeda, Honda and Shimao (2018) have pointed out that the mere exclusion of the 'sensitive variables' is insufficient. The publication further proposed the fairness of an algorithm through a coefficient of determination (CoD) of the sensitive attributes as a constraint. The CoD measures the predictable proportion of the variance of an estimator from sensitive attributes, effectively extending the correlation coefficient for multiple sensitive characteristics. For a deeper exploration of this topic, particularly in the realms of linear and logistic regression, the reader is directed to the works of Scutari, Panero and Proissl (2022), Komiyama et al. (2018), and Žliobaitė et al. (2011).

The crime data analysed here includes sensitive variables like racial population percentages and immigration status, emphasising the need for careful handling. Though a more in-depth exploration of sensitivity could be a progressive step beyond this work, it is deemed pertinent to acknowledge ongoing developments in data decision methodologies within ethically charged contexts. This thesis addresses ethical implications by exploring potential complex variable relationships, and aims to prevent bias through understanding intricate interactions.

7.2 Exploratory Data Analysis

Before proceeding with model fitting, conducting exploratory data analysis is standard statistical practice; it is essential to spot any unwanted data characteristics that could adversely impact the models.

The correlation matrix and the preliminary linear model suggest that the 'percentage of kids born to never-married parents' exhibits the most robust linear association with the target variable. Similarly, a notable significance is found with the 'percentage of divorced population'. Adhering to the principle of model hierarchy, the sensitive variables are not only individually incorporated but also as interaction terms with demographic proportions, namely African American, Caucasian, Asian, Hispanic, and foreign-born populations. Each interaction could provide a more nuanced understanding of the complex factors influencing crime rates. This approach is mindful of the potential for drawing harmful conclusions and, instead, focuses on how these variables may modify the relationship between other predictors and the outcome rather than directly linking demographic aspects to crime rates. Ten interaction variables were added to the final data set, resulting in 110

predictors.

The normality test, namely The Shapiro-Wilk test, yields p-values well below 0.05 for all variables, providing evidence to reject the normality hypothesis, illustrated by the sample histogram in Figure 9. Given the skewness in most variables' distribution, a $log(X + 1)$ transformation is applied across all variables; resulting distributions are depicted in Figure 10. While this aids in normalising the data, it also complicates the interpretation of variable importance later. Post-transformation, the normality test shows marginal improvement but still falls short of confirming normal distribution for all variables.

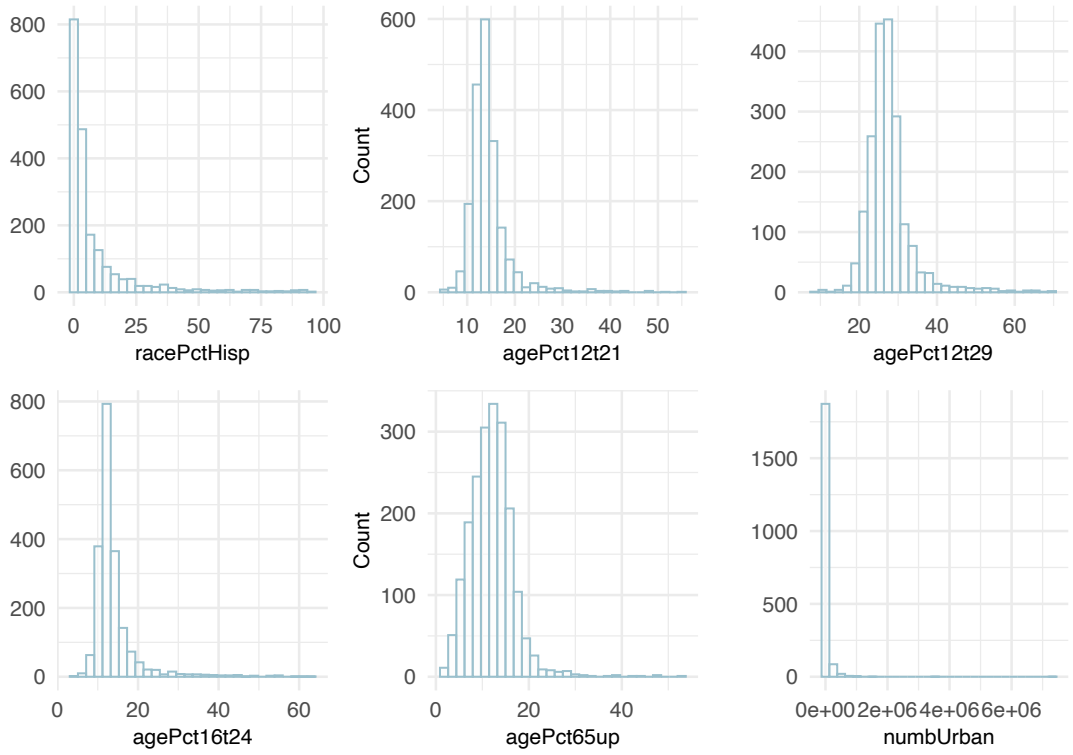

Figure 9: Crime Data. Sample of Variable Histograms Prior to Transformation

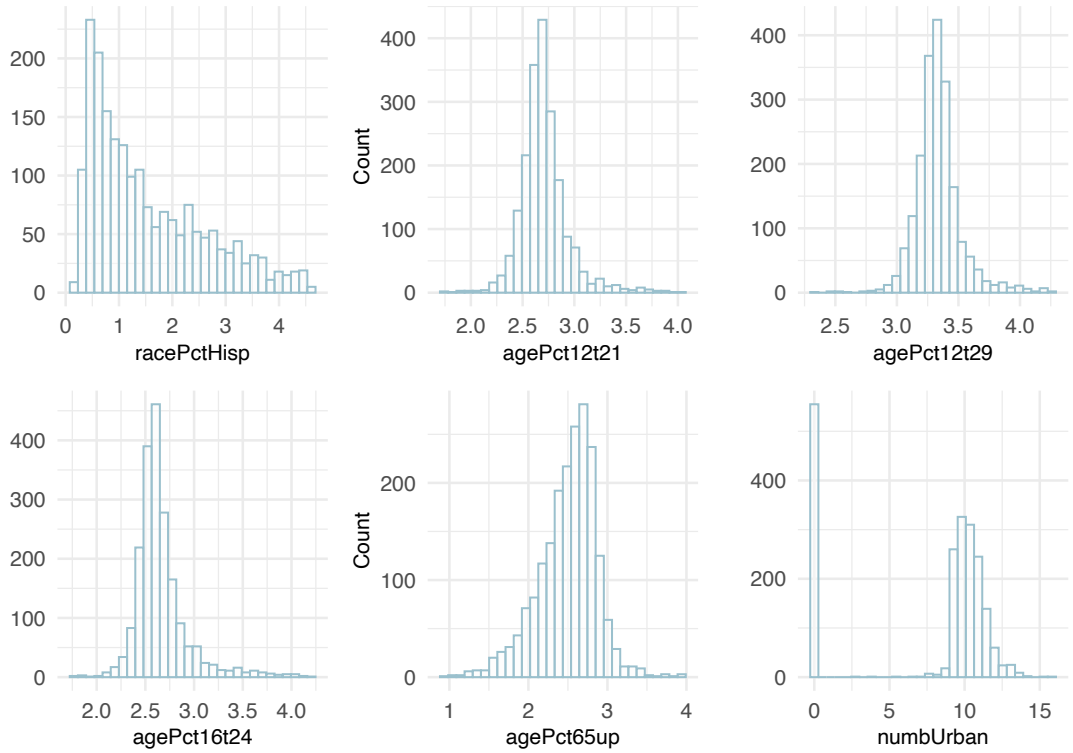

Figure 10: Crime Data. Sample of Variable Histograms Post Logarithmic Transformation

Addressing outliers is typically critical since they can influence model performance and alter the correlations between variables. Tukey's approach to spotting high outliers in data variables, as described in Kaliyaperumal, Kuppusamy and Arumugam (2015), entails determining the interquartile range IQR, which represents the difference between the third ($Q3$) and first ($Q1$) quartiles. Then the lower and upper bounds are then calculated as $Q1 - factorIQR$ and $Q3 + factorIQR$, respectively. After identifying 1300 high outliers, a significant portion of the data, the number reduces to 1216 following the transformation. Although this number is concerning, it also mirrors the complexity of real-world data. Consequently, these outliers will not be eliminated. Refer to Figure 11 for a representative selection of boxplots signifying the outliers.

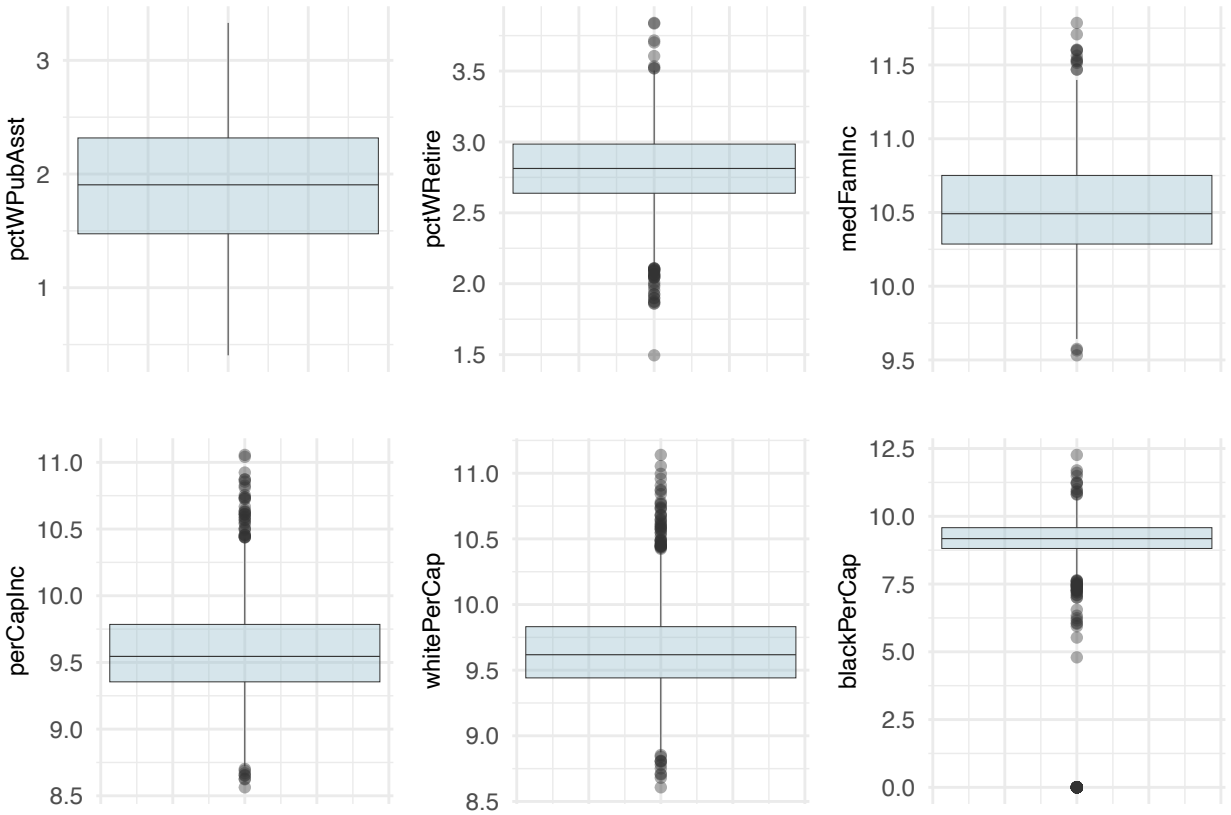

Figure 11: Crime Data. Sample of Boxplots for Visible Outliers

The linear correlations among variables have been examined, as shown in Figure 12. The correlations vary from negligible to near absolute 1, signifying diverse relationships among the variables and underscoring the necessity for variable selection. Hierarchical clustering suggests five distinct groups: urbanisation and immigration patterns; income, age, and housing metrics; education, marital status, and select housing indicators; financial health and family structure; and regionally stable, older demographics. The presence of these clusters suggests potential collinearity among variables within each group.

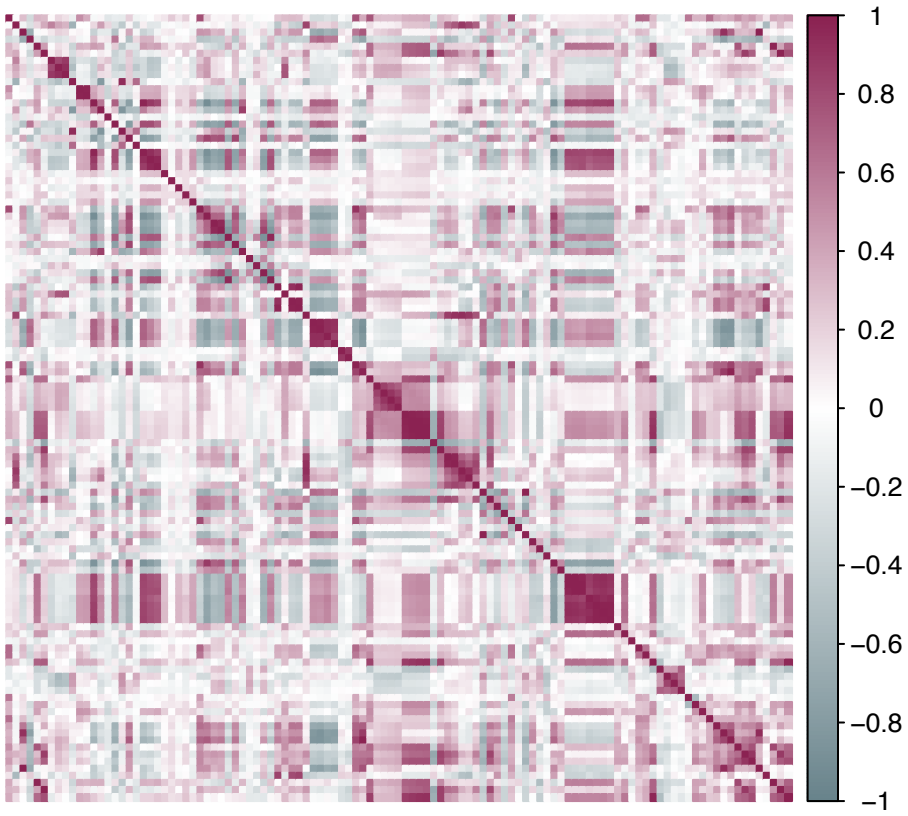

Figure 12: Crime Data. Linear Relationship Strength Among Variables

The scatter plot, see Figure 13, shows a sample of the analysis of linear correlations between predictors and the target. The correlations vary; some predictors show a linear relationship with the target.

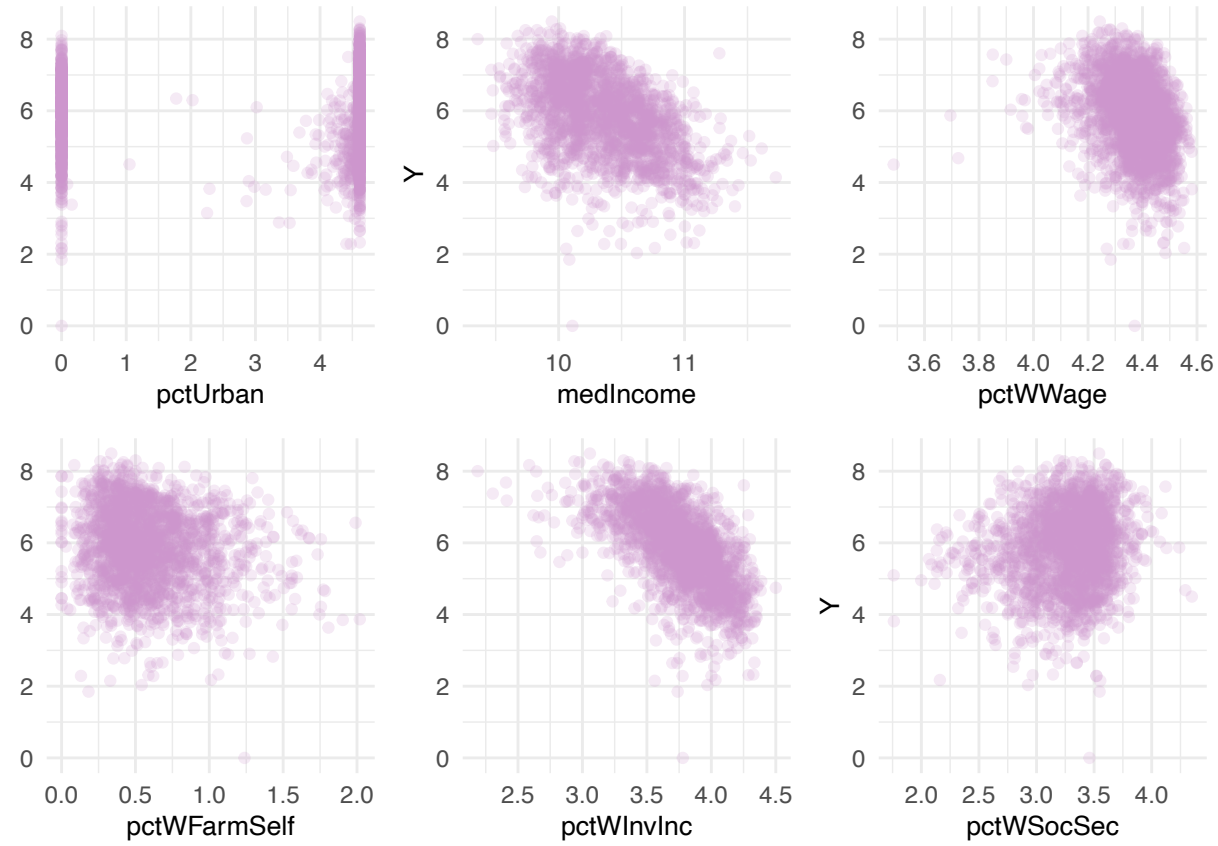

Figure 13: Crime Data. Sample Scatter Plot Analysis of Linear Correlations Between Predictors and Target Variable Post Logarithmic Transformation

Given the challenges of non-normality and violation of independence, Bayesian methodologies have become a valuable approach. With a sizeable sample and apparent relationships between variables and the target, the next step is applying the methodology tested on simulated data.

7.3 Results

Using frequentist regression *Lasso* and *elastic-net* methods, 86 variables of importance are collectively identified. An additional four variables were uniquely selected by the *elastic-net* method.

Fundamentally, very little variable selection is performed. The absolute coefficient values for the initial 40 and 41 predictors in *Lasso* and *elastic-net*, respectively, exceeded 0.1 but dipped below 0.002 for the final included variables. Table 6 provides a descending-order list of variables with coefficients exceeding an absolute value 0.2. Given the broad range of estimated coefficients, definitive conclusions from these selected variables remain challenging to draw from the applied penalised regression methods.

Table 6: Crime Data, Lasso (left) and Elastic-net (right) Regression Results. Ranked Coefficients (>|0.2|)

Lasso Variables	**Coefficients**	**Elastic-Net Variables**	**Coefficients**
(Intercept)	4.44	(Intercept)	4.14
pctWWage	-0.93	PctRecImmig10	-0.81
PctRecImmig10	-0.92	PctEmploy	0.67
PctEmploy	0.91	racePctAsian	0.64
racePctAsian	0.73	pctWWage	-0.59
pctWInvInc	-0.60	pctWInvInc	-0.58
PctFam2Par	-0.55	MedOwnCostPctIncNoMtg	-0.45
MedOwnCostPctIncNoMtg	-0.49	PctPersDenseHous	0.45
PctPersDenseHous	0.45	RentLowQ	-0.39
RentLowQ	-0.41	MedRent	0.38
PctSameState85	0.41	agePct65up	0.36
MedRent	0.40	PctFam2Par	-0.36
agePct65up	0.38	PctSameState85	0.33
PctRecImmig8	0.33	racePctWhite	-0.32
racePctWhite	-0.32	racepctblack	0.30
racepctblack	0.31	PctNotHSGrad	0.29
PctNotHSGrad	0.30	RentHighQ	0.28
MalePctNevMarr	0.30	MalePctNevMarr	0.27
RentHighQ	0.29	pctUrban	0.27
Asian.DivorcePct.Int	-0.28	householdsize	-0.26
pctUrban	0.27	NumUnderPov	0.25
PctLess9thGrade	-0.25	PctRecImmig8	0.25
PctHousOccup	-0.25	PctLess9thGrade	-0.25
NumUnderPov	0.24	Asian.DivorcePct.Int	-0.24
MalePctDivorce	0.23	PctImmigRec10	0.21
PctImmigRec10	0.23	MalePctDivorce	0.21

[a] Coefficients are rounded to two decimal places for presentation purposes

In the variable selection process conducted using *XGBoost*, a threshold of 80% for accumulated total gain is considered. The first variable contributed significantly to the gain, while subsequent variables diminished to below 0.1, as shown in Table 7.

Table 7: Crime Data, XGBoost Method. Results of Feature Importance Using Gain

Variables	**Gain**
PctKids2Par	0.40
racePctWhite	0.10
NumKidsBornNeverMar	0.07
PctKidsBornNeverMar	0.06
Black.KidsBornNeverMar.Int	0.05
PctPersDenseHous	0.03
FemalePctDiv	0.02
PctPopUnderPov	0.02
Hispanic.DivorcePct.Int	0.02
PctFam2Par	0.02
pctWInvInc	0.02
Black.DivorcePct.Int	0.01

[a] Gain values are rounded to two decimal places for presentation purposes

Implemented *Spike-and-slab* method includes 43 variables. Table 8 details BMA and gnet estimates for each variable, presented in descending order of their absolute values.

Table 8: Crime Data, Spike-and-Slab Method Results. BMA and GNet Estimates for Selected Variables Ranked by Absolute Value

Variables	**BMA**	**gnet**
householdsize	-2.45	-2.02
PctPersDenseHous	0.88	0.77
Black.DivorcePct.Int	0.49	0.59
pctWPubAsst	0.42	0.43
White.KidsBornNeverMar.Int	0.10	0.15
PctKidsBornNeverMar	0.09	0.17
pctUrban	0.08	0.09
numbUrban	0.07	0.07
HousVacant	0.07	0.12
agePct65up	0.07	0.13
PctUsePubTrans	-0.06	-0.08
PctPopUnderPov	0.06	0.06
PctWOFullPlumb	-0.05	-0.06
pctWFarmSelf	-0.05	-0.07
AsianPerCap	0.05	0.03
MalePctDivorce	0.05	0.04
NumKidsBornNeverMar	0.05	0.01
NumStreet	0.04	0.05

Table 8: Crime Data, Spike-and-Slab Method Results. BMA and GNet Estimates for Selected Variables Ranked by Absolute Value *(continued)*

Variables	**BMA**	**gnet**
PctRecImmig10	-0.04	-0.07
NumInShelters	0.04	0.05
Hispanic.DivorcePct.Int	0.04	0.13
ForeignBorn.KidsBornNeverMar.Int	0.04	0.05
TotalPctDiv	0.04	0.00
PctVacantBoarded	0.04	0.04
PctHousNoPhone	0.04	0.04
ForeignBorn.DivorcePct.Int	0.03	0.03
PctRecImmig8	-0.03	-0.02
PctEmplManu	-0.03	-0.08
PctRecImmig5	-0.03	-0.02
OtherPerCap	0.03	0.01
PctRecentImmig	-0.03	-0.01
PctBSorMore	-0.03	-0.09
pctWInvInc	-0.02	-0.07
PctVacMore6Mos	-0.02	-0.04
Hispanic.KidsBornNeverMar.Int	0.02	-0.03
PersPerOwnOccHous	-0.02	-0.03
PctBornSameState	-0.02	-0.02
MedNumBR	-0.02	-0.04
indianPerCap	-0.01	-0.01
PctLargHouseOccup	-0.01	0.00
Black.KidsBornNeverMar.Int	0.01	-0.05
LemasPctOfficDrugUn	0.01	0.01
racePctAsian	-0.01	0.01

The *SSL* method selects 104 variables. Out of these, 7 variables have emerged as significantly impactful with strong coefficients in absolute value, while the coefficients of the remaining variables fall below $10^{-}10$, suggesting their negligible contribution. Despite these observations, *SSL* performs commendably on simulated data. The top variables are presented in Table 9.

Table 9: Crime Data, Spike-and-Slab Lasso Results. Top Variables Selected, Ranked by Coefficient Strength

Variables	**Coefficients**
PctPersDenseHous	0.58
agePct65up	0.33
racepctblack	0.28
White.DivorcePct.Int	0.14
NumUnderPov	0.11
pctUrban	0.05

[a] Coefficients are rounded to two decimal places for presentation purposes

Employing the *Bayesian Lasso* approach, a selection of 39 variables is discerned. The selection includes 2 interaction terms; refer to Table 10.

Table 10: Crime Data, Bayesian Lasso Method. Selected Variables

Variables	**Probabilities**
PctPersDenseHous	1.00
pctWInvInc	1.00
agePct65up	0.96
pctUrban	0.94
PctRecImmig10	0.91
NumUnderPov	0.84
racepctblack	0.83
MedOwnCostPctIncNoMtg	0.80
PctWOFullPlumb	0.74
White.DivorcePct.Int	0.73
RentHighQ	0.70
PctImmigRec10	0.69
numbUrban	0.66
MedRent	0.64
PctKids2Par	0.64
RentLowQ	0.64
PctUsePubTrans	0.64
PctFam2Par	0.63
MalePctDivorce	0.62
PctHousOccup	0.61
householdsize	0.61
ForeignBorn.KidsBornNeverMar.Int	0.60
PctPopUnderPov	0.59

Table 10: Crime Data, Bayesian Lasso Method. Selected Variables *(continued)*

Variables	**Probabilities**
PctKidsBornNeverMar	0.58
PctLess9thGrade	0.56
OwnOccMedVal	0.56
TotalPctDiv	0.55
pctWWage	0.55
RentMedian	0.55
PctSameState85	0.54
PctNotHSGrad	0.54
PersPerOwnOccHous	0.53
racePctWhite	0.52
PctEmploy	0.52
PersPerOccupHous	0.52
PctWorkMomYoungKids	0.51
FemalePctDiv	0.51
PersPerFam	0.50
racePctAsian	0.50

In implementing the *horseshoe prior* method from the *'horseshoe'* package, identical predictor selection is observed for both the *truncated Cauchy* and *half Cauchy* methods; see Figures 14 and 15 for credible interval results. The three chosen predictors relate to percentages of the population aged over 65, households with 1989 investment/rent income, and residents in dense housing.

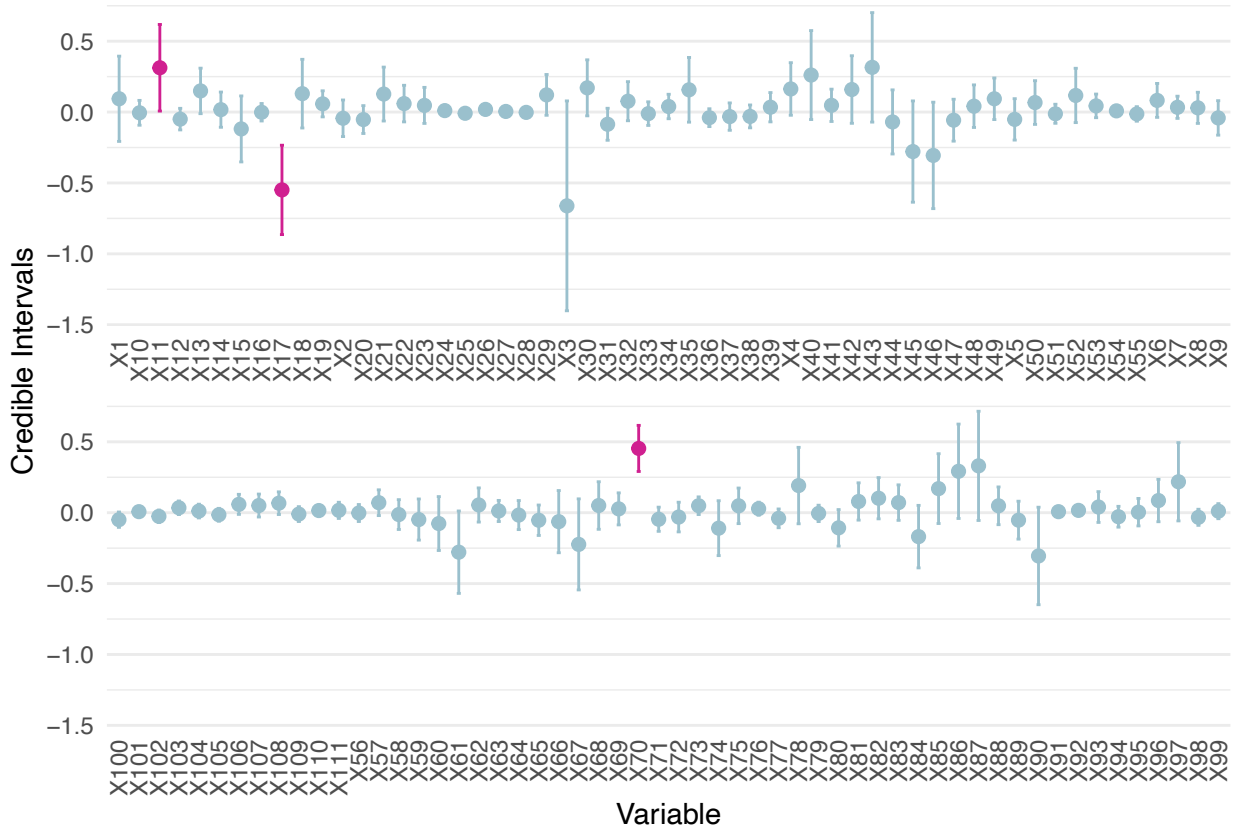

Figure 14: Crime Data. Credible Interval Results using 'horseshoe' Truncated Cauchy Prior

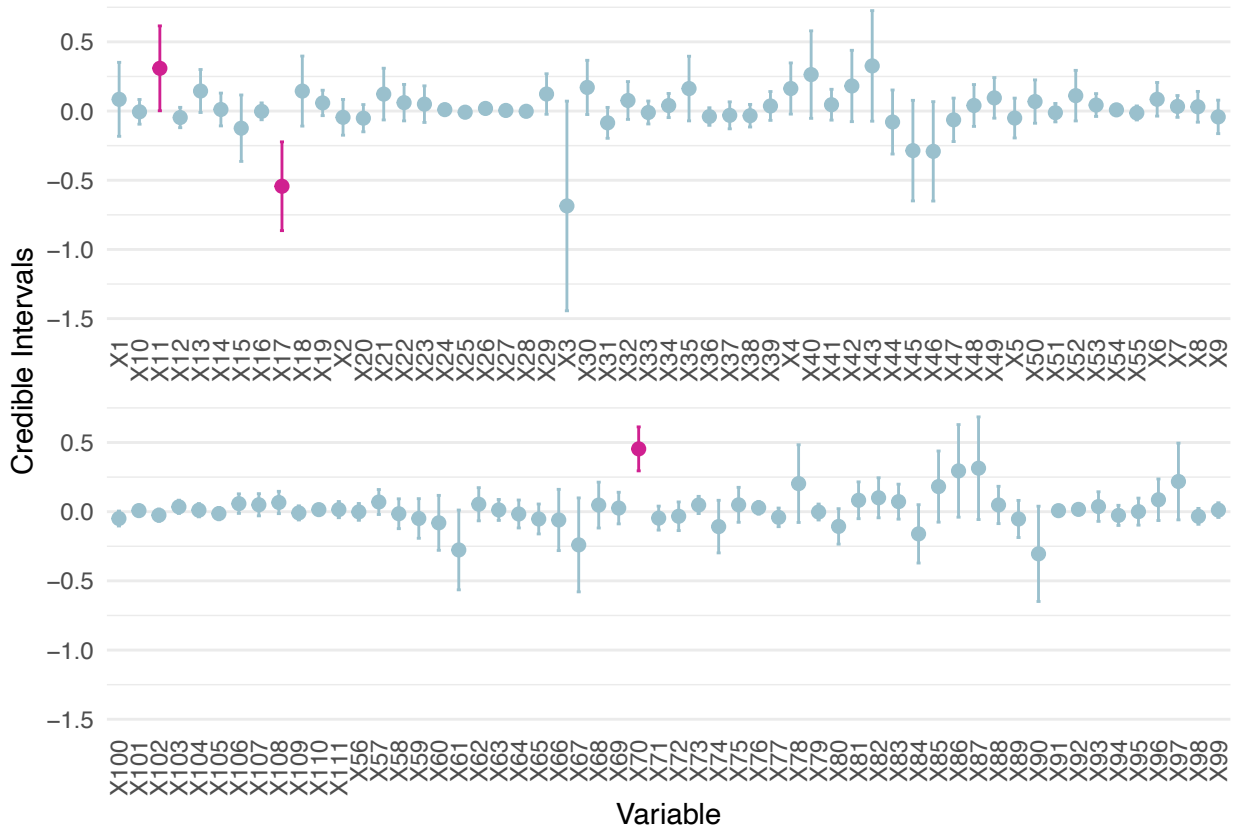

Figure 15: Crime Data. Credible Interval Results using 'horseshoe' Half Cauchy Prior

Employing the *'bayesreg' horseshoe* and *horseshoe+* prior selection methods identify two key variables: the percentage of households with investment/rent income in 1989 and the percentage of persons in dense housing, see Figures 16 and 17 for credible interval results.

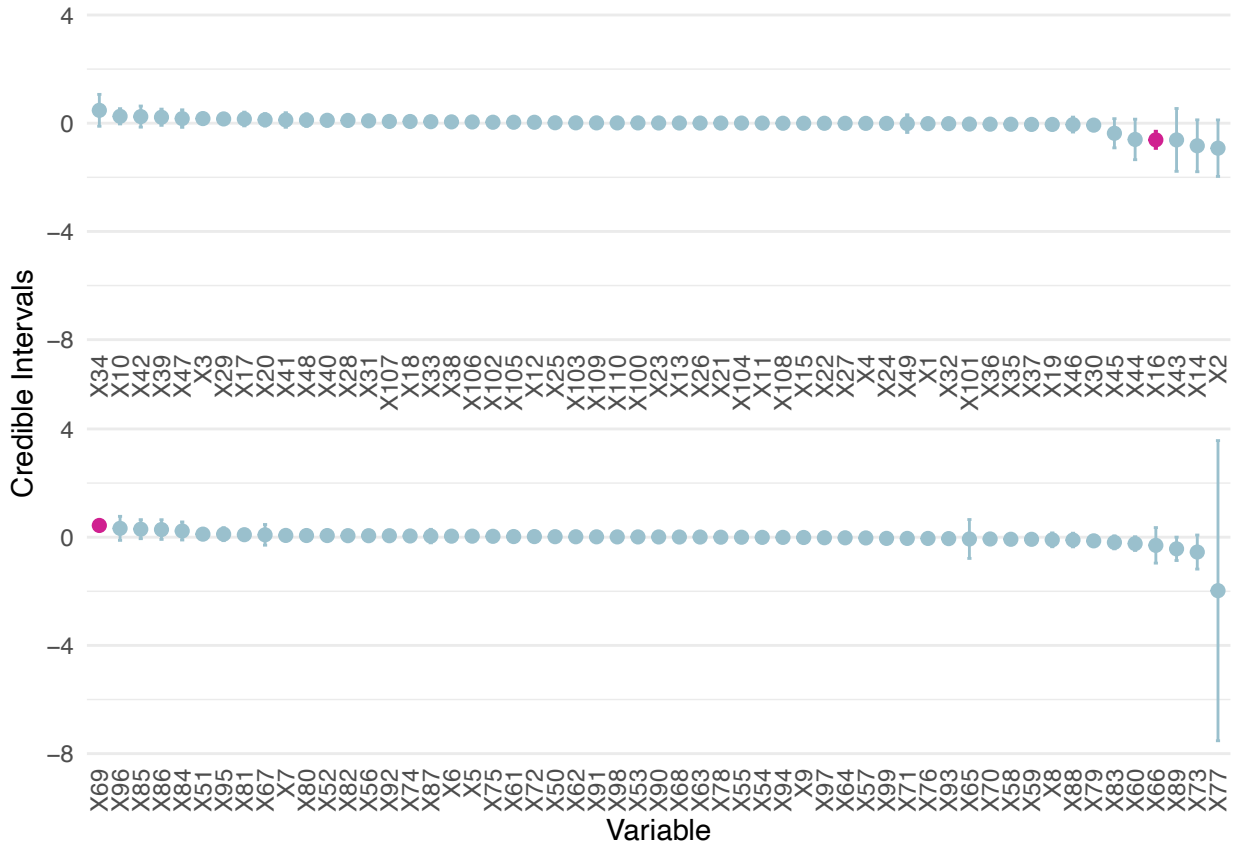

Figure 16: Crime Data. Credible Interval Results using 'bayesreg' Horseshoe Prior

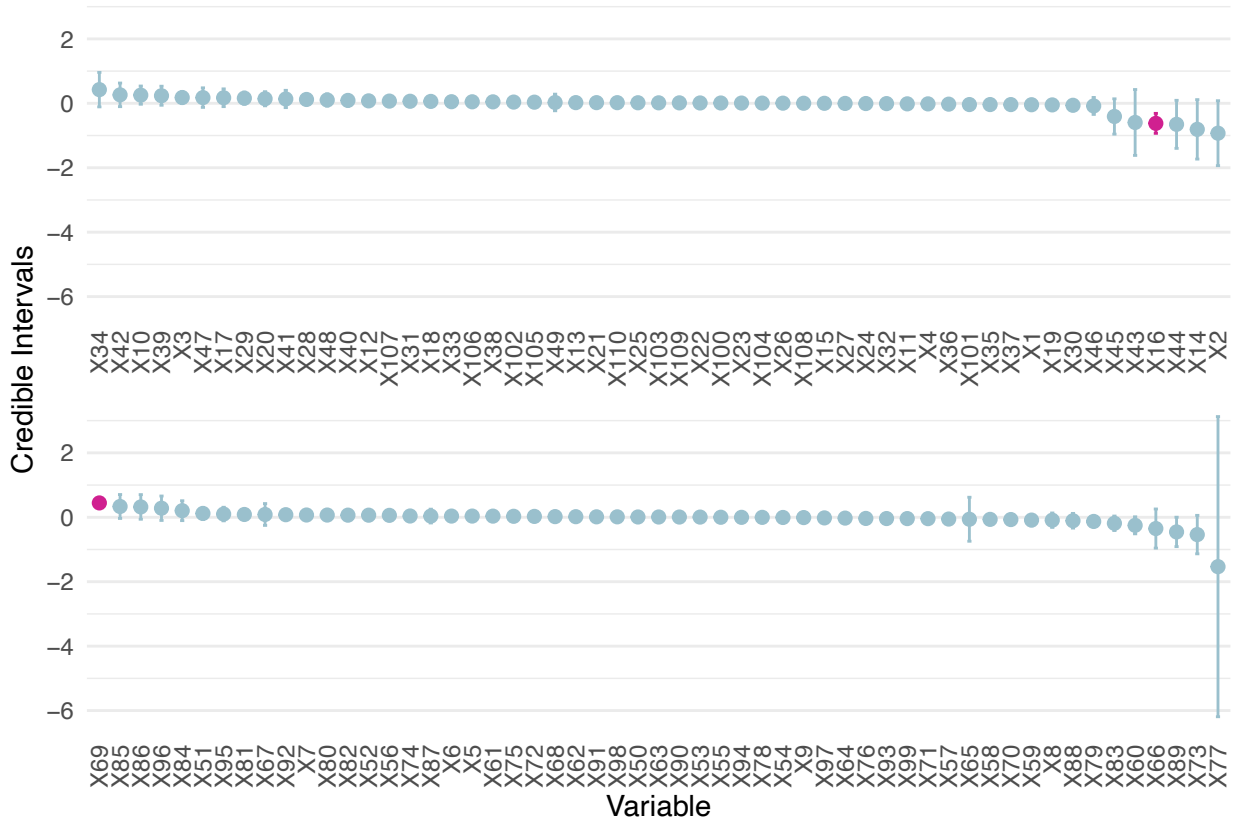

Figure 17: Crime Data. Credible Interval Results using 'bayesreg' Horseshoe+ Prior

The *S5* method selects 6 total variables with high inclusion probabilities, including one interaction variable, refer to Table 11.

Table 11: Crime Data, S5 Method. Selected Variables and Their Marginal Inclusion Probabilities

Variables	**Probabilities**
population	1.00
racepctblack	1.00
PctKids2Par	1.00
PctPersDenseHous	1.00
White.DivorcePct.Int	1.00
agePct65up	0.96

The summary of selected variables by each method is presented in Table 12. The number of selected variables varies significantly across methods, which leads to different interpretations.

Table 12: Summary of Crime Data Results

Package	Method	Predictors	Approach
Frequentist Methods			
glmnet	Lasso	86	Point estimates for coefficients
glmnet	Elastic-Net	90	Point estimates for coefficients
Machine Learning Method			
caret	XGBoost	12	Predictor importance
Bayesian Methods			
spikeslab	Spike-and-Slab Prior	43	Point estimates
SSLASSO	Spike-and-Slab Lasso	7 significant/104 total	Point estimates
horseshoe	Horseshoe Prior, TC	3	Credible intervals
horseshoe	Horseshoe Prior, HC	3	Credible intervals
bayesreg	Horseshoe Prior	2	Credible intervals
bayesreg	Horseshoe+ Prior	2	Credible intervals
BayesS5	S5 Method	6	Predictor inclusion probabilities
monomvn	Bayesian Lasso	39	Predictor inclusion probabilities

[a] Predictors refer to the variables chosen by each method for inclusion in the final model. The selection criteria and mechanisms vary by method, as detailed in the methodology chapters.

In the synthetic data study, applying *horseshoe* priors demonstrates superior performance compared to other methodologies, thus warranting their consideration as the foundation for crime data analysis. The variables denoting the percentage of individuals in dense housing and the percentage of households with investment/rent income in 1989 are consistently identified as significant across all employed methods. The latter variable coefficient from the *SSL* method is negligible. The variable indicating the percentage of the population aged 65 and over is recognised as significant by all techniques, except for the *'bayesreg'* package when *horseshoe* and *horseshoe+* priors are applied, as well as *XGBoost*. Among the manually crafted interaction terms, several are deemed important. Notably, the interaction between the percentage of the Caucasian population and the percentage of the divorced population is highlighted by all methods except those using *horseshoe* priors. The findings should be further verified through consultation with domain experts or a comprehensive review of associated literature encompassing socio-economic, law enforcement, and crime data.

8 Conclusions

This thesis investigates Bayesian model selection techniques and associated R packages, contrasting their efficacy with frequentist methods *Lasso* and *elastic-net* and the machine learning *Extreme Gradient Boosting* using simulated datasets. The results underscore the superior performance of Bayesian methods when model uncertainty, credible intervals specifically, is integrated into coefficient estimates. Simulated data comparisons encompass independent regressions, imposed correlations, interaction terms, polynomials, and categorical covariates across diverse dimensionality settings of predictors and data points. Subsequently, these methods are applied to a real crime dataset. While frequentist and dimensionality-reduction techniques remain dominant among statisticians, the rising popularity of Bayesian methods is justified. This thesis discusses the advantages of Bayesian approaches, such as the seamless incorporation of model uncertainty and the generation of interval estimates or inclusion probabilities for covariate coefficients.

For future reference, when the target is known, training data can inform the optimal alpha value for *elastic-net* and should be manually cross-validated when using the *'glmnet'* package. *XGBoost*, while not excelling in linear regression tasks, warrants further exploration through random hyperparameter tuning and diverse variable selection strategies. Despite its computational intensity - requiring an hour for each simulated dataset due to an extensive grid search - *XGBoost's* linear regression capabilities merit deeper investigation, given the current lack of publications on the topic. While it is resilient to multi-collinearity, it struggles with untreated outliers, as observed in the crime data. Leveraging L1 and L2 regularisation, which *XGBoost* supports, should be considered. *XGBoost* is typically employed for prediction rather than inference, the latter being this thesis's focus. Even with increased algorithm repetitions, *S5* from *'BayesS5'* package showcases commendable efficiency, especially when juxtaposed with XGBoost's grid search duration. Its performance excels when dealing with uncorrelated data.

For beginner users, it is crucial to note that *SSL* from *'SSLASSO'* and *S5* are unsuitable for categorical variables and should be clearly stated in the package instructions. The *'spikeslab'* package's *spike-and-slab* formulation couples parameter estimates with respective variable stabilities, that is, the consistent selection frequency of a variable, which can further refine its performance. Adopting a conservative stance, variables with stability above 80% - as noted by the package authors (Ishwaran et al., 2010) - could be chosen, potentially proposing even lower dimensions in the final model. Note that for relative model comparison, AIC is used, which is not a Bayesian approach. The *horseshoe* prior induces a strong shrinkage effect, and performance from both the *'bayesreg'* and *'horseshoe'* packages markedly outshines alternative methods. Using credible intervals to measure uncertainty allows the methods to avoid selecting false positive signals. For *Bayesian Lasso* from *'monomvn'*, future work should cross-validate the *lambda*2 penalty. A bespoke cross-validation function is necessary as the primary function lacks this feature. Notably, *Bayesian Lasso* surpasses other methods in scenarios where data is time-correlated and the number of predictors greatly

exceeds the number of data points. However, in other data within this dimensionality, there is a pressing need to explore alternative methods or techniques, given the current suboptimal performance of the tested approaches.

Future evaluations should probe the stability of these methodologies, particularly by fitting models with varying initial states that dictate the onset of random number generation algorithms. Simulated data effectively illuminates the relative strengths of different packages, especially regarding data dimensionality and correlation. The packages' performance on real data largely mirrors the simulation results, emphasising their practical relevance: the inclusion of different variables shows significant inconsistency. For a robust real data analysis, it is recommended to use multiple methods, consult domain experts for validation, and consider additional interaction terms and polynomial features to reveal complex underlying patterns.

9 Bibliography

Auguie, B. (2017). *gridExtra: Miscellaneous functions for "grid" graphics*. https://CRAN.R-project.org/package=gridExtra

Barker, R. J., & Link, W. A. (2013). Bayesian multimodel inference by RJMCMC: A gibbs sampling approach. *The American Statistician*, *67*(3), 150–156. https://doi.org/10.1080/00031305.2013.791644

Bhadra, A., Datta, J., Polson, N. G., & Willard, B. T. (2016). *Default bayesian analysis with global-local shrinkage priors*. https://doi.org/10.48550/arXiv.1510.03516

Bhattacharya, A., Chakraborty, A., & Mallick, B. K. (2016). *Fast sampling with gaussian scale-mixture priors in high-dimensional regression*. http://arxiv.org/abs/1506.04778

Breheny, P., & Huang, J. (2011). Coordinate descent algorithms for nonconvex penalized regression, with applications to biological feature selection. *The Annals of Applied Statistics*, *5*(1), 232–253. https://doi.org/10.1214/10-AOAS388

Brooks, S. P., & Gelman, A. (1998). General methods for monitoring convergence of iterative simulations. *Journal of Computational and Graphical Statistics*, *7*(4), 434–455. https://doi.org/10.1080/10618600.1998.10474787

Carvalho, C. M., Polson, N. G., & Scott, J. G. (2010). The horseshoe estimator for sparse signals. *Biometrika*, *97*, 465–480. https://doi.org/10.1093/biomet/asq017

Chen, T., & Guestrin, C. (2016). *XGBoost: A scalable tree boosting system*. https://doi.org/10.1145/2939672.2939785

Dairu, X., & Shilong, Z. (2021). Machine learning model for sales forecasting by using XGBoost. *2021 IEEE International Conference on Consumer Electronics and Computer Engineering (ICCECE)*, 480–483. https://doi.org/10.1109/ICCECE51280.2021.9342304

Daniel F. Schmidt, & Enes Makalic. (2021). *Bayesreg: Bayesian regression models with global-local shrinkage priors*. https://CRAN.R-project.org/package=bayesreg

Dias, L. C., Morton, A., & Quigley, J. (Eds.). (2017). *Elicitation: The science and art of structuring judgement*. Springer.

Fan, J., & Lv, J. (2008). Sure independence screening for ultrahigh dimensional feature space. *Journal of the Royal Statistical Society. Series B: Statistical Methodology*, *70*, 849–911. https://doi.org/10.1111/j.1467-9868.2008.00674.x

Fan, Z. (2016). *Lecture 20 - bayesian analysis*. https://web.stanford.edu/class/stats200/Lecture20.pdf

Friedman, J., Hastie, T., Tibshirani, R., Narasimhan, B., Tay, K., Simon, N., & Yang, J. (2022). *Glmnet: Lasso and elastic-net regularized generalized linear models*. https://CRAN.R-project.org/package=glmnet

Gelling, N., Schofield, M. R., & Barker, R. J. (2019). R package rjmcmc: Reversible jump MCMC using post-processing. *Australian and New Zealand Journal of Statistics*, *61*, 189–212. https://doi.org/10.1111/ANZS.12263

Gelman, A., Carlin, J. B., Stern, H. S., Dunson, D. B., Vehtari, A., & Rubin, D. B. (2020). *Bayesian*

data analysis. Third edition (pp. 165–175).

Geweke, J. (1996). *Variable selection and model comparison in regression* (Working Papers 539). Federal Reserve Bank of Minneapolis. https://ideas.repec.org/p/fip/fedmwp/539.html

Gramacy, R. B., Moler, C., & Turlach, B. A. (2023). *Monomvn: Estimation for MVN and student-t data with monotone missingness*. https://bobby.gramacy.com/r_packages/monomvn/

Green, P. J. (1995). Reversible jump markov chain monte carlo computation and bayesian model determination. *Biometrika*, *82*, 711–732. https://doi.org/10.2307/2337340

Guo, R., Zhao, Z., Wang, T., Liu, G., Zhao, J., & Gao, D. (2020). Degradation state recognition of piston pump based on ICEEMDAN and XGBoost. *Applied Sciences*, *10*, 6593. https://doi.org/10.3390/app10186593

Hans, C. (2009). Bayesian lasso regression. *Biometrika*, *96*(4), 835–845. http://www.jstor.org/stable/27798870

Hans, C., Dobra, A., & West, M. (2007). Shotgun stochastic search for "large p" regression. *Journal of the American Statistical Association*, *102*, 507–516. https://doi.org/10.1198/016214507000000121

Harrell, F. E., Jr. (2022). *Hmisc: Harrell miscellaneous*. https://hbiostat.org/R/Hmisc/

Hastie, T., Tibshirani, R., & Friedman, J. H. (2017). *The elements of statistical learning: Data mining, inference, and prediction* (2nd Edition). Springer. http://www.springer.com/series/692

Hoeting, J. A., Madigan, D., Raftery, A. E., & Volinsky, C. T. (1999). Bayesian model averaging: A tutorial. *Statistical Science*, *14*(4), 382–401. http://www.jstor.org/stable/2676803

Ishwaran, H. (2022). *Spikeslab: Prediction and variable selection using spike and slab regression*. https://ishwaran.org/

Ishwaran, H., Kogalur, U. B., & Rao, J. S. (2010). Spikeslab: Prediction and variable selection using spike and slab regression. *R Journal*, *2*, 68–73. https://doi.org/10.32614/rj-2010-018

Ishwaran, H., & Rao, J. S. (2005). Spike and slab variable selection: Frequentist and bayesian strategies. In *Annals of Statistics* (Vol. 33, pp. 730–773). https://doi.org/10.1214/009053604000001147

Jeffreys, H. (1935). Some tests of significance, treated by the theory of probability. *Mathematical Proceedings of the Cambridge Philosophical Society*, *31*(2), 203–222. https://doi.org/10.1017/S030500410001330X

Jianqing, F., Moore, F. L., & Jinchi, L. (2010). A selective overview of variable selection in high dimensional feature space. In *Statistica Sinica*.

Johnstone, I. M., & Silverman, B. W. (2004). Needles and straw in haystacks: Empirical BAYES estimates of possibly sparse sequences. *Annals of Statistics*, *32*, 1594–1649. https://doi.org/10.1214/009053604000000030

Kahneman, D. (2011). *Thinking, fast and slow*. Farrar, Straus; Giroux.

Kaliyaperumal, S. K., Kuppusamy, M., & Arumugam, S. (2015). Labeling methods for identifying outliers. In *International Journal of Statistics and Systems* (Vol. 10, pp. 231–238). http://www.ripublication.com

Kass, R. E., & Raftery, A. E. (1995). Bayes factors. *Journal of the American Statistical Association*,

90, 773. https://doi.org/10.2307/2291091

Kassambara, A. (2020). *Ggpubr: ggplot2 based publication ready plots*. https://rpkgs.datanovia.com/ggpubr/

Kirkpatrick, S., Gelatt, C. D., & Vecchi, M. P. (1983). Optimization by simulated annealing. *Science*, *220*(4598), 671–680. https://doi.org/10.1126/science.220.4598.671

Komiyama, J., Takeda, A., Honda, J., & Shimao, H. (2018). Nonconvex optimization for regression with fairness constraints. In J. Dy & A. Krause (Eds.), *Proceedings of the 35th international conference on machine learning* (Vol. 80, pp. 2737–2746). PMLR. https://proceedings.mlr.press/v80/komiyama18a.html

Kuhn, M. (2023). *Caret: Classification and regression training*. https://github.com/topepo/caret/

Lempers, F. B. (1971). *Posterior probabilities of alternative linear models: Some theoretical considerations and empirical experiments*. Rotterdam University Press.

Li, J. (2020). *When to not use XGBoost?* https://www.kaggle.com/discussions/general/196542

Makalic, E., & Schmidt, D. F. (2016). *High-dimensional bayesian regularised regression with the BayesReg package*. http://arxiv.org/abs/1611.06649

Meier, L., Van De Geer, S., & Bühlmann, P. (2008). The group lasso for logistic regression. *Journal of the Royal Statistical Society Series B: Statistical Methodology*, *70*(1), 53–71. https://doi.org/10.1111/j.1467-9868.2007.00627.x

Mitchell, T. J., & Beauchamp, J. J. (1988). Bayesian variable selection in linear regression. *Journal of the American Statistical Association*, *83*, 1023. https://doi.org/10.2307/2290129

Moran, G. E., Ročková, V., & George, E. I. (2019). Variance prior forms for high-dimensional bayesian variable selection. *Bayesian Analysis*, *14*. https://doi.org/10.1214/19-BA1149

Neath, A. A., & Cavanaugh, J. E. (2012). The bayesian information criterion: Background, derivation, and applications. *WIREs Comput. Stat.*, *4*(2), 199–203. https://doi.org/10.1002/wics.199

Olvera Astivia, O., & Kroc, E. (2018). Centering in multiple regression does not always reduce multicollinearity: How to tell when your estimates will not benefit from centering. *Educational and Psychological Measurement*, *79*, 001316441881780. https://doi.org/10.1177/0013164418817801

Park, T., & Casella, G. (2008). The bayesian lasso. *Journal of the American Statistical Association*, *103*(482), 681–686. https://doi.org/10.1198/016214508000000337

Pas, S. van der, Scott, J., Chakraborty, A., & Bhattacharya, A. (2019). *Horseshoe: Implementation of the horseshoe prior*. https://CRAN.R-project.org/package=horseshoe

Pas, S. van der, Szabó, B., & Vaart, A. van der. (2017). Uncertainty Quantification for the Horseshoe (with Discussion). *Bayesian Analysis*, *12*(4), 1221–1274. https://doi.org/10.1214/17-BA1065

Plummer, M., Best, N., Cowles, K., & Vines, K. (2005). CODA: Convergence diagnosis and output analysis for MCMC. *R News*, *6*.

Polson, N. G., & Scott, J. G. (2010). Shrink globally, act locally: Sparse bayesian regularization and prediction*. In *Bayesian Statistics 9* (pp. 501–538). Oxford University Press. https:

//doi.org/10.1093/acprof:oso/9780199694587.003.0017

Polson, N., & Sun, L. (2017). Bayesian l_0 regularized least squares. *Applied Stochastic Models in Business and Industry*, *35*. https://doi.org/10.1002/asmb.2381

Price, H., & Manson, A. (2002). Uninformative priors for bayes' theorem. *AIP Conference Proceedings*, *617*, 379–391. https://doi.org/10.1063/1.1477060

Redmond, M. (2011). *Communities and Crime Unnormalized*. UCI Machine Learning Repository. https://doi.org/10.24432/C5PC8X

Ripley, B. (2022). *MASS: Support functions and datasets for venables and ripley's MASS*. http://www.stats.ox.ac.uk/pub/MASS4/

Rockova, V., & Moran, G. (2019). *SSLASSO: The spike-and-slab LASSO*. http://faculty.chicagobooth.edu/veronika.rockova/ssl.pdf

Ročková, V., & George, E. I. (2018). The spike-and-slab LASSO. *Journal of the American Statistical Association*, *113*, 431–444. https://doi.org/10.1080/01621459.2016.1260469

Rue, H. (2001). Fast sampling of gaussian markov random fields. *Journal of the Royal Statistical Society Series B: Statistical Methodology*, *63*, 325–338. https://doi.org/10.1111/1467-9868.00288

Scutari, M., Panero, F., & Proissl, M. (2022). Achieving fairness with a simple ridge penalty. *Statistics and Computing*, *32*, 77. https://doi.org/10.1007/s11222-022-10143-w

Shin, M., Bhattacharya, A., & Johnson, V. E. (2017). *Scalable bayesian variable selection using nonlocal prior densities in ultrahigh-dimensional settings*. https://arxiv.org/abs/1507.07106

Shin, M., & Tian, R. (2020). *BayesS5: Bayesian variable selection using simplified shotgun stochastic search with screening (S5)*. https://arxiv.org/abs/1507.07106v4

Spiegelhalter, D. J., Best, N. G., Carlin, B. P., & Linde, A. van der. (2014). The deviance information criterion: 12 years on. In *Journal of the Royal Statistical Society. Series B (Statistical Methodology): Vols. 76. No. 3* (pp. 485–493).

Steyerberg, E. W. (2019). *Clinical prediction models: Statistics for biology and health clinical prediction models a practical approach to development, validation, and updating second edition*. Springer Cham. https://doi.org/10.1007/978-3-030-16399-0

Tadesse, M. G., & Vannucci, M. (2022). *Handbook of bayesian variable selection*. Chapman & Hall. https://doi.org/10.1080/00031305.2013.791644

Tibshirani, R. (1996). Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society. Series B (Methodological)*, *58*(1), 267–288. http://www.jstor.org/stable/2346178

Train, K. E. (2012). *Discrete choice methods with simulation* (2nd Edition, pp. 284–314). Cambridge University Press. https://doi.org/10.1017/CBO9780511805271

Wang, J., Xu, J., Zhao, C., Peng, Y., & Wang, H. (2019). An ensemble feature selection method for high-dimensional data based on sort aggregation. *Systems Science and Control Engineering*, *7*, 32–39. https://doi.org/10.1080/21642583.2019.1620658

Wei, T., & Simko, V. (2021). *Corrplot: Visualization of a correlation matrix*. https://github.com/taiyun/corrplot

Wickham, H. (2023). *Tidyverse: Easily install and load the tidyverse*. https://CRAN.R-project.org/package=tidyverse

Wilke, C. O. (2020). *Cowplot: Streamlined plot theme and plot annotations for ggplot2*. https://wilkelab.org/cowplot/

Xie, Y. (2022). *Knitr: A general-purpose package for dynamic report generation in r*. https://yihui.org/knitr/

Xu, C. (2021). *Ultra-high dimensional bayesian variable selection with lasso-type priors* [PhD thesis, Southern Methodist University]. https://scholar.smu.edu/hum_sci_statisticalscience_etds/27

Zafar, M. B., Valera, I., Rodriguez, M. G., & Gummadi, K. P. (2017). Fairness beyond disparate treatment & disparate impact: Learning classification without disparate mistreatment. *26th International World Wide Web Conference, WWW 2017*, 1171–1180. https://doi.org/10.1145/3038912.3052660

Zhu, H. (2021). *kableExtra: Construct complex table with kable and pipe syntax*. https://CRAN.R-project.org/package=kableExtra

Žliobaitė, I., Kamiran, F., & Calders, T. (2011). Handling conditional discrimination. *Proceedings - IEEE International Conference on Data Mining, ICDM*, 992–1001. https://doi.org/10.1109/ICDM.2011.72

Zou, H., & Hastie, T. (2005). Regularization and variable selection via the elastic net. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, *67*, 301–320. https://doi.org/10.1111/j.1467-9868.2005.00503.x

10 Appendix

10.1 Tables

Table 13: Crime Data. Summary of Selected Variables and Their Characteristics for Model Fitting

Variable	Included	Type
US state	No	Nominal
numeric code for county	No	Categorical
numeric code for community	No	Categorical
community name	No	Text
fold number	No	Categorical
population of community	Yes	Continuous
mean people per household	Yes	Continuous
percentage of population that is african american	Yes	Continuous
percentage of population that is caucasian	Yes	Continuous
percentage of population that is of asian heritage	Yes	Continuous
percentage of population that is of hispanic heritage	Yes	Continuous
percentage of population that is 12-21 in age	Yes	Continuous
percentage of population that is 12-29 in age	Yes	Continuous
percentage of population that is 16-24 in age	Yes	Continuous
percentage of population that is 65 and over in age	Yes	Continuous
number of people living in areas classified as urban	Yes	Continuous
percentage of people living in areas classified as urban	Yes	Continuous
median household income	Yes	Continuous
percentage of households with wage or salary income in 1989	Yes	Continuous
percentage of households with farm or self employment income in 1989	Yes	Continuous
percentage of households with investment / rent income in 1989	Yes	Continuous
percentage of households with social security income in 1989	Yes	Continuous
percentage of households with public assistance income in 1989	Yes	Continuous
percentage of households with retirement income in 1989	Yes	Continuous
median family income	Yes	Continuous
per capita income	Yes	Continuous
per capita income for caucasians	Yes	Continuous
per capita income for african americans	Yes	Continuous
per capita income for native americans	Yes	Continuous

Table 13: Crime Data. Summary of Selected Variables and Their Characteristics for Model Fitting *(continued)*

Variable	Included	Type
per capita income for people with asian heritage	Yes	Continuous
per capita income for people with other heritage	Yes	Continuous
per capita income for people with hispanic heritage	Yes	Continuous
number of people under the poverty level	Yes	Continuous
percentage of people under the poverty level	Yes	Continuous
percentage of people 25 and over with less than a 9th grade education	Yes	Continuous
percentage of people 25 and over that are not high school graduates	Yes	Continuous
percentage of people 25 and over with a bachelors degree or higher education	Yes	Continuous
percentage of people 16 and over, in the labor force, and unemployed	Yes	Continuous
percentage of people 16 and over who are employed	Yes	Continuous
percentage of people 16 and over who are employed in manufacturing	Yes	Continuous
percentage of people 16 and over who are employed in professional services	Yes	Continuous
percentage of people 16 and over who are employed in management	Yes	Continuous
percentage of people 16 and over who are employed in professional occup.	Yes	Continuous
percentage of males who are divorced	Yes	Continuous
percentage of males who have never married	Yes	Continuous
percentage of females who are divorced	Yes	Continuous
percentage of population who are divorced	Yes	Continuous
mean number of people per family	Yes	Continuous
percentage of families (with kids) that are headed by two parents	Yes	Continuous
percentage of kids in family housing with two parents	Yes	Continuous
percent of kids 4 and under in two parent households	Yes	Continuous
percent of kids age 12-17 in two parent households	Yes	Continuous
percentage of moms of kids 6 and under in labor force	Yes	Continuous
percentage of moms of kids under 18 in labor force	Yes	Continuous
number of kids born to never married	Yes	Continuous
percentage of kids born to never married	Yes	Continuous
total number of people known to be foreign born	Yes	Continuous
percentage of immigrants who immigated within last 3 years	Yes	Continuous
percentage of immigrants who immigated within last 5 years	Yes	Continuous
percentage of immigrants who immigated within last 8 years	Yes	Continuous
percentage of immigrants who immigated within last 10 years	Yes	Continuous

Table 13: Crime Data. Summary of Selected Variables and Their Characteristics for Model Fitting *(continued)*

Variable	Included	Type
percent of population who have immigrated within the last 3 years	Yes	Continuous
percent of population who have immigrated within the last 5 years	Yes	Continuous
percent of population who have immigrated within the last 8 years	Yes	Continuous
percent of population who have immigrated within the last 10 years	Yes	Continuous
percent of people who speak only English	Yes	Continuous
percent of people who do not speak English well	Yes	Continuous
percent of family households that are large (6 or more)	Yes	Continuous
percent of all occupied households that are large (6 or more people)	Yes	Continuous
mean persons per household (numeric - decimal)	Yes	Continuous
mean persons per owner occupied household	Yes	Continuous
mean persons per rental household (numeric - decimal)	Yes	Continuous
percent of people in owner occupied households (numeric - decimal)	Yes	Continuous
percent of persons in dense housing (more than 1 person per room)	Yes	Continuous
percent of housing units with less than 3 bedrooms	Yes	Continuous
median number of bedrooms	Yes	Continuous
number of vacant households	Yes	Continuous
percent of housing occupied	Yes	Continuous
percent of households owner occupied	Yes	Continuous
percent of vacant housing that is boarded up	Yes	Continuous
percent of vacant housing that has been vacant more than 6 months	Yes	Continuous
median year housing units built	Yes	Continuous
percent of occupied housing units without phone (in 1990, this was rare!)	Yes	Continuous
percent of housing without complete plumbing facilities	Yes	Continuous
owner occupied housing - lower quartile value	Yes	Continuous
owner occupied housing - median value	Yes	Continuous
owner occupied housing - upper quartile value	Yes	Continuous
rental housing - lower quartile rent	Yes	Continuous
rental housing - median rent (Census variable H32B from file STF1A)	Yes	Continuous
rental housing - upper quartile rent	Yes	Continuous
median gross rent (Census H43A from STF3A - with utilities)	Yes	Continuous
median gross rent as a percentage of household income	Yes	Continuous
median owners cost as a pct of household income (mortgage)	Yes	Continuous
median owners cost as a pct of household income (without mortgage)	Yes	Continuous

Table 13: Crime Data. Summary of Selected Variables and Their Characteristics for Model Fitting *(continued)*

Variable	Included	Type
number of people in homeless shelters	Yes	Continuous
number of homeless people counted in the street	Yes	Continuous
percent of people foreign born	Yes	Continuous
percent of people born in the same state as currently living	Yes	Continuous
percent of people living in the same house as in 1985 (5 years before)	Yes	Continuous
percent of people living in the same city as in 1985 (5 years before)	Yes	Continuous
percent of people living in the same state as in 1985 (5 years before)	Yes	Continuous
number of sworn full time police officers	No	Continuous
sworn full time police officers per 100K population	No	Continuous
number of sworn full time police officers in field operations	No	Continuous
sworn full time police officers in field operations	No	Continuous
total requests for police	No	Continuous
total requests for police per 100K popuation	No	Continuous
total requests for police per police officer	No	Continuous
police officers per 100K population	No	Continuous
a measure of the racial match between the community and the police force	No	Continuous
percent of police that are caucasian	No	Continuous
percent of police that are african american	No	Continuous
percent of police that are hispanic	No	Continuous
percent of police that are asian	No	Continuous
percent of police that are minority of any kind	No	Continuous
number of officers assigned to special drug units	No	Continuous
number of different kinds of drugs seized	No	Continuous
police average overtime worked	No	Continuous
land area in square miles	Yes	Continuous
population density in persons per square mile	Yes	Continuous
percent of people using public transit for commuting	Yes	Continuous
number of police cars	No	Continuous
police operating budget	No	Continuous
percent of sworn full time police officers on patrol	No	Continuous
gang unit deployed	No	Categorical
percent of officers assigned to drug units	Yes	Continuous

Table 13: Crime Data. Summary of Selected Variables and Their Characteristics for Model Fitting *(continued)*

Variable	Included	Type
police operating budget per population	No	Continuous
total number of violent crimes per 100K popuation	Target	Continuous
pct african american * pct kids never married	Yes	Continuous
pct caucasian * pct kids never married	Yes	Continuous
pct asian * pct kids never married	Yes	Continuous
pct hispanic * pct kids never married	Yes	Continuous
pct foreign born * pct kids never married	Yes	Continuous
pct african american * pct divorced	Yes	Continuous
pct caucasian * pct divorced	Yes	Continuous
pct asian * pct divorced	Yes	Continuous
pct hispanic * pct divorced	Yes	Continuous
pct foreign born * pct divorced	Yes	Continuous

10.2 Figures

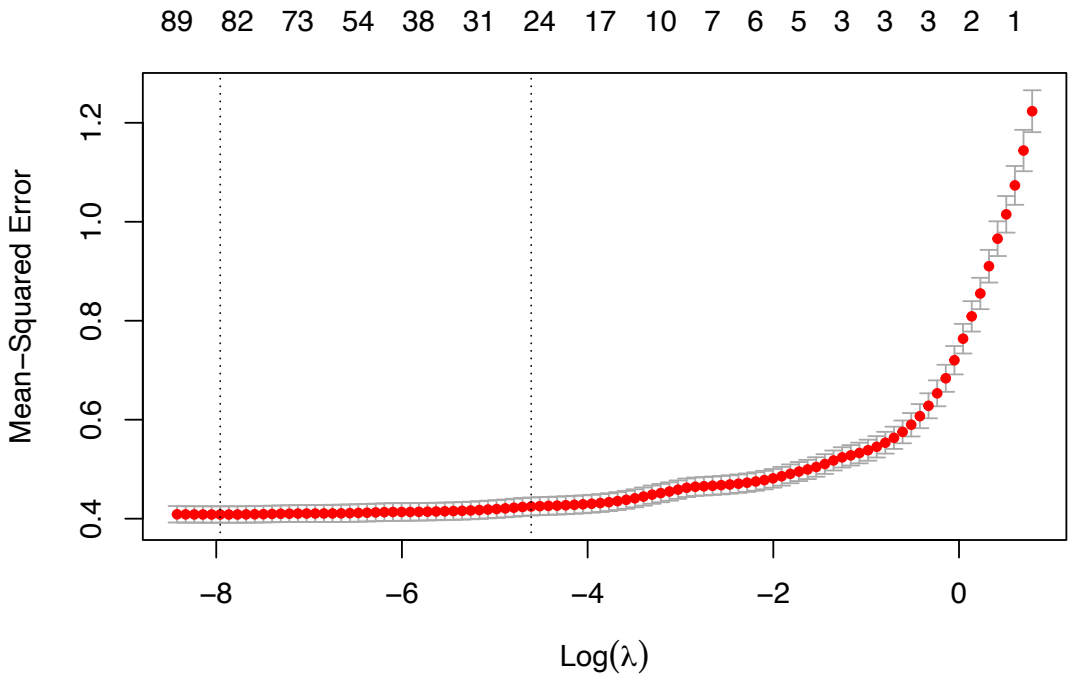

Figure 18: Crime Data. Cross-Validation Error as a Function of Lambda for Lasso

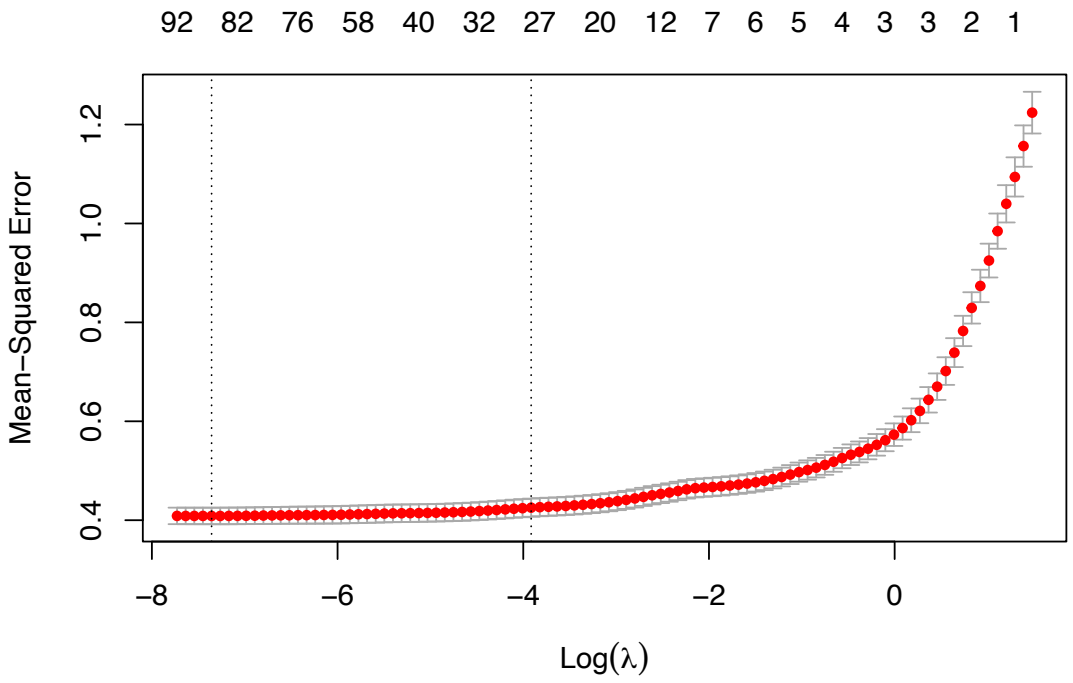

Figure 19: Crime Data. Cross-Validation Error as a Function of Lambda for Elastic-Net

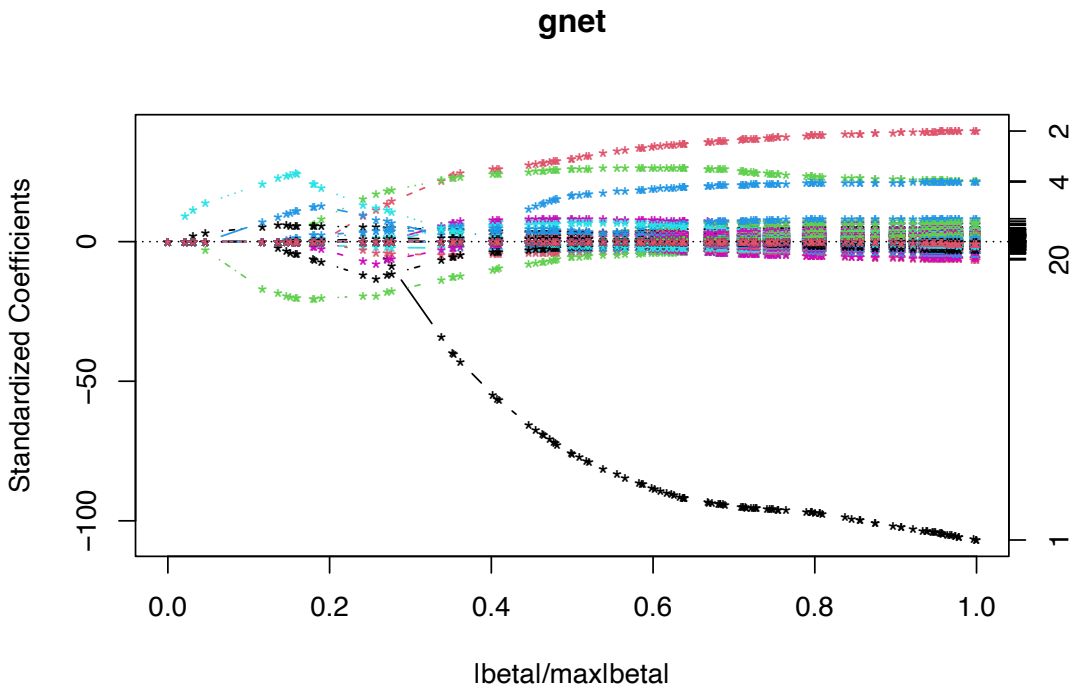

Figure 20: Crime Data. Spike-and-Slab Standardised Coefficients gnet Solution Path

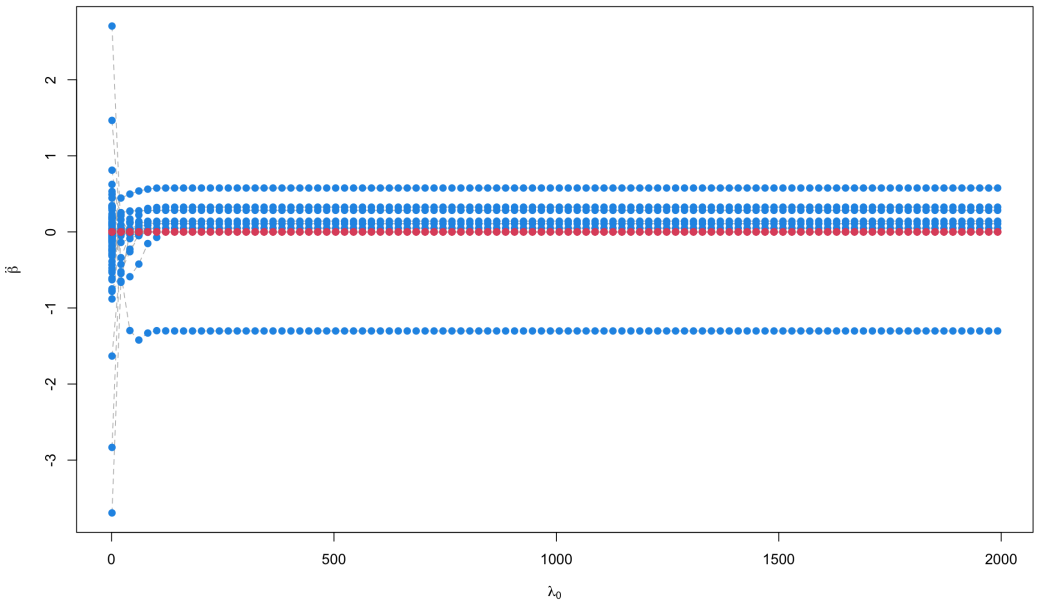

Figure 21: Crime Data. Spike-and-Slab Lasso Coefficients Paths

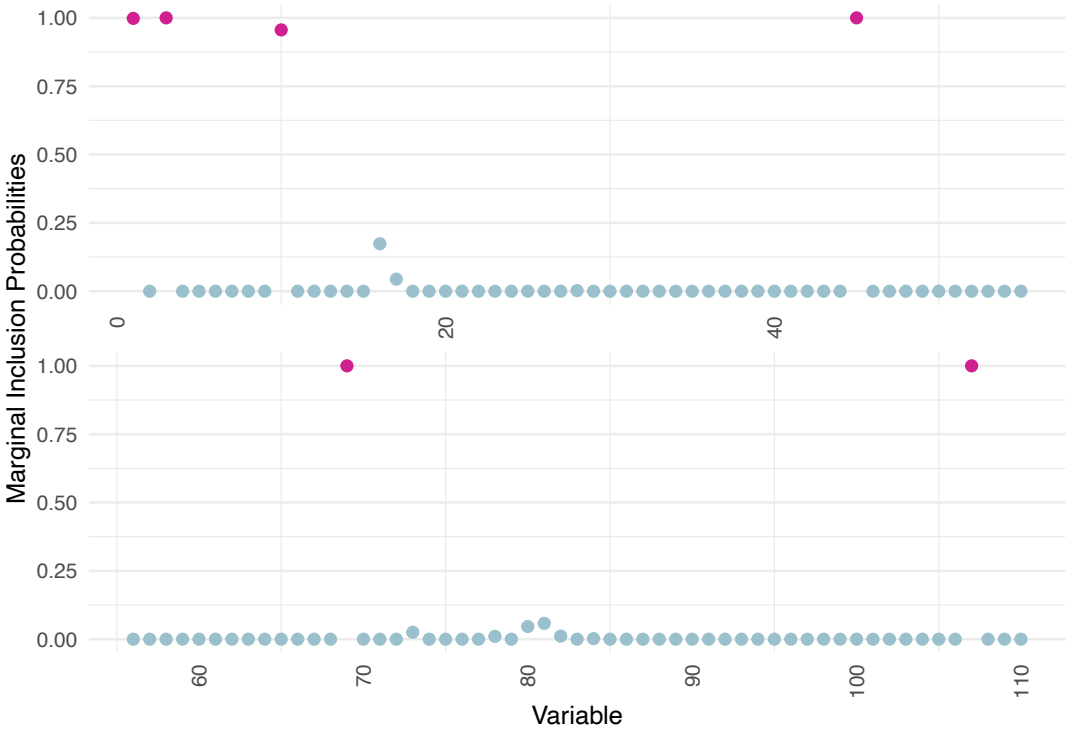

Figure 22: Crime Data. S5 Marginal Inclusion Probabilities

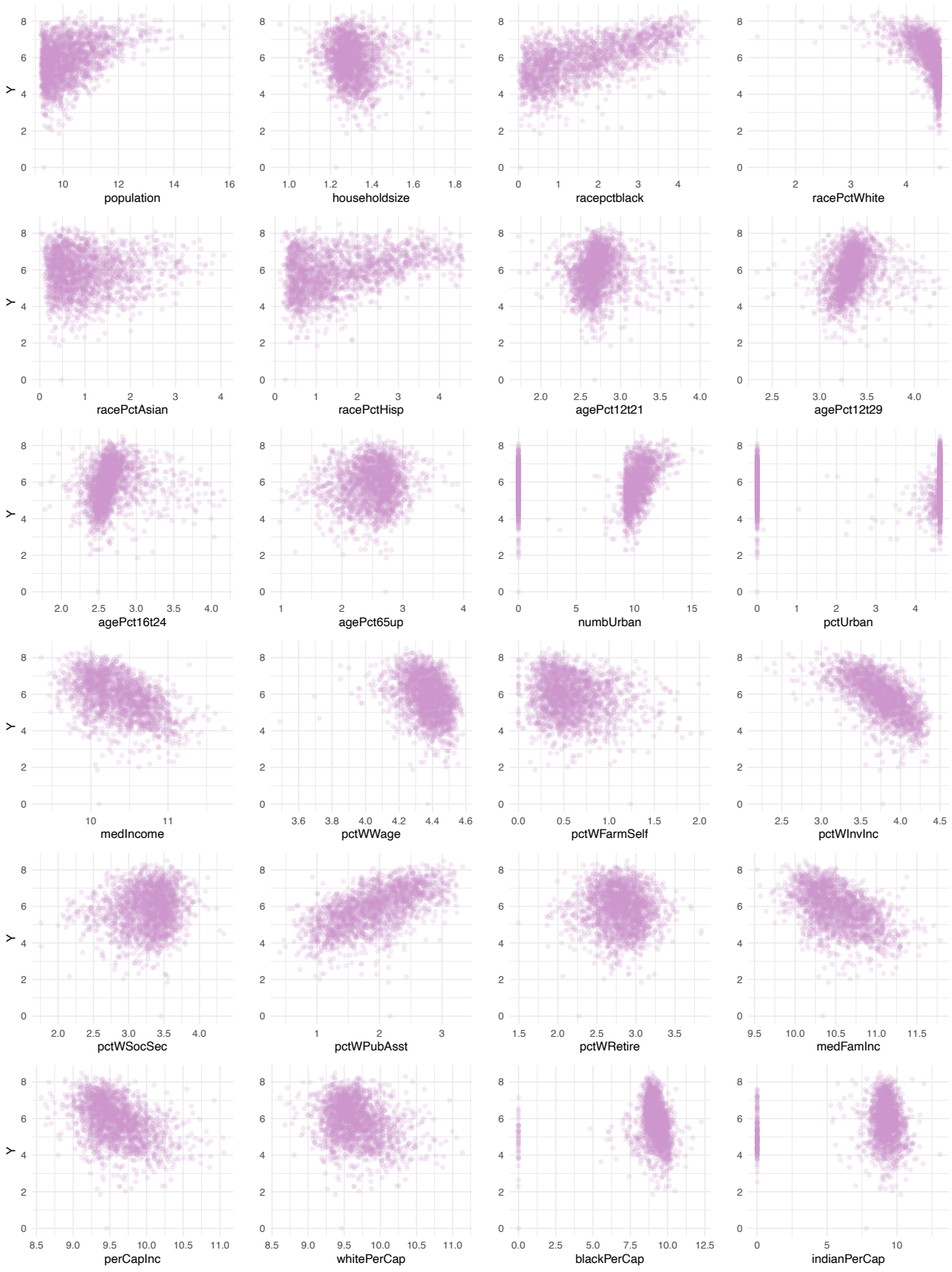

Figure 23: Crime Data. Full Scatter Plot Analysis of Linear Correlations Between Predictors and Target Variable Post Logarithmic Transformation (Over Multiple Pages)

Figure 24: Crime Data. Full Scatter Plot Analysis of Linear Correlations Between Predictors and Target Variable Post Logarithmic Transformation (Over Multiple Pages)

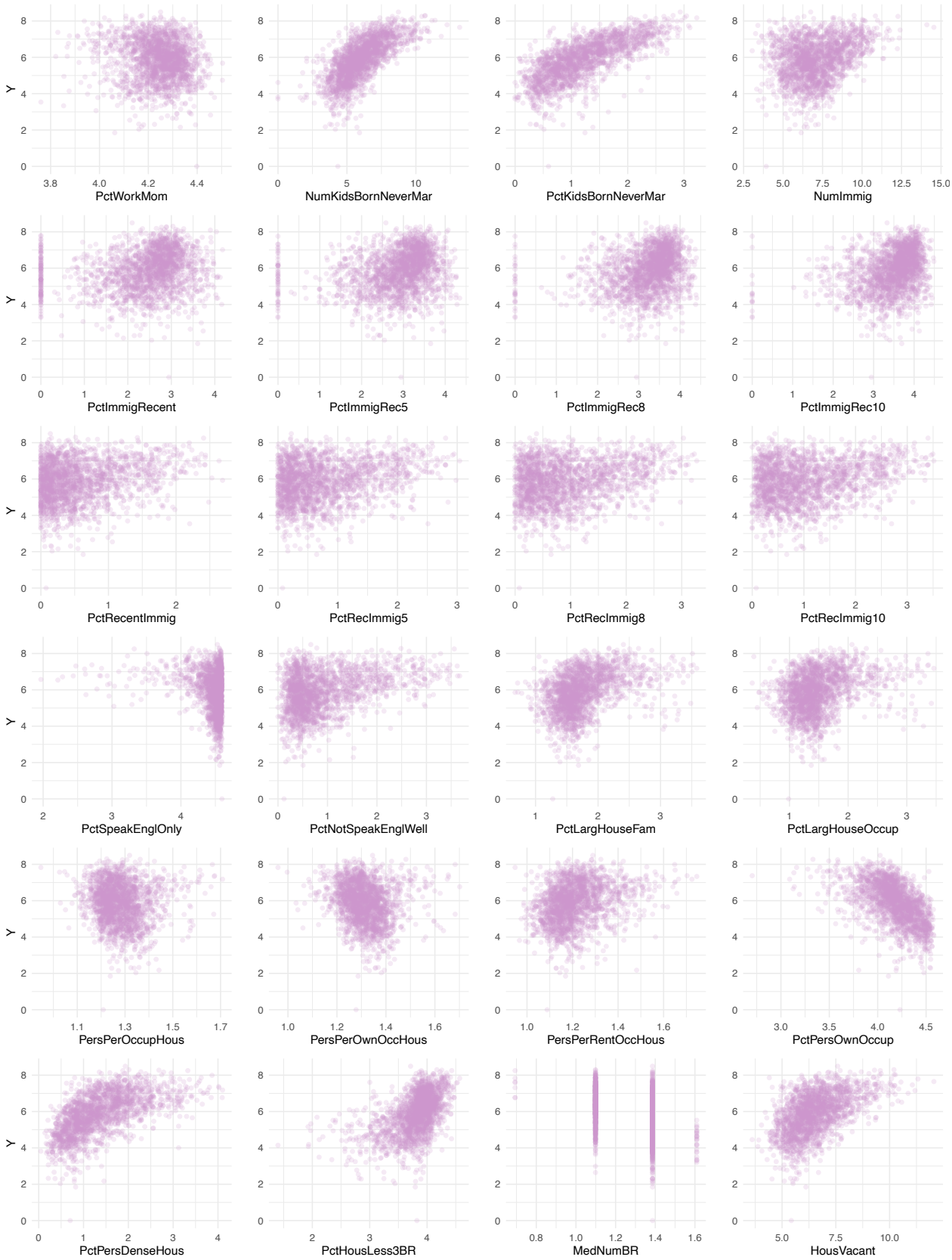

Figure 25: Crime Data. Full Scatter Plot Analysis of Linear Correlations Between Predictors and Target Variable Post Logarithmic Transformation (Over Multiple Pages)

Figure 26: Crime Data. Full Scatter Plot Analysis of Linear Correlations Between Predictors and Target Variable Post Logarithmic Transformation (Over Multiple Pages)

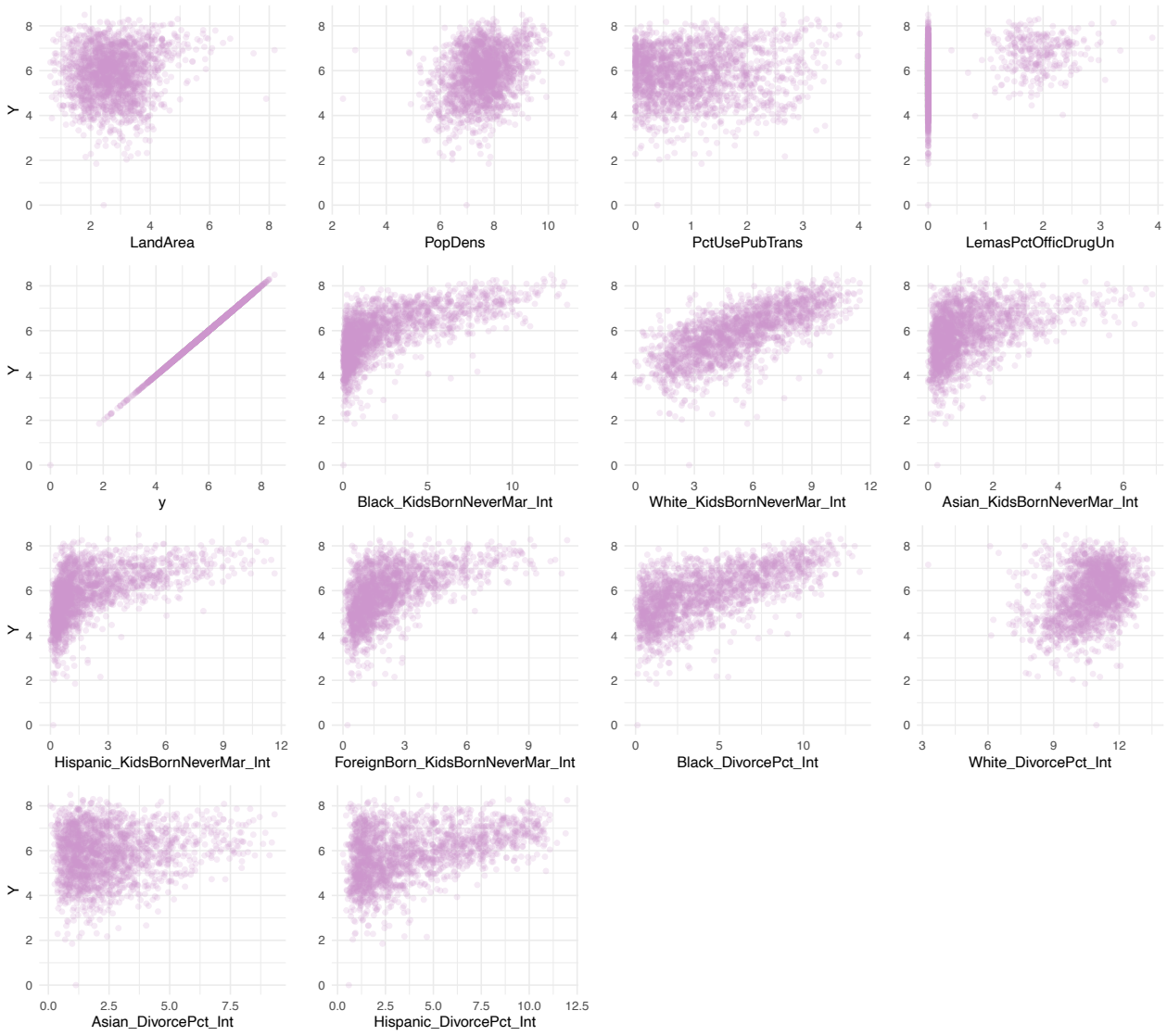

Figure 27: Crime Data. Full Scatter Plot Analysis of Linear Correlations Between Predictors and Target Variable Post Logarithmic Transformation (Over Multiple Pages)

10.3 Code

Note that the code should be run from the Analysis Execution part of the code, while other files should be sourced to run appropriately.

10.3.1 Data Simulation

The file should be saved as 'simulate_data.R'.

```
#### SIMULATE T1 UNCORRELATED CONTINUOUS DATA ####


# Function to simulate a T1 type data set
```

```r
# INPUT:
#       p - number of covariates.
#       n - number of data points to simulate.
#       sigma_e - variance of the error term.
#       seed - seed for random number generation.
#       standardise - whether to standardise the variables.
# OUTPUT:
#       sim_data - data frame with simulated data
simulate_T1 <- function(p, n, sigma_e = sqrt(15),
                        seed = 42, standardise = TRUE) {

  # Input validation
  if (!is.null(seed) && (!is.numeric(seed) || seed < 0 || seed %% 1 != 0)) {
    # Check if the seed is a non-negative integer.
    stop("seed must be a non-negative integer")
  }

  if (!is.numeric(p) || p <= 0 || floor(p) != p) {
    # Check if p is a positive integer.
    stop("p must be a positive integer")
  }

  if (!is.numeric(n) || n <= 0 || floor(n) != n) {
    # Check if n is a positive integer.
    stop("n must be a positive integer")
  }

  if (!is.numeric(sigma_e) || sigma_e <= 0) {
    # Check if sigma_e is a positive number.
    stop("sigma_e must be a positive number")
  }

  # Set seed if not provided
  if (!is.null(seed)) {
    set.seed(seed)
  }

  # Set the mean vector
  u_x <- rep(5, p)
```

```r
  # Set the correlation matrix as identity matrix
  sigma_x <- diag(p)

  # Generate the covariates X
  X <- MASS::mvrnorm(n, mu = u_x, Sigma = sigma_x)

  # Generate the true regression coefficients beta
  beta <- c(rep(3, 10), rep(5, 10), rep(0, p - 20))

  # Generate the error terms
  epsilon <- rnorm(n, mean = 0, sd = sigma_e)

  # Generate the response variable y
  y <- X %*% beta + epsilon

  # Combine X and y into a data frame
  sim_data <- as.data.frame(cbind(y, X))

  # Name the columns of the data frame
  colnames(sim_data) <- c("y", paste0("X", 1:p))

  if (standardise) {
    # Standardise only X variables, not y
    sim_data[ , -1] <- scale(sim_data[ , -1])
  }

  # Return the dataset
  return(sim_data)
}

# Extract the simulated data
# Simulate where p < n
T1_LD <- simulate_T1(p = 50, n = 200)
# Simulate where p = n
T1_ED <- simulate_T1(p = 100, n = 100)
# Simulate where p > n
T1_HD <- simulate_T1(p = 200, n = 150)
# Simulate where p >> n
```

```r
T1_VD <- simulate_T1(p = 200, n = 50)
# Simulate for XGBoost p << n
T1_XD <- simulate_T1(p = 50, n = 500)

#### SIMULATE T2 TEMPORAL CORRELATED CONTINUOUS DATA ####

# Function to simulate a T2 type data set
# INPUT:
#       p - number of covariates
#       n - number of data points to simulate
#       rho - AR(1) correlation coefficient
#       sigma_e - variance of the error term
#       seed - seed for random number generation
#       standardise - whether to standardise the variables.
# OUTPUT:
#       sim_data - data frame with simulated data
simulate_T2 <- function(p, n, rho = 0.8, sigma_e = sqrt(10),
                        seed = 42, standardise = TRUE) {

  # Input validation
  if (!is.null(seed) && (!is.numeric(seed) || seed < 0 || seed %% 1 != 0)) {
    # Check if the seed is a non-negative integer.
    stop("seed must be a non-negative integer")
  }

  if (!is.numeric(p) || p <= 0 || floor(p) != p) {
    # Check if p is a positive integer.
    stop("p must be a positive integer")
  }

  if (!is.numeric(n) || n <= 0 || floor(n) != n) {
    # Check if n is a positive integer.
    stop("n must be a positive integer")
  }

  if (!is.numeric(sigma_e) || sigma_e <= 0) {
    # Check if sigma_e is a positive number.
    stop("sigma_e must be a positive number")
  }
```

```r
if (!is.numeric(rho) || rho < -1 || rho > 1) {
  # Check if rho is a number between -1 and 1.
  stop("rho must be a number between -1 and 1")
}

# Set seed if not provided
if (!is.null(seed)) {
  set.seed(seed)
}

# Set the mean vector
u_x <- c(rep(3, 30), rep(7, p-30))

# Set the covariance matrix with AR(1) structure
sigma_x <- matrix(rho^abs(outer(1:p, 1:p, "-")), p, p)

# Generate the covariates X
X <- MASS::mvrnorm(n, mu = u_x, Sigma = sigma_x)

# Generate the true regression coefficients beta
beta <- c(seq(1, 20, 1), rep(0, p - 20))

# Generate the error terms
epsilon <- rnorm(n, mean = 0, sd = sigma_e)

# Generate the response variable y
y <- X %*% beta + epsilon

# Combine X and y into a data frame
sim_data <- as.data.frame(cbind(y, X))

# Name the columns of the data frame
colnames(sim_data) <- c("y", paste0("X", 1:p))

if (standardise) {
  # Standardise only X variables, not y
  sim_data[ , -1] <- scale(sim_data[ , -1])
}
```

```
  # Return the dataset
  return(sim_data)
}

# Simulate where p < n
T2_LD <- simulate_T2(p = 50, n = 200)
# Simulate where p = n
T2_ED <- simulate_T2(p = 100, n = 100)
# Simulate where p > n
T2_HD <- simulate_T2(p = 200, n = 150)
# Simulate where p >> n
T2_VD <- simulate_T2(p = 200, n = 50)
# Simulate for XGBoost p << n
T2_XD <- simulate_T2(p = 50, n = 500)

#### SIMULATE T3 MIXED CONTINUOUS AND CATEGORICAL DATA ####

# Function to simulate a T3 type data set with mixed continuous and
#     categorical variables, and some polynomials, interaction terms.
# INPUT:
#       p - number of continuous covariates (minimum of 10).
#       n - number of data points to simulate.
#       rho - AR(1) correlation coefficient.
#       sigma_e - variance of the error term.
#       seed - seed for random number generation.
#       standardise - whether to standardise the variables.
# OUTPUT:
#       sim_data - data frame with simulated data
simulate_T3 <- function(p, n, rho = 0.6, sigma_e = sqrt(12),
                        seed = 42, standardise = TRUE) {

  # Input validation
  if (!is.null(seed) && (!is.numeric(seed) || seed < 0 || seed %% 1 != 0)) {
    # Check if the seed is a non-negative integer.
    stop("seed must be a non-negative integer")
  }

  if (!is.numeric(p) || p <= 0 || floor(p) != p) {
```

```
  # Check if p is a positive integer.
  stop("p must be a positive integer")
}

if (!is.numeric(n) || n <= 0 || floor(n) != n) {
  # Check if n is a positive integer.
  stop("n must be a positive integer")
}

if (!is.numeric(sigma_e) || sigma_e <= 0) {
  # Check if sigma_e is a positive number.
  stop("sigma_e must be a positive number")
}

if (!is.numeric(rho) || rho < -1 || rho > 1) {
  # Check if rho is a number between -1 and 1.
  stop("rho must be a number between -1 and 1")
}

# Set seed if not provided
if (!is.null(seed)) {
  set.seed(seed)
}

# Calculate the number of continuous covariates needed
# 10 for 2 categorical, 4 interactions, and 4 polynomials
p <- p - 10

# Set the mean vector for continuous covariates
u_x <- c(rep(2, 10), rep(5, 30), rep(8, p - 40))

# Set the covariance matrix with AR(1) structure
sigma_x <- matrix(rho^abs(outer(1:p, 1:p, "-")), p, p)

# Generate the continuous covariates X
X <- MASS::mvrnorm(n, mu = u_x, Sigma = sigma_x)

# Generate binary categorical variable
cat_var1 <- sample(c(0, 1), n, replace = TRUE) %>%
```

```
  as.factor()
# Treat as ordinal categorical variable
cat_var2 <- sample(1:5, n, replace = TRUE) %>%
  as.factor()

# Generate interaction terms (multiplying first and second continuous covariate)
interaction_term_1_2 <- X[, 1] * X[, 2]
interaction_term_3_4 <- X[, 3] * X[, 4]
interaction_term_21_22 <- X[, 21] * X[, 22]
interaction_term_c1_22 <- interaction(cat_var1, X[, 22])

# Generate polynomial feature (squared third continuous covariate)
polynomial_feature_5_2 <- X[, 5]^2
polynomial_feature_6_3 <- X[, 6]^3
polynomial_feature_23_2 <- X[, 23]^2
polynomial_feature_23_3 <- X[, 23]^3

# Generate the true regression coefficients beta
beta <- c(rep(6, 5), rep(4, 5), rep(3, 5), rep(0, p - 15))

# Generate the error terms
epsilon <- rnorm(n, mean = 0, sd = sigma_e)

# Add the intercept too
intercept <- 2

# Define the betas
beta_cat_var1 <- 4
beta_it_1_2 <- 3
beta_p_23_2 <- 6

# Use zero for all other betas
beta_cat_var2 <- 0
beta_it_3_4 <- 0
beta_it_21_22 <- 0
beta_it_c1_22 <- 0
beta_p_5 <- 0
beta_p_6 <- 0
beta_p_23_3 <- 0
```

```r
# Generate the response variable y
y <- intercept + X %*% beta +
  beta_cat_var1 * as.numeric(cat_var1) +
  beta_cat_var2 * as.numeric(cat_var2) +
  beta_it_1_2 * interaction_term_1_2 +
  beta_it_3_4 * interaction_term_3_4 +
  beta_it_21_22 * interaction_term_21_22 +
  beta_it_c1_22 * as.numeric(interaction_term_c1_22) +
  beta_p_5 * polynomial_feature_5_2 +
  beta_p_6 * polynomial_feature_6_3 +
  beta_p_23_2 * polynomial_feature_23_2 +
  beta_p_23_3 * polynomial_feature_23_3 +
  epsilon

# Combine continuous covariates, categorical vars, interaction terms,
# polynomial features and y into a data frame
sim_data <- as.data.frame(cbind(y,
                                cat_var1,
                                cat_var2,
                                interaction_term_1_2,
                                interaction_term_3_4,
                                interaction_term_21_22,
                                interaction_term_c1_22,
                                polynomial_feature_5_2,
                                polynomial_feature_6_3,
                                polynomial_feature_23_2,
                                polynomial_feature_23_3, X))

# Make sure the categorical variables are factors
sim_data$cat_var1 <- as.factor(sim_data$cat_var1)
sim_data$cat_var2 <- as.factor(sim_data$cat_var2)

# Name the columns of the data frame
colnames(sim_data) <- c("y", "cat_var1", "cat_var2",
                        "interaction_1_2", "interaction_3_4",
                        "interaction_21_22", "interaction_term_c1_22",
                        "poly_5_2", "poly_6_3", "poly_23_2", "poly_23_3",
                        paste0("X", 1:p))
```

```r
  if (standardise) {
    # Select all variables except y, cat_var1, cat_var2
    variables_to_scale <- setdiff(colnames(sim_data), c("y", "cat_var1", "cat_var2"))

    # Apply scale() to these variables
    sim_data[variables_to_scale] <- scale(sim_data[variables_to_scale])
  }

  # Return the dataset
  return(sim_data)
}

# Simulate where p < n
T3_LD <- simulate_T3(p = 50, n = 200)
# Simulate where p = n
T3_ED <- simulate_T3(p = 100, n = 100)
# Simulate where p > n
T3_HD <- simulate_T3(p = 200, n = 150)
# Simulate where p >> n
T3_VD <- simulate_T3(p = 200, n = 50)
# Simulate for XGBoost p << n
T3_XD <- simulate_T3(p = 50, n = 500)

#### SIMULATE T4 GROUPED CONTINUOUS DATA WITH CATEGORICAL VARIABLES AND INTERACTIONS ####

# Function to simulate a T4 type data set
# INPUT:
#       p - number of continuous covariates.
#       n - number of data points to simulate.
#       rho - within-group correlation coefficient.
#       sigma_e - variance of the error term.
#       seed - seed for random number generation.
#       standardise - whether to standardise the variables.
# OUTPUT:
#       sim_data - data frame with simulated data
simulate_T4 <- function(p, n, rho = 0.6, sigma_e = sqrt(10),
                        seed = 42, standardise = TRUE) {
```

```r
# Input validation
if (!is.null(seed) && (!is.numeric(seed) || seed < 0 || seed %% 1 != 0)) {
  # Check if the seed is a non-negative integer.
  stop("seed must be a non-negative integer")
}

if (!is.numeric(p) || p <= 0 || floor(p) != p) {
  # Check if p is a positive integer.
  stop("p must be a positive integer")
}

if (!is.numeric(n) || n <= 0 || floor(n) != n) {
  # Check if n is a positive integer.
  stop("n must be a positive integer")
}

if (!is.numeric(sigma_e) || sigma_e <= 0) {
  # Check if sigma_e is a positive number.
  stop("sigma_e must be a positive number")
}

if (!is.numeric(rho) || rho < -1 || rho > 1) {
  # Check if rho is a number between -1 and 1.
  stop("rho must be a number between -1 and 1")
}

# Set seed if not provided
if (!is.null(seed)) {
  set.seed(seed)
}

# Adjust p to generate the correct number of continuous variables
p <- p - 5

# Group sizes
group_sizes <- rep(p / 5, 5)

# Covariance matrices for each group
cov_matrices <- lapply(1:5, function(i) {
```

```
  matrix(rho, nrow = group_sizes[i], ncol = group_sizes[i]) +
    diag(rep(1 - rho, group_sizes[i]))
})

# Set the means for each group
u_x <- rep(seq(2, 10, by = 2), times = group_sizes)

# Generating continuous covariates X
X <- do.call(cbind, lapply(1:length(cov_matrices), function(i) {
  MASS::mvrnorm(n, mu = rep(u_x[i], ncol(cov_matrices[[i]])), Sigma = cov_matrices[[i]])
}))

# Generate binary categorical variable
cat_var1 <- sample(c(0, 1), n, replace = TRUE) %>%
  as.factor()
# Treat as ordinal categorical variable
cat_var2 <- sample(1:5, n, replace = TRUE) %>%
  as.factor()

# Inclute 2 interaction terms
# Generate interaction terms
interaction_term_1_2_3 <- X[, 1] * X[, 2] * X[, 3]
interaction_term_4_5 <- X[, 4] * X[, 5]
interaction_term_16_17 <- X[, 16] * X[, 17]

# Generating true regression coefficients beta
beta <- c(rep(6, 5), rep(4, 5), rep(3, 5), rep(0, p - 15))

# Define the betas
beta_cat_var1 <- 4
beta_cat_var2 <- 0
beta_it_1_2_3 <- 3
beta_it_4_5 <- 0
beta_it_16_17 <- 0

# Generate the error terms
epsilon <- rnorm(n, mean = 0, sd = sigma_e)

# Generate the response variable y
```

```
y <- X %*% beta +
  beta_cat_var1 * as.numeric(cat_var1) +
  beta_cat_var2 * as.numeric(cat_var2) +
  beta_it_1_2_3 * interaction_term_1_2_3 +
  beta_it_4_5 * interaction_term_4_5 +
  beta_it_16_17 * interaction_term_16_17 +
  epsilon


# Combine continuous covariates and y into a data frame
# Combine continuous covariates, categorical vars, interaction terms,
# polynomial features and y into a data frame
sim_data <- as.data.frame(cbind(y,
                                cat_var1,
                                cat_var2,
                                interaction_term_1_2_3,
                                interaction_term_4_5,
                                interaction_term_16_17,
                                X))

# Make sure the categorical variables are factors
sim_data$cat_var1 <- as.factor(sim_data$cat_var1)
sim_data$cat_var2 <- as.factor(sim_data$cat_var2)

# Name the columns of the data frame
colnames(sim_data) <- c("y", "cat_var1", "cat_var2",
                        "interaction_term_1_2_3", "interaction_term_4_5",
                        "interaction_term_16_17",
                        paste0("X", 1:p))

if (standardise) {
  # Select all variables except y, cat_var1, cat_var2
  variables_to_scale <- setdiff(colnames(sim_data), c("y", "cat_var1", "cat_var2"))

  # Apply scale() to these variables
  sim_data[variables_to_scale] <- scale(sim_data[variables_to_scale])
}

# Return the dataset
```

```
  return(sim_data)
}

# Simulate where p < n
T4_LD <- simulate_T4(p = 50, n = 200)
# Simulate where p = n
T4_ED <- simulate_T4(p = 100, n = 100)
# Simulate where p > n
T4_HD <- simulate_T4(p = 200, n = 150)
# Simulate where p >> n
T4_VD <- simulate_T4(p = 200, n = 50)
# Simulate for XGBoost p << n
T4_XD <- simulate_T4(p = 50, n = 500)

# Remove functions
rm(simulate_T1, simulate_T2, simulate_T3, simulate_T4)
```

10.3.2 Implementing Statistical Methods: Function Definitions

The file should be saved as 'functions.R'.

```
#### PENALISED REGRESSION 'glmnet' ####

# Function to fit penalised regression on different datasets and extract the
#   selected predictors
# INPUTS:
#         data - a data frame containing the predictors and the response variable.
#                The response variable should be named "y".
#         nfolds - number of folds for cross-validation, default = 10.
#         alpha - numeric entry 1 for Lasso, 0.5 for Elastic Net.
# OUTPUT:
#         A list containing:
#               selected_predictors - a data frame with the predictors selected
#                                     by the model with their respective coefficients.
#               model_fit - the fitted glmnet model.
#               model_cv - cross-validated penalty model.
#
fit_glmnet <- function(data, nfolds = 10, alpha = 0.5) {

  # Input checks
  # Ensure data is a data.frame
```

```r
if (!is.data.frame(data)) {
  stop("Input 'data' must be a data frame.")
}

# Ensure 'y' is in the data
if (!"y" %in% names(data)) {
  stop("The response variable 'y' is not present in the input data.")
}

# Ensure 'nfolds' is a positive integer
if (!is.numeric(nfolds) || nfolds <= 0 || round(nfolds) != nfolds) {
  stop("'nfolds' must be a positive integer.")
}

# Ensure only Lasso or Elnet alpha values are fit
if (!is.numeric(alpha) || !(alpha %in% c(0.5, 1))) {
  stop("Alpha should be a numeric value of either 0.5 (Elastic Net) or 1 (Lasso).")
}

# Extract the target
y <- data$y

# Remove y
data <- data.frame(subset(data, select = -y))

# Combine intercept, variables
X <- model.matrix(~ . -1, data = data)

# Perform k-fold cross-validation to find the optimal value of the regularisation
#   parameter lambda that minimises the cross-validation error
set.seed(7)
model_cv <- cv.glmnet(x = X, y = y, alpha = alpha, nfolds = nfolds,
                      standardize = FALSE)

# Plot the cross-validation errors as a function of lambda.
plot(model_cv)
abline(v = log(model_cv$lambda.min), lwd = 4, lty = 2)

# Refit the model using the optimal lambda value obtained from cross-validation
```

```
  set.seed(7)
  model_fit <- glmnet(x = X, y = y, alpha = alpha, lambda = model_cv$lambda.min,
                      standardize = FALSE)

  # Extract the coefficients from the  model
  coefficients <- coef(model_fit, s = model_fit$lambda.min)

  # Find the names of the variables with non-zero coefficients
  selected_variable_names <- rownames(coefficients)[coefficients[, 1] != 0]

  # Extract the non-zero coefficients
  selected_predictors <- coefficients[selected_variable_names, 1] %>% data.frame()

  # Sort the selected_predictors in descending order by the absolute value of its column
  selected_predictors <- selected_predictors %>% arrange(desc(abs(selected_predictors[,1])))

  # Return the list of selected predictors and model itself
  return(list(selected_predictors = selected_predictors, model_fit = model_fit,
              model_cv = model_cv))
}

#### XGBOOST 'caret' ####

# Function to train and evaluate an XGBoost model from 'caret' package on different
#   datasets and plot feature importance
# INPUTS:
#         data - a data frame containing the predictors and the response variable.
#                The response variable should be named "y".
#         xgb_cv   - a trainControl object defining the cross-validation strategy.
#         xgb_grid - a data frame defining the grid of hyperparameters to search over.
# OUTPUT:
#         A list containing:
#               model - the trained XGBoost model.
#               rmse  - the root mean squared error of the model on the test set.
#               feature_importance - a data frame showing the importance of each
#                                    feature.
#               importance_plot - a plot object showing the feature importance.
#
fit_xgb <- function(data, xgb_cv = xgb_cv, xgb_grid = xgb_grid) {
```

```r
# Input validation
if (!is.data.frame(data)) {
  stop("data should be a data frame.")
}

# Check if data contains 'y' target
if (!"y" %in% names(data)) {
  stop("data should contain a column named 'y' as the response variable.")
}

# Extract the target
y <- data$y

# Remove y
data <- data.frame(subset(data, select = -y))

# Combine intercept, variables
X <- model.matrix(~ . + 0, data = data)

# Split the dataset into training and testing sets
# createDataPartition helps in creating stratified random samples
set.seed(42)
index <- createDataPartition(y, p = 0.8, list = FALSE)
# Extract training features
X_train <- X[index, ]
# Extract training target
y_train <- y[index]
# Extract testing features
X_test <- X[-index, ]
# Extract testing target
y_test <- y[-index]

# Train the XGBoost model with cross-validation and parameter tuning
xgb_model <- train(
  # Feature matrix
  x = X_train,
  # Target vector
  y = y_train,
```

```
    # Cross-validation strategy
    trControl = xgb_cv,
    # Grid of hyperparameters to tune
    tuneGrid = xgb_grid,
    # XGBoost model
    method = "xgbTree",
    metric = "RMSE",
    maximize = FALSE,
    # Specify the learning task and the corresponding learning objective
    objective = "reg:linear"
  )

  # Make predictions on the test set using the trained model
  predictions <- predict(xgb_model, X_test)

  # Calculate the Root Mean Squared Error (RMSE) on the test set
  # RMSE is a measure of the differences between predicted and actual values
  rmse <- sqrt(mean((predictions - y_test)^2))
  cat("Root Mean Squared Error on Test Set:", rmse, "\n")

  # Extract feature importance from the trained model
  # Feature importance helps in understanding which features are most influential
  #   in making predictions
  importance_matrix <- xgboost::xgb.importance(feature_names = colnames(X),
                                               model = xgb_model$finalModel)

  # Save the plot to an object so it can be returned
  importance_plot <- recordPlot(xgb.plot.importance(importance_matrix))

  # Return the model, feature importance dataframe, RMSE, and plot
  return(list(
    "model" = xgb_model,
    "feature_importance" = importance_matrix,
    "rmse" = rmse,
    "importance_plot" = importance_plot
  ))
}

#### SPIKE AND SLAB PRIOR 'spikeslab' ####
```

```r
# Function to fit a Spike and Slab prior model using 'spikeslab' package and
#   evaluate it on different datasets. It also plots the path of the
#   estimates for the Spike and Slab model.
#
# INPUTS:
#     data               - A data frame containing the predictors and the
#                              response variable.
#                          The response variable should be named "y".
#     bigp_smalln        - A logical indicating if the high-dimensional low
#                              sample size adjustments
#                          should be made. Should be either TRUE or FALSE.
#     bigp_smalln_factor - A numeric adjustment factor to be used when
#                              bigp.smalln is TRUE.
#     seed               - An NEGATIVE integer used for setting the seed for
#                              reproducibility.
#     screen             - Whether to first screen the variables, as defined
#                              in the package (in p>>n).
#
# OUTPUT:
#     A list containing:
#           model - The fitted Spike and Slab model.
#           results - A data frame containing selected variables with gnet,
#                     bma and stability metrics.
#
fit_spikeslab_prior <- function(data, bigp_smalln = FALSE, bigp_smalln_factor = 0,
                                screen = FALSE, seed = -42) {

  # Input validation
  if (!is.data.frame(data)) {
    stop("data should be a data frame.")
  }

  if (!"y" %in% names(data)) {
    stop("data should contain a column named 'y' as the response variable.")
  }

  if (!is.logical(bigp_smalln) || length(bigp_smalln) != 1) {
    stop("bigp_smalln should be a logical value (either TRUE or FALSE).")
```

```
}

if (!is.numeric(bigp_smalln_factor) || length(bigp_smalln_factor) != 1) {
  stop("bigp_smalln_factor should be a single numeric value.")
}

if (!is.logical(screen) || length(screen) != 1) {
  stop("screen should be a logical value (either TRUE or FALSE).")
}

if (!is.numeric(seed) || length(seed) != 1 || seed > 0 || seed != as.integer(seed)) {
  stop("seed should be a single negative integer value.")
}

# Extract the target
y <- data$y

# Remove y
data <- data.frame(subset(data, select = -y))

# Combine intercept, variables
X <- model.matrix(~ ., data = data)

# Run cv.spikeslab to get the stability measures
ss_results <- spikeslab::spikeslab(
  # Formula representing the relationship between predictors and response
  #formula,
  # The matrix containing the variables in the formula
  x = X,
  y = y,
  # The number of iterations in the two MCMC chains used in spikeslab.
  # n.iter1 Number of burn-in Gibbs sampled values (i.e., discarded values)
  #    and n.iter2 is for the number of Gibbs sampled values, following burn-in.
  n.iter1 = 1000,
  n.iter2 = 5000,
  # Calculate the mean squared error as part of the model evaluation
  #mse = TRUE,
  # High-dimensional low sample size adjustments.
  # bigp.smalln - logical flag, if TRUE adjustments for high-dimensional low
```

```
  #   sample size are made.
  # bigp.smalln.factor - controls the magnitude of the adjustments.
  bigp.smalln = bigp_smalln,
  bigp.smalln.factor = bigp_smalln_factor,
  # To screen the variables when p is big
  screen = screen,
  # If TRUE, an intercept term is included in the model
  intercept = TRUE,
  standardize = FALSE,
  # If TRUE, outputs progress and additional information while fitting the model
  verbose = TRUE,
  # Seed for random number generator, for reproducibility of results
  seed = seed,
  # Do not centre
  center = FALSE
)

# Extract bma values
bma_val <- data.frame(ss_results$bma)

# Extract gnet values
gnet_val <- data.frame(ss_results$gnet)

# Merge the data frames
results <- data.frame(
  Variable = rownames(bma_val),
  bma = bma_val[, 1],
  gnet = gnet_val[, 1]
)

# Order by stability descending
results <- results %>%
  dplyr::arrange(desc(abs(bma)))

# Plot the path of the estimates for the Spike and Slab model
plot(ss_results, plot.type = "path")

# Return the result
return(list(model = ss_results, results = results))
```

```
}



#### SPIKE-AND-SLAB LASSO 'SSLASSO' ####

# Function to fit the Spike-and-Slab LASSO model, plot the coefficients,
#   and extract selected variables from a given data frame.
#
# INPUTS:
#     data - Data frame where the first column is the response variable, and
#              the rest are predictors.
#     lambda1 - Slab variance parameter.
#     lambda0 - Vector of spike penalty parameters.
#     var - variance of error, unknown of fixed.
# OUTPUTS:
#     A list containing:
#         model - Final fitted SSLASSO model.
#         coefficients - The fitted matrix of coefficients for all variables
#                           and for each lambda.
#         ever_selected - A binary vector indicating which variables were
#                          ever selected along the regularization path.
#         plot - A plot of the coefficient paths for the fitted model.
#         selected_variables - A data frame of selected variable names and
#               their respective coefficients in descending order.
#
fit_sslasso <- function(data, lambda1 = 1,
                        lambda0 = seq(1, nrow(data), length.out = 100),
                        var = "unknown") {
  # Input checks
  # Check that 'data' is a data frame
  if (!is.data.frame(data)) {
    stop("'data' must be a data frame.")
  }

  # Check that 'lambda1' is a positive numeric value
  if (!is.numeric(lambda1) || lambda1 <= 0) {
    stop("'lambda1' must be a positive numeric value.")
```

```
}

# Check that 'lambda0' is a numeric sequence
if (!is.numeric(lambda0)) {
  stop("'lambda0' must be a numeric sequence.")
}

# Extract the target
y <- data$y

# Remove y
data <- data.frame(subset(data, select = -y))

# Combine intercept, variables
X <- model.matrix(~ ., data = data)

# Fit the SSLASSO model
set.seed(42)
result <- SSLASSO(X = X, y = y, penalty = "adaptive", variance = var,
                  lambda1 = lambda1, lambda0 = lambda0, warn = TRUE)

# Extract selection indicators and the names of the selected variables
selected_variables <- result$select
ever_selected <- apply(selected_variables, 1, max)
variable_names <- colnames(X)
selected_variable_indices <- which(ever_selected == 1)
selected_variable_names <- variable_names[selected_variable_indices]

# Get the coefficients of the selected variables
selected_coefficients <- result$beta[selected_variable_indices, ]
final_selected_coefficients <- selected_coefficients[, ncol(selected_coefficients)]

# Create a data frame of selected variable names and their coefficients,
#   sorted by absolute value of coefficient
selected_variables_df <- data.frame(Variable = selected_variable_names,
                                    Coefficient = final_selected_coefficients)
selected_variables_df <- selected_variables_df[order(abs(selected_variables_df$Coefficient),
                                                     decreasing = TRUE), ]
```

```
  # Return the results as a list
  return(list(model = result, coefficients = result$beta, ever_selected = ever_selected,
              plot = result, selected_variables = selected_variables_df))
}

#### HORSESHOE PRIOR. 'horseshoe' ####

# Function to fit the Horseshoe model, plot predicted values against observed values,
# and plot credible intervals for coefficients.
#
# INPUTS:
#     data - Data frame where the first column is the response variable, and the
#                rest are predictors.
#     method.tau - Method for handling tau (truncatedCauchy, halfCauchy, or fixed).
#     method.sigma - Method for handling sigma (Jeffreys or fixed).
#     burn - Number of burn-in MCMC samples.
#     nmc - Number of posterior draws to be saved.
#     thin - Thinning parameter of the chain.
#     alpha - Level for the credible intervals.
# OUTPUTS:
#     A list containing:
#       - model: The fitted horseshoe model.
#       - sel_var: The names of the selected variables in the model.
#       - plot_CI: Plot of credible intervals.
#       - plot_pred: Plot predicted values against the observed data.
#
fit_hs_horseshoe <- function(data, method.tau, method.sigma = "Jeffreys",
                             burn = 1000, nmc = 5000, thin = 1, alpha = 0.05){

  # Input checks
  # Ensure data is a data.frame
  if (!is.data.frame(data)) {
    stop("Input data must be a data frame.")
  }

  # Ensure method.tau is one of the allowed values
  if (!method.tau %in% c("truncatedCauchy", "halfCauchy", "fixed")) {
    stop("method.tau must be one of 'truncatedCauchy', 'halfCauchy', or 'fixed'.")
  }
```

```r
# Ensure tau is a positive number if method.tau is "fixed"
if (method.tau == "fixed" && (!is.numeric(tau) || tau <= 0)) {
  stop("tau must be a positive number when method.tau is 'fixed'.")
}

# Ensure method.sigma is one of the allowed values
if (!method.sigma %in% c("Jeffreys", "fixed")) {
  stop("method.sigma must be one of 'Jeffreys' or 'fixed'.")
}

# Ensure burn, nmc, and thin are positive integers
if (!is.numeric(burn) || burn <= 0 || floor(burn) != burn ||
    !is.numeric(nmc) || nmc <= 0 || floor(nmc) != nmc ||
    !is.numeric(thin) || thin <= 0 || floor(thin) != thin) {
  stop("burn, nmc, and thin must be positive integers.")
}

# Ensure alpha is a number between 0 and 1
if (!is.numeric(alpha) || alpha <= 0 || alpha >= 1) {
  stop("alpha must be a number between 0 and 1.")
}

# Extract the target
y <- data$y

# Remove y
data <- data.frame(subset(data, select = -y))

# Combine intercept, variables
X <- model.matrix(~ ., data = data)

# Fit the horseshoe model using the horseshoe package
set.seed(42)
fit_horseshoe <- horseshoe::horseshoe(y = y, X = X,
                                      method.tau = method.tau,
                                      method.sigma = method.sigma,
                                      burn = burn,
                                      nmc = nmc,
```

```
                                        thin = thin,
                                        alpha = alpha)

  # Plot predicted values against the observed data
  plot_pred <- plot(y, X %*% fit_horseshoe$BetaHat,
       xlab = "Observed values", ylab = "Predicted values")

  # Plot the credible intervals for coefficients
  plot_CI <- xYplot(Cbind(fit_horseshoe$BetaHat, fit_horseshoe$LeftCI,
                          fit_horseshoe$RightCI) ~ 1:ncol(X),
         ylab = "Coefficients", xlab = "Variables")

  # Use HS.var.select to get the selected variables
  selected <- HS.var.select(fit_horseshoe, y = y,
                            # Threshold left as default
                            method = "intervals", threshold = 0.5)

  # The variable names for the selected variables
  variable_names <- colnames(X)

  # Get the indices of the selected variables
  selected_indices <- which(selected == 1)

  # Get the names of the selected variables
  sel_var <- variable_names[selected_indices]

  # Return the fitted horseshoe model
  return(list(model = fit_horseshoe, sel_var = sel_var, plot_CI = plot_CI,
              plot_pred = plot_pred))
}

#### HORSESHOE PRIOR 'bayesreg' ####

# Function to fit the Horseshoe prior (or HS+) model with bayesreg package,
#   extract selected variables based on coefficient threshold, and refit
#   the model using only the selected variables.
#
# INPUTS:
#     data - Data frame where the first column is the response variable,
```

```r
#            and the rest are predictors.
#     n.samples - Number of posterior samples to draw.
#     burnin - Number of burn-in samples.
#     thin - Thinning parameter of the chain.
# OUTPUTS:
#     A list containing:
#       - model: Summary of the initial fitted horseshoe model.
#       - conf_intervals: Confidence intervals of the coefficients of the
#                         initial model.
#       - selected_variables: Names of the selected variables based on non-zero
#                             95% confidence intervals.
#
fit_horseshoe_bs <- function(data, n.samples = 5000, burnin = 1000,
                             thin = 1, prior = "hs") {

  # Input checks
  # Ensure data is a data.frame
  if (!is.data.frame(data)) {
    stop("Input 'data' must be a data frame.")
  }

  # Ensure 'n.samples' is a positive integer
  if (!is.numeric(n.samples) || n.samples <= 0 || round(n.samples) != n.samples) {
    stop("'n.samples' must be a positive integer.")
  }

  # Ensure 'burnin' is a positive integer
  if (!is.numeric(burnin) || burnin <= 0 || round(burnin) != burnin) {
    stop("'burnin' must be a positive integer.")
  }

  # Ensure 'thin' is a positive integer
  if (!is.numeric(thin) || thin <= 0 || round(thin) != thin) {
    stop("'thin' must be a positive integer.")
  }

  # Ensure 'prior' is a character and contains valid value
  if (!is.character(prior) || !(prior %in% c("hs", "hs+"))) {
    stop("'prior' must be a character and contain a valid value.")
```

```
  }

  # Fit the initial horseshoe model using the bayesreg package
  set.seed(42)
  fit_bayesreg <- bayesreg::bayesreg(y ~ ., data = data,
                                     # Distribution of the target
                                     model = "gaussian",
                                     prior = prior,
                                     n.samples = n.samples,
                                     burnin = burnin,
                                     thin = thin)

  # Generate the summary of the bayesreg model fit
  bayesreg_summary <- summary(fit_bayesreg)

  # Extract the confidence interval (CI) of the coefficients
  ci <- bayesreg_summary$CI.coef

  # Identify the coefficients whose 95% CI does not contain zero
  non_zero_ci_indicator <-  ifelse(ci[, 1] < 0 & ci [, 2] > 0, 0, 1)

  # Extract the variables (coefficients) whose 95% CI does not contain zero
  selected_variables <- names(non_zero_ci_indicator[non_zero_ci_indicator == 1])

  # Return the summary of the refitted model and selected variables
  return(list(model = bayesreg_summary, conf_intervals = ci,
              selected_variables = selected_variables))
}

#### SSS WITH SCREENING 'BayesS5' ####

# This function fits a sparse Bayesian linear regression model using the
#   BayesS5 package.
#   The S5 function promotes sparsity in the regression coefficients.
#
# Inputs:
#     data - Data frame where the first column is the response variable,
#            and the remaining columns are predictors.
#     ind_fun - A function to define the inclusion indicators of the model.
```

```r
#               Default: ind_fun_pimom.
#     model - An object of class Model defining the prior distribution.
#             Bernoulli_Uniform or Uniform. Default is Bernoulli_Uniform.
#     C0 - Normalisation constant for the S5 function.
#     type - a type of nonlocal priors; 'pimom' or 'pemom'.
#
# Outputs:
#     list containing the fitted model, model summary, and selected variables.
#
fit_S5 <- function(data, ind_fun = ind_fun_pimom, model = Bernoulli_Uniform,
                   C0 = 5, type = "pimom") {

  # Ensure data is a data frame
  if (!is.data.frame(data)) {
    stop("Input 'data' must be a data frame.")
  }

  # Ensure 'ind_fun' is a function
  if (!is.function(ind_fun)) {
    stop("'ind_fun' must be a function.")
  }

  # Check if 'type' is either 'pimom' or 'pemom'
  if (!(type %in% c("pimom", "pemom"))) {
    stop("'type' must be either 'pimom' or 'pemom'.")
  }

  # Check 'C0' is a positive numeric value
  if (!is.numeric(C0) || C0 <= 0) {
    stop("'C0' must be a positive numeric value.")
  }

  # Extract the target
  y <- data$y

  # Remove y
  data <- data.frame(subset(data, select = -y))

  # Combine intercept, variables
```

```r
  X <- model.matrix(~ .- 1, data = data)

  # Tuning parameters before fitting the model
  # Set seed for reproducibility
  set.seed(42)
  tuning <- hyper_par(type = type, X = X, y = y)

  # Fit the model using the S5 function from the BayesS5 package
  set.seed(42)
  fit_S5 <- BayesS5::S5(X = X, y = y, ind_fun = ind_fun, model = model,
                        tuning = tuning, C0 = C0)

  # Save the results of the model
  fit_S5_res <- result(fit_S5)

  # Extract the marginal probabilities
  marg_probs <- fit_S5_res$marg.prob %>% round(5)

  # Get the variable names
  var_names <- colnames(X)

  # Create a data frame with variable names and inclusion probabilities
  selected_variables <- data.frame(Variable = var_names, Included = marg_probs)

  # Return the fitted model, results summary and selected variables
  return(list(model = fit_S5, summary = fit_S5_res,
              selected_variables = selected_variables))
}


#### BAYESIAN LASSO 'monomvn' ####

# Function to fit a Bayesian LASSO regression model using the 'monomvn' package.
#   This function performs hyperparameter tuning and variable selection in a
#   Bayesian LASSO regression model. Includes the ability to perform RJMCMC.
#
# INPUTS:
#   data - A data frame where the first column is the response variable,
#          and the remaining columns are predictors.
```

```r
#   T - Number of iterations in the MCMC chain.
#   RJ - Logical flag indicating whether to perform Reversible Jump MCMC.
#   verb - Verbosity level of the function's output.
#   lambda2 - Value for penalty.
#   threshold - Threshold for variable selection based on the posterior
#               inclusion probabilities.
#   burnin - Burnin value.
#
# OUTPUTS:
#   A list containing:
#     model: The fitted Bayesian LASSO regression model.
#     var_prob: The posterior inclusion probabilities of each predictor.
#     sel_var_df: The variables that were selected by the model
#               based on the threshold with probabilities.
#
fit_blasso <- function(data,  T = 5000, RJ = TRUE, verb = 1, lambda2 = 1,
                       threshold = 0.5, burnin = 1000) {

  # Input validation
  # Check if the input data is of the correct format: a data frame
  if (!is.data.frame(data)) {
    stop("'data' must be a data frame.")
  }

  # Check if the number of iterations 'T' is a positive numeric value
  if (!is.numeric(T) || T <= 0) {
    stop("'T' must be a positive numeric value.")
  }

  # Check if the lambda2 is a positive numeric value
  if (!is.numeric(lambda2) || lambda2 <= 0) {
    stop("'lambda2' must be a positive numeric value.")
  }

  # Check if the burnin is a positive numeric value
  if (!is.numeric(burnin) || burnin <= 0) {
    stop("'burnin' must be a positive numeric value.")
  }
```

```r
# Check if the flag 'RJ' is a logical value
if (!is.logical(RJ)) {
  stop("'RJ' must be a logical value.")
}

# Check if the verbosity level 'verb' is a non-negative numeric value
if (!is.numeric(verb) || verb < 0) {
  stop("'verb' must be a non-negative numeric value.")
}

# Check if the threshold is a numeric value between 0 and 1
if (!is.numeric(threshold) || threshold < 0 || threshold > 1) {
  stop("'threshold' must be a numeric value between 0 and 1.")
}

# Extract the target
y <- data$y

# Remove y
data <- data.frame(subset(data, select = -y))

# Set seed for reproducibility outside the loop
set.seed(42)

# Fit the model on the full dataset
blasso_model <- monomvn::blasso(X = data, y = y, lambda2 = 1, RJ = RJ,
                                T = T, verb = verb, normalize = FALSE)

# Get the summary with burn-in
blasso_summary <- summary(blasso_model, burnin = burnin)

# Get the probabilities of each variable
var_prob <- blasso_summary$bn0

# Extract the variable names selected based on the threshold
sel_var <- colnames(data)[blasso_summary$bn0 > threshold]

# Extract the variable probabilities selected based on the threshold
sel_var_prob <- var_prob[blasso_summary$bn0 > threshold]
```

```
  # Create a data frame with variable names and respective probabilities
  sel_var_df <- data.frame(Variable = sel_var, Probability = sel_var_prob)

  # Order data frame by probability in descending order
  sel_var_df <- sel_var_df[order(-sel_var_df$Probability), ]

  # Return the fitted model, variable inclusion probabilities,
  #   selected variables above a threshold
  return(list(model = blasso_model, var_prob = var_prob,
              sel_var_df = sel_var_df))
}
```

10.3.3 Data Preparation: Crime Dataset

The file should be saved as 'data_crime_raw.R'.

```
#### SET UP ####

# Combine the list of libraries from both scripts
library_list <- c("tidyverse", "corrplot", "glmnet", "cowplot", "car", "MASS",
                  "caret", "spikeslab", "SSLASSO", "horseshoe", "bayesreg",
                  "Hmisc", "BayesS5", "monomvn", "gridExtra", "ggpubr")

# Un-comment the following lines if you need to install the packages
# for (i in library_list) {
#   install.packages(i, character.only = TRUE)
# }

# Load the libraries
for (i in library_list) {
  library(i, character.only = TRUE)
}

# Set working directory (assuming you want to set it to the 'main' directory)
setwd("~/Documents")

# Remove unwanted objects
rm(library_list, i)

#### READ IN AND WRANGLE DATA ####
```

```
# Read the text file into a dataframe
df <- read.csv('~/Documents/CommViolPredUnnormalizedData.txt')

# Rename the columns
# Function to rename columns of the data frame
# INPUT:
#       df - data frame with old column names.
# OUTPUT:
#       df - data frame with new column names.
rename_columns <- function(df) {
  new_column_names <- c("communityname", "state", "countyCode", "communityCode", "fold",
                        "population", "householdsize", "racepctblack", "racePctWhite",
                        "racePctAsian", "racePctHisp", "agePct12t21", "agePct12t29",
                        "agePct16t24", "agePct65up", "numbUrban", "pctUrban",
                        "medIncome", "pctWWage", "pctWFarmSelf", "pctWInvInc",
                        "pctWSocSec", "pctWPubAsst", "pctWRetire", "medFamInc",
                        "perCapInc", "whitePerCap", "blackPerCap", "indianPerCap",
                        "AsianPerCap", "OtherPerCap", "HispPerCap", "NumUnderPov",
                        "PctPopUnderPov", "PctLess9thGrade", "PctNotHSGrad",
                        "PctBSorMore", "PctUnemployed", "PctEmploy", "PctEmplManu",
                        "PctEmplProfServ", "PctOccupManu", "PctOccupMgmtProf",
                        "MalePctDivorce", "MalePctNevMarr", "FemalePctDiv", "TotalPctDiv",
                        "PersPerFam", "PctFam2Par", "PctKids2Par", "PctYoungKids2Par",
                        "PctTeen2Par", "PctWorkMomYoungKids", "PctWorkMom",
                        "NumKidsBornNeverMar", "PctKidsBornNeverMar", "NumImmig",
                        "PctImmigRecent", "PctImmigRec5", "PctImmigRec8", "PctImmigRec10",
                        "PctRecentImmig", "PctRecImmig5", "PctRecImmig8", "PctRecImmig10",
                        "PctSpeakEnglOnly", "PctNotSpeakEnglWell", "PctLargHouseFam",
                        "PctLargHouseOccup", "PersPerOccupHous", "PersPerOwnOccHous",
                        "PersPerRentOccHous", "PctPersOwnOccup", "PctPersDenseHous",
                        "PctHousLess3BR", "MedNumBR", "HousVacant", "PctHousOccup",
                        "PctHousOwnOcc", "PctVacantBoarded", "PctVacMore6Mos",
                        "MedYrHousBuilt", "PctHousNoPhone", "PctWOFullPlumb",
                        "OwnOccLowQuart", "OwnOccMedVal", "OwnOccHiQuart",
                        "OwnOccQrange", "RentLowQ", "RentMedian", "RentHighQ",
                        "RentQrange", "MedRent", "MedRentPctHousInc", "MedOwnCostPctInc",
                        "MedOwnCostPctIncNoMtg", "NumInShelters", "NumStreet",
                        "PctForeignBorn", "PctBornSameState", "PctSameHouse85",
```

```
                        "PctSameCity85", "PctSameState85", "LemasSwornFT",
                        "LemasSwFTPerPop", "LemasSwFTFieldOps", "LemasSwFTFieldPerPop",
                        "LemasTotalReq", "LemasTotReqPerPop", "PolicReqPerOffic",
                        "PolicPerPop", "RacialMatchCommPol", "PctPolicWhite",
                        "PctPolicBlack", "PctPolicHisp", "PctPolicAsian", "PctPolicMinor",
                        "OfficAssgnDrugUnits", "NumKindsDrugsSeiz", "PolicAveOTWorked",
                        "LandArea", "PopDens", "PctUsePubTrans", "PolicCars",
                        "PolicOperBudg", "LemasPctPolicOnPatr", "LemasGangUnitDeploy",
                        "LemasPctOfficDrugUn", "PolicBudgPerPop", "murders", "murdPerPop",
                        "rapes", "rapesPerPop", "robberies", "robbbPerPop", "assaults",
                        "assaultPerPop", "burglaries", "burglPerPop", "larcenies",
                        "larcPerPop", "autoTheft", "autoTheftPerPop", "arsons",
                        "arsonsPerPop", "ViolentCrimesPerPop", "nonViolPerPop")
  # Rename columns
  colnames(df) <- new_column_names
  return(df)
}

# Renaming columns
df <- rename_columns(df)

# Replace "?" with NA in the entire data frame
df[df == "?"] <- NA

# Calculate number of NA's in each column
columns_with_na <- colSums(is.na(df))

# There are many variables that have a great deal of missing values,
  # they are also not very relevant to the analysis here,
  # hence, shall delete them altogether
columns_with_na <- names(columns_with_na[columns_with_na > 221])

# Delete columns with more than 13 NA from data frame
df <- df[, !(names(df) %in% columns_with_na)]

# As there are still missing data (0 to 13 values in some columns)
  # find which specific rows have missing data
rows_with_na <- rowSums(is.na(df)) > 0
# Save these rows in a data frame
```

```
rows_with_na_indices <- df[rows_with_na, ]

# Check which columns have missing data and save them
cols_with_na <- colSums(is.na(rows_with_na_indices))

# Variables that hold information on specific convictions should be deleted
  # as the overall Violent Crimes number will be the target variable
# List of column names to be removed
cols_to_remove <- c("countyCode", "communityCode", "rapes", "rapesPerPop",
                    "robberies", "robbbPerPop", "assaults", "assaultPerPop",
                    "burglaries", "burglPerPop", "larcenies", "larcPerPop",
                    "autoTheft", "autoTheftPerPop", "arsons", "arsonsPerPop",
                    "nonViolPerPop", "fold", "murders", "murdPerPop",
                    "OwnOccQrange", "RentQrange", "state")

# Remove columns
df <- df[, !(names(df) %in% cols_to_remove)]

# Make sure that each column that has been left is numeric
df[, -1] <- sapply(df[, -1], as.numeric)

# Create a logical vector indicating which columns have NA's
  # Also keep the 'communityname' column regardless as it indicates the row
cols_with_na_logical <- (cols_with_na > 0) |
  (colnames(rows_with_na_indices) == "communityname")

# Subset the data frame to retain only the columns with NA's and 'communityname'
  # as to figure out what is missing in the data set and if these variables
  # or rows can be deleted altogether
df_with_na_cols_with_na <- rows_with_na_indices[, cols_with_na_logical]

# Remove the rows with missing data
df <- df[complete.cases(df), ]

# Remove community names from the main df
df <- df[, -1]

# Rename the target
df <- rename(df, y = ViolentCrimesPerPop)
```

```r
# Remove unwanted objects
rm(df_with_na_cols_with_na, rows_with_na_indices, cols_to_remove, cols_with_na,
     cols_with_na_logical, columns_with_na, rows_with_na, rename_columns)

#### TRANSFORMING DATA ####

# Transform all data
df_t <- log(df + 1)

#### ADDING INTERACTION TERMS ####

# As from the linear model and correlation matrix, parents who have not married,
#   is an important predictor. Interesting to see a similar relationship
#   for the divorce rates too.
# Kids born to parents who are not married against demographic groups
df_t$Black_KidsBornNeverMar_Int <- df_t$racepctblack * df_t$PctKidsBornNeverMar
df_t$White_KidsBornNeverMar_Int <- df_t$racePctWhite * df_t$PctKidsBornNeverMar
df_t$Asian_KidsBornNeverMar_Int <- df_t$racePctAsian * df_t$PctKidsBornNeverMar
df_t$Hispanic_KidsBornNeverMar_Int <- df_t$racePctHisp * df_t$PctKidsBornNeverMar
df_t$ForeignBorn_KidsBornNeverMar_Int <- df_t$PctForeignBorn * df_t$PctKidsBornNeverMar

# Divorced against demographic groups
df_t$Black_DivorcePct_Int <- df_t$racepctblack * df_t$TotalPctDiv
df_t$White_DivorcePct_Int <- df_t$racePctWhite * df_t$TotalPctDiv
df_t$Asian_DivorcePct_Int <- df_t$racePctAsian * df_t$TotalPctDiv
df_t$Hispanic_DivorcePct_Int <- df_t$racePctHisp * df_t$TotalPctDiv
df_t$ForeignBorn_DivorcePct_Int <- df_t$PctForeignBorn * df_t$TotalPctDiv

# Save a standardised version to check if different variables are selected
df_st <- scale(df_t) %>% data.frame()


#### EXPLORATORY ANALYSIS ####

## PLOTTING HISTOGRAMS ##

# Create a list to hold the plots
hist_list <- list()
```

```r
# Loop over the columns of the dataframe
for (i in 1:(ncol(df))) {
  # Base plot
  p <- ggplot(df, aes_string(names(df)[i])) +
    geom_histogram(fill = "gray99", color = "lightblue3", size = 0.4, alpha = 0.5) +
    theme_minimal() +
    theme(
      # Adjust font size
      axis.title.x = element_text(size = 13),
      axis.title.y = element_text(size = 13)
    )

  # Only add y label if it's the first in a set of 3
  if (i %% 3 == 1) {
    p <- p + ylab("Count")
  } else {
    p <- p + ylab("")
  }

  # Save the plot to the list
  hist_list[[i]] <- p
}


## PLOTTING HISTOGRAMS TRANSFORMED DATA ##

# Create a list to hold the plots
hist_t_list <- list()

# Loop over the columns of the dataframe
for (i in 1:(ncol(df_t))) {
  # Base plot
  p <- ggplot(df_t, aes_string(names(df_t)[i])) +
    geom_histogram(fill = "gray99", color = "lightblue3", size = 0.4, alpha = 0.5) +
    theme_minimal() +
    theme(
      # Adjust font size
      axis.title.x = element_text(size = 13),
```

```
      axis.title.y = element_text(size = 13)
    )

  # Only add y label if it's the first in a set of 3
  if (i %% 3 == 1) {
    p <- p + ylab("Count")
  } else {
    p <- p + ylab("")
  }

  # Save the plot to the list
  hist_t_list[[i]] <- p
}



## PLOT INTERACTIONS WITH THE PREDICTION ##

# Create a list to hold the plots
scatter_t_list <- list()

# Loop over the columns of the dataframe
for (i in 1:(ncol(df_t)-1)) {
  # Create the scatter plot for column i vs prediction variable
  p <- ggplot(df_t, aes_string(x = names(df_t)[i], y = "y")) +
    geom_point(colour = "plum3", alpha = 0.2) +
    theme(
      # Adjust font size
      axis.title.x = element_text(size = 17)
    ) +
    # Minimal theme of plot
    theme_minimal()

  # Only add y label if it's the first in a set of 3
  if (i %% 4 == 1) {
    p <- p + ylab("Y")
  } else {
    p <- p + ylab("")
  }
```

```
  scatter_t_list[[i]] <- p
}


## PLOT BOXPLOTS ##

# Create a list to hold the plots
boxplot_t_list <- list()

# Loop over the columns of the dataframe
for (i in 1:(ncol(df_t))) {
  # Create the boxplot for column i
  boxplot_t_list[[i]] <- ggplot(df_t, aes_string(x = 1, y = names(df_t)[i])) +
    geom_boxplot(fill = "lightblue3", colour = "gray20", size = 0.1,
                 alpha = 0.4) +
    # Removing the x-label as it's not meaningful here
    xlab("") +
    theme_minimal() +
    theme(axis.text.x = element_blank())  # Remove x-axis text
}

## PLOT CORRELATION MATRIX ##

# Calculate the correlation matrix
cor_matrix <- cor(df_t)

# Dendrogram
hc <- hclust(dist(1 - cor_matrix))
# plot(hc)
# split(names(clusters), clusters)


## MULTICOLLINEARITY CHECK ##

# Fit a linear model
model <- lm(y ~ ., data = df)

# Calculate VIF
```

```r
vif_values <- vif(model)

## NORMALITY CHECKS ##

# Apply Shapiro-Wilk test to each column
p_values <- apply(df_t, 2, function(x) shapiro.test(x)$p.value)

# Create a data frame of the results
results <- data.frame(Variable = names(p_values), P_Value = p_values)
#any(results$P_Value > 0.05)

## MISSING VALUES ##

check_data <- function(data) {
  # Loop over columns
  for (col in names(data)) {
    # Check if the column is numeric
    if (!is.numeric(data[[col]])) {
      stop(paste("Column", col, "is not numeric."))
    }
    # Check for missing values
    if (anyNA(data[[col]])) {
      stop(paste("Column", col, "contains missing values."))
    }
  }
  # If no problems found, print a success message
  print("All columns are numeric and contain no missing values.")
}

# Does the data have any missing values?
#df_t_missing <- check_data(df_t)

#### OUTLIERS ####

# Define a function to detect outliers based on the IQR
# INPUTS:
#         data - the data frame containing the data
#         columns - the columns in which to look for outliers
#         factor - the factor to multiply the IQR by to find the bounds (default 2)
```

```r
# OUTPUT:
#         outlier_indices - indices of outliers.
#
detect_outliers_iqr <- function(data, columns, factor = 2){

  # Initialize a vector to hold the indices of outlier rows
  outlier_indices <- c()

  # Loop over each specified column
  for(col in columns){

    # Calculate the first quartile (25th percentile)
    Q1 <- quantile(data[[col]], 0.25, na.rm = TRUE)

    # Calculate the third quartile (75th percentile)
    Q3 <- quantile(data[[col]], 0.75, na.rm = TRUE)

    # Calculate the interquartile range (IQR)
    IQR <- Q3 - Q1

    # Calculate the lower bound for what will be considered an outlier
    lower_bound <- Q1 - factor * IQR

    # Calculate the upper bound for what will be considered an outlier
    upper_bound <- Q3 + factor * IQR

    # Identify the indices of rows where the column value is an outlier
    outliers <- which(data[[col]] < lower_bound | data[[col]] > upper_bound)

    # Add the indices of the outliers to the list of outlier indices
    outlier_indices <- c(outlier_indices, outliers)
  }

  # Return the unique outlier indices
  return(unique(outlier_indices))
}

# Call the detect_outliers_iqr function, passing the data frame and column names
#   of the numeric columns.
```

```
outlier_indices <- detect_outliers_iqr(df_t, names(df_t))

# Print the number of outliers and their indices
#cat(paste0("Number of outliers detected: ", length(outlier_indices)), "\n")
#cat(paste0("Outlier indices: ", outlier_indices), "\n")

# Remove the outliers from the data frame by subsetting the data frame to exclude
#   these rows.
# The negative sign before outlier_indices means "all rows EXCEPT these indices".
df_no_outliers <- df_t[-outlier_indices, ]

# Remove the unwanted data
rm(df_no_outliers, p_values, results, vif_values, i, check_data,
   detect_outliers_iqr, scatter_list, model, outlier_indices,
   df, model, df_t_missing)
```

10.3.4 Analysis Execution

Main working file should be saved as 'main.R'.

```
#### SET UP ####

# Combine the list of libraries from both scripts
library_list <- c("tidyverse", "corrplot", "glmnet", "cowplot", "car", "MASS",
                  "caret", "spikeslab", "SSLASSO", "horseshoe", "bayesreg",
                  "Hmisc", "BayesS5", "monomvn", "gridExtra", "ggpubr")

# Un-comment the following lines if you need to install the packages
# for (i in library_list) {
#   install.packages(i, character.only = TRUE)
# }

# Load the libraries
for (i in library_list) {
  library(i, character.only = TRUE)
}

# Set working directory (assuming you want to set it to the 'main' directory)
setwd("~/Documents")

# Remove unwanted objects
```

```r
rm(library_list, i)

#### SOURCE SIMULATED DATA ####

# Source the file that contains the simulation functions
source("simulate_data.R")

#### SOURCE FUNCTIONS ####

# Source the file that contains the simulation functions
source("functions.R")

#### SIM DATA. LASSO AND ELASTIC NET FITTING 'glmnet' ####

# Extract BOTH the Lasso and Elastic net penalisation models using all types
#   of simulated data

# List of dataset prefixes
prefixes <- c("T1", "T2", "T3", "T4")
# List of dataset suffixes
suffixes <- c("LD", "ED", "HD", "VD")
# Initialize an empty list to store the models
models_list_glmnet <- list()

# Loop over prefixes and suffixes
for (prefix in prefixes) {
  for (suffix in suffixes) {
    # Construct the dataset name
    dataset_name <- paste(prefix, suffix, sep = "_")

    # Access the dataset from the global environment
    dataset <- get(dataset_name)

    # Fit the model for each alpha value (0.5 and 1)
    for (alpha in c(0.5, 1)) {
      # Fit the model
      model <- fit_glmnet(dataset, alpha = alpha)

      # Generate a model name
```

```
      model_name <- paste(prefix, suffix, ifelse(alpha == 0.5, "elnet", "lasso"), sep = "_")

      # Store the model in the list
      models_list_glmnet[[model_name]] <- model
    }
  }
}

#### SIM DATA. XGBOOST 'caret' ####

# Set the initial parameters for XGBoost
# Define an extensive grid for hyperparameter tuning
# This grid consists of multiple values for each parameter, allowing for more refined tuning
xgb_grid <- expand.grid(
  # Number of boosting rounds
  nrounds = c(50, 100, 150),
  # Maximum depth of the trees
  max_depth = c(3, 5, 7, 9),
  # Learning rate
  eta = c(0.01, 0.1, 0.3),
  # Minimum loss reduction required
  gamma = c(0, 0.1, 1),
  # Fraction of features to be randomly sampled for each tree
  colsample_bytree = c(0.6, 0.8, 1),
  # Minimum sum of instance weight needed in a leaf
  min_child_weight = c(1, 3, 5),
  # Fraction of observations to be randomly sampled for each tree
  subsample = c(0.8, 1))

# Define cross-validation strategy
# This helps in assessing the model's performance in an unbiased
#   way using a subset of the data
xgb_cv <- trainControl(
  # Repeated cross-validation
  method = "repeatedcv",
  # Number of folds
  number = 5,
  # Number of complete sets of folds to compute
  repeats = 3,
```

```
  # Display training progress
  verboseIter = TRUE,
  # Do not return the training data
  returnData = FALSE,
  # Save all resampling scores
  returnResamp = "all",
  # Allow parallel processing
  allowParallel = TRUE)

# Fit the function
# Initialize an empty list to store the models
models_xgboost_list <- list()

# Loop over prefixes and suffixes
for (prefix in prefixes) {
  for (suffix in suffixes) {
    # Construct the dataset name
    dataset_name <- paste(prefix, suffix, sep = "_")

    # Access the dataset from the global environment
    dataset <- get(dataset_name)

    # Fit the model
    model <- fit_xgboost(dataset)

    # Generate a model name
    model_name <- paste(prefix, suffix, "xgboost", sep = "_")

    # Store the model in the list
    models_xgboost_list[[model_name]] <- model
  }
}

#### SIM DATA. SPIKE AND SLAB PRIOR 'spikeslab' ####

# Extract the selected variables
# T1 data
ssp_T1_LD <- fit_spikeslab_prior(data = T1_LD, bigp_smalln = FALSE)
ssp_T1_ED <- fit_spikeslab_prior(data = T1_ED, bigp_smalln = FALSE)
```

```
ssp_T1_HD <- fit_spikeslab_prior(data = T1_HD, bigp_smalln = FALSE)
ssp_T1_VD <- fit_spikeslab_prior(data = T1_VD, bigp_smalln = TRUE,
                                 bigp_smalln_factor = 1, screen = TRUE)

# T2 data
ssp_T2_LD <- fit_spikeslab_prior(data = T2_LD, bigp_smalln = FALSE)
ssp_T2_ED <- fit_spikeslab_prior(data = T2_ED, bigp_smalln = FALSE)
ssp_T2_HD <- fit_spikeslab_prior(data = T2_HD, bigp_smalln = FALSE)
ssp_T2_VD <- fit_spikeslab_prior(data = T2_VD, bigp_smalln = TRUE,
                                 bigp_smalln_factor = 1, screen = TRUE)

# T3 data
ssp_T3_LD <- fit_spikeslab_prior(data = T3_LD, bigp_smalln = FALSE)
ssp_T3_ED <- fit_spikeslab_prior(data = T3_ED, bigp_smalln = FALSE)
ssp_T3_HD <- fit_spikeslab_prior(data = T3_HD, bigp_smalln = FALSE)
ssp_T3_VD <- fit_spikeslab_prior(data = T3_VD, bigp_smalln = TRUE,
                                 bigp_smalln_factor = 1, screen = TRUE)

# T4 data
ssp_T4_LD <- fit_spikeslab_prior(data = T4_LD, bigp_smalln = FALSE)
ssp_T4_ED <- fit_spikeslab_prior(data = T4_ED, bigp_smalln = FALSE)
ssp_T4_HD <- fit_spikeslab_prior(data = T4_HD, bigp_smalln = FALSE)
ssp_T4_VD <- fit_spikeslab_prior(data = T4_VD, bigp_smalln = TRUE,
                                 bigp_smalln_factor = 1, screen = TRUE)

#### SIM DATA. SPIKE-AND-SLAB LASSO 'SSLASSO' ####

# List of dataset prefixes
prefixes_ss <- c("T1", "T2")

# Call the function with the simulated data
# Initialize an empty list to store the models
models_sslasso_list <- list()

#   Loop over prefixes and suffixes
for (prefix in prefixes_ss) {
  for (suffix in suffixes) {
    # Construct the dataset name
    dataset_name <- paste(prefix, suffix, sep = "_")
```

```
    # Access the dataset from the global environment
    dataset <- get(dataset_name)

    # Fit the model
    model <- fit_sslasso(dataset)

    # Generate a model name
    model_name <- paste("ssl", dataset_name, sep = "_")

    # Store the model in the list
    models_sslasso_list[[model_name]] <- model
  }
}

#### SIM DATA. HORSESHOE PRIOR. 'horseshoe' ####

# Fit the models with Truncated Cauchy priors
# Call the function with the simulated data
# Initialize an empty list to store the models
models_hs_tc_list <- list()

#   Loop over prefixes and suffixes
for (prefix in prefixes) {
  for (suffix in suffixes) {
    # Construct the dataset name
    dataset_name <- paste(prefix, suffix, sep = "_")

    # Access the dataset from the global environment
    dataset <- get(dataset_name)

    # Fit the model
    model <- fit_hs_horseshoe(dataset, method.tau = "truncatedCauchy")

    # Generate a model name
    model_name <- paste("hs_tc", dataset_name, sep = "_")

    # Store the model in the list
    models_hs_tc_list[[model_name]] <- model
```

```
  }
}

# Fit the models with Half Cauchy priors
# Call the function with the simulated data
# Initialize an empty list to store the models
models_hs_hc_list <- list()

#   Loop over prefixes and suffixes
for (prefix in prefixes) {
  for (suffix in suffixes) {
    # Construct the dataset name
    dataset_name <- paste(prefix, suffix, sep = "_")

    # Access the dataset from the global environment
    dataset <- get(dataset_name)

    # Fit the model
    model <- fit_hs_horseshoe(dataset, method.tau = "halfCauchy")

    # Generate a model name
    model_name <- paste("hs_tc", dataset_name, sep = "_")

    # Store the model in the list
    models_hs_hc_list[[model_name]] <- model
  }
}

#### SIM DATA. HORSESHOE PRIOR 'bayesreg' ####

# Fit the models with horseshoe prior
# Call the function with the simulated data
# Initialize an empty list to store the models
models_bs_h_list <- list()

#   Loop over prefixes and suffixes
for (prefix in prefixes) {
  for (suffix in suffixes) {
    # Construct the dataset name
```

```
    dataset_name <- paste(prefix, suffix, sep = "_")

    # Access the dataset from the global environment
    dataset <- get(dataset_name)

    # Fit the model
    model <- fit_horseshoe_bs(dataset)

    # Generate a model name
    model_name <- paste("bs_h", dataset_name, sep = "_")

    # Store the model in the list
    models_bs_h_list[[model_name]] <- model
  }
}

# Fit the models with horseshoe prior
# Call the function with the simulated data
# Initialize an empty list to store the models
models_bs_hp_list <- list()

#   Loop over prefixes and suffixes
for (prefix in prefixes) {
  for (suffix in suffixes) {
    # Construct the dataset name
    dataset_name <- paste(prefix, suffix, sep = "_")

    # Access the dataset from the global environment
    dataset <- get(dataset_name)

    # Fit the model
    model <- fit_horseshoe_bs(dataset, prior = "hs+")

    # Generate a model name
    model_name <- paste("bs_hp", dataset_name, sep = "_")

    # Store the model in the list
    models_bs_hp_list[[model_name]] <- model
  }
```

```
}

#### SIM DATA. S5 'BayesS5' ####

# Set prefices
prefixes_s5 <- c("T1", "T2")

# Fit the model with data with only continuous variables
# Call the function with the simulated data
# Initialize an empty list to store the models
models_s5c_list <- list()

#   Loop over prefixes and suffixes
for (prefix in prefixes_s5) {
  for (suffix in suffixes) {
    # Construct the dataset name
    dataset_name <- paste(prefix, suffix, sep = "_")

    # Access the dataset from the global environment
    dataset <- get(dataset_name)

    # Fit the model
    model <- fit_S5(dataset)

    # Generate a model name
    model_name <- paste("s5c", dataset_name, sep = "_")

    # Store the model in the list
    models_s5c_list[[model_name]] <- model
  }
}

#### SIM DATA. BAYESIAN LASSO 'monomvn' ####

# Fit the model with data with only continuous variables
# Call the function with the simulated data
# Initialize an empty list to store the models
models_blasso_list <- list()
```

```
#   Loop over prefixes and suffixes
for (prefix in prefixes) {
  for (suffix in suffixes) {
    # Construct the dataset name
    dataset_name <- paste(prefix, suffix, sep = "_")

    # Access the dataset from the global environment
    dataset <- get(dataset_name)

    # Fit the model
    model <- fit_blasso(dataset)

    # Generate a model name
    model_name <- paste("blasso", dataset_name, sep = "_")

    # Store the model in the list
    models_blasso_list[[model_name]] <- model
  }
}

#### SOURCE CRIME DATA ####

# Source the file that contains the crime data
source("data_crime_raw.R")

#### CRIME. PENALISED REGRESSION 'glmnet' ####

# Run the LASSO function and extract the selected coefficients
crime_lasso <- fit_glmnet(data = df_t, alpha = 1)

# Run the elnet function and extract the selected coefficients
crime_elnet <- fit_glmnet(data = df_t, alpha = 0.5)

#### CRIME. XGBOOST ####

# Initial parameters for xgboost were already defined
# Run the function to extract XGBoost model and feature importances
crime_xgboost <- fit_xgb(data = df_t, xgb_cv = xgb_cv, xgb_grid = xgb_grid)
```

```r
#### CRIME. SPIKE AND SLAB PRIOR 'spikeslab' ####

# Fit spikeslab model
crime_spikeslab_prior <- fit_spikeslab_prior(data = df_t, bigp_smalln = FALSE)

#### CRIME. SPIKE AND SLAB LASSO 'SSLASSO' ####

# Call the function with the Crimes data
crime_ssl <- fit_sslasso(data = df_t, var = "fixed")

#### CRIME. HORSESHOE PRIOR 'horseshoe' ####

# Run the functions an extract the results
# Truncated Cauchy prior
crime_horseshoe_tc <- fit_hs_horseshoe(data = df_t, method.tau = "truncatedCauchy",
                                       method.sigma = "Jeffreys", burn = 1000,
                                       nmc = 5000, thin = 1, alpha = 0.05)

# Half Cauchy prior
crime_horseshoe_hc <- fit_hs_horseshoe(data = df_t, method.tau = "halfCauchy",
                                          method.sigma = "Jeffreys",
                                          burn = 1000, nmc = 5000,
                                          thin = 1, alpha = 0.05)

#### CRIME. HORSESHOE AND PLUS PRIOR 'bayesreg' ####

# Fit the model with "hs" prior
crime_hs_bs <- fit_horseshoe_bs(data = df_t, n.samples = 5000, burnin = 1000,
                                prior = "hs")


# Fit the model with "hs+" prior
crime_hsp_bs <- fit_horseshoe_bs(data = df_t, n.samples = 5000, burnin = 1000,
                                 prior = "hs+")

#### CRIME. S5 'BayesS5' ####

# Fit the model
crime_S5 <- fit_S5(data = df_t)
```

```
#### CRIME. BAYESIAN LASSO 'monomvn' ####

# Fit the model
crime_blasso <- fit_blasso(data = df_t)
```